Open Sound Control

Objects to manage values on an Open Sound Control port.

OscSend takes the first value of each buffersize and send it on an OSC port.

OscReceive creates and returns audio streams from the value in its input port.

The audio streams of these objects are essentially intended to be controls and can't be sent to the output soundcard.

Note: These objects are available only if pyo is built with OSC (Open Sound Control) support.

OscDataReceive

class **`OscDataReceive`**(*port*, *address*, *function*) [source]

Receives data values over a network via the Open Sound Control protocol.

Uses the OSC protocol to receive data values from other softwares or other computers. When a message is received, the function given at the argument *function* is called with the current address destination in argument followed by a tuple of values.

Parent: `PyoObject`

Args: port: int

Port on which values are received. Sender should output on the same port. Unlike OscDataSend object, there can be only one port per OscDataReceive object. Available at initialization time only.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: "/pitch"). There can be as many addresses as needed on a single port.

function: callable (can't be a list)

This function will be called whenever a message with a known address is received. there can be only one function per OscDataReceive object. Available at initialization time only.

Note: The header of the callable given at *function* argument must be in this form :

def my_func(address, *args):

...

The out() method is bypassed. OscDataReceive has no audio signal.

OscDataReceive has no *mul* and *add* attributes.

```
>>> s = Server().boot()
>>> s.start()
>>> def pp(address, *args):
...     print(address)
...     print(args)
>>> r = OscDataReceive(9900, "/data/test", pp)
>>> # Send various types
>>> a = OscDataSend("fissif", 9900, "/data/test")
>>> msg = [3.14159, 1, "Hello", "world!", 2, 6.18]
>>> a.send(msg)
>>> # Send a blob
>>> b = OscDataSend("b", 9900, "/data/test")
>>> msg = [[chr(i) for i in range(10)]]
>>> b.send(msg)
>>> # Send a MIDI noteon on port 0
>>> c = OscDataSend("m", 9900, "/data/test")
>>> msg = [[0, 144, 60, 100]]
>>> c.send(msg)
```

getAddresses() [source]

Returns the addresses managed by the object.

addAddress(*path*) [source]

Adds new address(es) to the object's handler.

Args: path: string or list of strings

New path(s) to receive from.

delAddress(*path*) [source]

Removes address(es) from the object's handler.

Args: path: string or list of strings

Path(s) to remove.

OscDataSend

class **OscDataSend**(*types*, *port*, *address*, *host='127.0.0.1'*) [source]

Sends data values over a network via the Open Sound Control protocol.

Uses the OSC protocol to share values to other softwares or other computers. Values are sent on the form of a list containing *types* elements.

Parent: `PyoObject`

Args: types: str

String specifying the types sequence of the message to be sent. Possible values are:

- "i": integer
- "h": long integer

- “f”: float
- “d”: double
- “s” ; string
- “b”: blob (list of chars)
- “m”: MIDI packet (list of 4 bytes: [midi port, status, data1, data2])
- “c”: char
- “T”: True
- “F”: False
- “N”: None (nil)

The string “ssfi” indicates that the value to send will be a list containing two strings followed by a float and an integer.

port: int

Port on which values are sent. Receiver should listen on the same port.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: ‘/pitch’).

host: string, optional

IP address of the target computer. The default, ‘127.0.0.1’, is the localhost.

Note: The out() method is bypassed. OscDataSend has no audio signal.
OscDataSend has no *mul* and *add* attributes.

```
>>> s = Server().boot()
>>> s.start()
>>> def pp(address, *args):
...     print(address)
...     print(args)
>>> r = OscDataReceive(9900, "/data/test", pp)
>>> # Send various types
>>> a = OscDataSend("fissif", 9900, "/data/test")
>>> msg = [3.14159, 1, "Hello", "world!", 2, 6.18]
>>> a.send(msg)
>>> # Send a blob
>>> b = OscDataSend("b", 9900, "/data/test")
>>> msg = [[chr(i) for i in range(10)]]
>>> b.send(msg)
>>> # Send a MIDI noteon on port 0
>>> c = OscDataSend("m", 9900, "/data/test")
>>> msg = [[0, 144, 60, 100]]
>>> c.send(msg)
```

getAddresses() [source]

Returns the addresses managed by the object.

addAddress(*types*, *port*, *address*, *host='127.0.0.1'*) [source]

Adds new address(es) to the object’s handler.

Args: types: str

String specifying the types sequence of the message to be sent. Possible values are: - “i”: integer - “h”: long integer - “f”: float - “d”: double - “s” ; string - “b”: blob (list of chars) - “m”: MIDI packet (list of 4 bytes: [midi port, status, data1, data2]) - “c”: char - “T”: True - “F”: False - “N”: None (nil)
The string “ssfi” indicates that the value to send will be a list containing two strings followed by a float and an integer.

port: int

Port on which values are sent. Receiver should listen on the same port.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: ‘/pitch’).

host: string, optional

IP address of the target computer. The default, ‘127.0.0.1’, is the localhost.

`delAddress`(*path*) [source]

Removes address(es) from the object’s handler.

Args: path: string or list of strings

Path(s) to remove.

`send`(*msg*, *address=None*) [source]

Method used to send *msg* values as a list.

Args: msg: list

List of values to send. Types of values in list must be of the kind defined of *types* argument given at the object’s initialization.

address: string, optional

Address destination to send values. If None, values will be sent to all addresses managed by the object.

OscListReceive

class **`OscListReceive`**(*port*, *address*, *num=8*, *mul=1*, *add=0*) [source]

Receives list of values over a network via the Open Sound Control protocol.

Uses the OSC protocol to receive list of floating-point values from other softwares or other computers. The list are converted into audio streams. Get values at the beginning of each buffersize and fill buffers with them.

Parent: `PyoObject`

Args: port: int

Port on which values are received. Sender should output on the same port. Unlike OscSend object, there can be only one port per OscListReceive object. Available at initialization time only.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: '/pitch').

num: int, optional

Length of the lists in input. The object will generate *num* audio streams per given address. Available at initialization time only. This value can't be a list. That means all addresses managed by an OscListReceive object are of the same length. Defaults to 8.

Note: Audio streams are accessed with the *address* string parameter. The user should call : OscReceive['/pitch'] to retreive list of streams named '/pitch'.

The out() method is bypassed. OscReceive's signal can not be sent to audio outs.

```
>>> s = Server().boot()
>>> s.start()
>>> # 8 oscillators
>>> a = OscListReceive(port=10001, address=['/pitch', '/amp'], num=8)
>>> b = Sine(freq=a['/pitch'], mul=a['/amp']).mix(2).out()
```

`getAddresses()` [source]

Returns the addresses managed by the object.

`addAddress`(*path*, *mul=1*, *add=0*) [source]

Adds new address(es) to the object's handler.

Args: path: string or list of strings

New path(s) to receive from.

mul: float or PyoObject

Multiplication factor. Defaults to 1.

add: float or PyoObject

Addition factor. Defaults to 0.

`delAddress`(*path*) [source]

Removes address(es) from the object's handler.

Args: path: string or list of strings

Path(s) to remove.

setInterpolation(*x*) [source]

Activate/Deactivate interpolation. Activated by default.

Args: x: boolean

True activates the interpolation, False deactivates it.

setValue(*path*, *value*) [source]

Sets value for a given address.

Args: path: string

Address to which the value should be attributed.

value: list of floats

List of values to attribute to the given address.

get(*identifier=None*, *all=False*) [source]

Return the first list of samples of the current buffer as floats.

Can be used to convert audio stream to usable Python data.

Address as string must be given to *identifier* to specify which stream to get value from.

Args: identifier: string

Address string parameter identifying audio stream. Defaults to None, useful when *all* is True to retreive all streams values.

all: boolean, optional

If True, the first list of values of each object's stream will be returned as a list of lists. Otherwise, only the the list of the object's identifier will be returned as a list of floats. Defaults to False.

OscReceive

class **OscReceive**(*port*, *address*, *mul=1*, *add=0*) [source]

Receives values over a network via the Open Sound Control protocol.

Uses the OSC protocol to receive values from other softwares or other computers. Get a value at the beginning of each buffersize and fill its buffer with it.

Parent: `PyoObject`

Args: port: int

Port on which values are received. Sender should output on the same port. Unlike OscSend object, there can be only one port per OscReceive object. Available at initialization time only.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: '/pitch').

Note: Audio streams are accessed with the *address* string parameter. The user should call : OscReceive['/pitch'] to retreive stream named '/pitch'.

The out() method is bypassed. OscReceive's signal can not be sent to audio outs.

```
>>> s = Server().boot()
>>> s.start()
>>> a = OscReceive(port=10001, address=['/pitch', '/amp'])
>>> b = Sine(freq=a['/pitch'], mul=a['/amp']).mix(2).out()
```

`getAddresses`() [source]

Returns the addresses managed by the object.

`addAddress`(*path*, *mul=1*, *add=0*) [source]

Adds new address(es) to the object's handler.

Args: path: string or list of strings

New path(s) to receive from.

mul: float or PyoObject

Multiplication factor. Defaults to 1.

add: float or PyoObject

Addition factor. Defaults to 0.

`delAddress`(*path*) [source]

Removes address(es) from the object's handler.

Args: path: string or list of strings

Path(s) to remove.

`setInterpolation`(*x*) [source]

Activate/Deactivate interpolation. Activated by default.

Args: x: boolean

True activates the interpolation, False deactivates it.

`setValue`(*path*, *value*) [source]

Sets value for a given address.

Args: path: string

Address to which the value should be attributed.

value: float

Value to attribute to the given address.

`get`(*identifier=None*, *all=False*) [source]

Return the first sample of the current buffer as a float.

Can be used to convert audio stream to usable Python data.

Address as string must be given to *identifier* to specify which stream to get value from.

Args: identifier: string

Address string parameter identifying audio stream. Defaults to None, useful when *all* is True to retreive all streams values.

all: boolean, optional

If True, the first value of each object's stream will be returned as a list. Otherwise, only the value of the first object's stream will be returned as a float. Defaults to False.

OscSend

class `OscSend`(*input*, *port*, *address*, *host='127.0.0.1'*) [source]

Sends values over a network via the Open Sound Control protocol.

Uses the OSC protocol to share values to other softwares or other computers. Only the first value of each input buffersize will be sent on the OSC port.

Parent: `PyoObject`

Args: input: PyoObject

Input signal.

port: int

Port on which values are sent. Receiver should listen on the same port.

address: string

Address used on the port to identify values. Address is in the form of a Unix path (ex.: '/pitch').

host: string, optional

IP address of the target computer. The default, '127.0.0.1', is the localhost.

Note: The out() method is bypassed. OscSend's signal can not be sent to audio outs. OscSend has no *mul* and *add* attributes.

```
>>> s = Server().boot()
>>> s.start()
>>> a = Sine(freq=[1,1.5], mul=[100,.1], add=[600, .1])
>>> b = OscSend(a, port=10001, address=['/pitch','/amp'])
```

`setInput`(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Defaults to 0.05.

`setBufferRate`(*x*) [source]

Sets how many buffers to wait before sending a new value.

Args: x: int

Changes the data output frequency in multiples of the buffer size. Should be greater or equal to 1.

`input`

PyoObject. Input signal.