

---

# **pastream Documentation**

***Release***

**Author**

**Nov 24, 2017**



---

## Contents

---

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Dependencies</b>	<b>5</b>
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Building From Source</b>	<b>9</b>
<b>5</b>	<b>Building Documentation</b>	<b>11</b>
<b>6</b>	<b>Examples</b>	<b>13</b>
<b>7</b>	<b>Command Line Application</b>	<b>15</b>
<b>8</b>	<b>API Reference</b>	<b>17</b>
<b>9</b>	<b>Release Notes</b>	<b>19</b>



`pastream` builds on top of [portaudio](#) and the excellent [sounddevice](#) python bindings to provide some more advanced functionality right out of the box. Note that in addition to the `pastream library`, `pastream` includes a *command line application* for playing and recording audio files.



# CHAPTER 1

---

## Features

---

**GIL-less Audio Callbacks** Having the portaudio callback implemented in C means audio interrupts can be serviced quickly and reliably without ever needing to acquire the Python Global Interpreter Lock (GIL). This is crucial when working with libraries like [Pillow](#) which may greedily grab and hold the GIL subsequently causing audio overruns/underruns.

**Input Stream iterators** Efficiently retrieve live audio capture data through an iterable. As simple as:

```
import pastream as ps
for chunk in ps.chunks():
    process(chunk)
```

See `pastream.chunks` and `pastream.InputStream.chunks` method.

**Reader/Writer Threads** `pastream` simplifies the process of implementing stream reader and writer threads to manipulate and/or generate data in the background while leaving the main thread free for higher level management tasks.





## CHAPTER 2

---

### Dependencies

---

`cffi`

`sounddevice` (depends on `PortAudio`)

`soundfile` (depends on `libsndfile`)

(Optional) `numpy`



## CHAPTER 3

---

### Installation

---

For linux platforms a recent version of the `PortAudio` and `libsndfile` C libraries are required. (For Windows and OSX, the `sounddevice` and `soundfile` packages include prebuilt versions for you). You can either install the latest available from your package manager (e.g. `apt-get install libportaudio2 libsndfile` for debian/raspbian) or install the latest stable build from the package website (Recommended); see links in [Dependencies](#).

`pastream` is now available on PyPI. Installation is as easy as:

```
$ pip install pastream
```



## CHAPTER 4

---

### Building From Source

---

To compile from source under unix platforms, `libffi` is required. (For Windows, this is already included with `cffi`). `libffi` is available through most package managers (e.g., `yum install libffi-devel`, `apt-get install libffi-dev`, `brew install libffi`). More information on installing `libffi` is available [here](#).

If doing a fresh checkout:

```
$ git clone --recursive http://github.com/tgarc/pastream
```

If you already have a checkout:

```
$ git submodule update --init
```

Then do a pip install from your working copy:

```
$ pip install <path/to/checkout>
```



---

### Building Documentation

---

Documentation for pastream can be easily generated in a wide variety of formats using Sphinx. Just follow the steps below.

Checkout the repository:

```
$ git clone --recursive http://github.com/tgarc/pastream
```

Then use the included makefile/make.bat to generate documentation. (Here we output to the html format):

```
$ cd pastream/docs  
$ make html
```





## CHAPTER 6

---

### Examples

---

Record 1000 frames to file, then play it back:

```
import pastream as ps

# Use *with* statements to auto-close the stream
with ps.SoundFileInputStream('recording.wav') as stream:
    stream.frames = 1000
    stream.start()
    stream.wait() # Block until recording is done

with ps.SoundFileOutputStream('recording.wav') as stream:
    stream.frames = 1000
    stream.start()
    stream.wait()
```

Grab (real) frequency transformed live audio stream with 50% overlap:

```
import pastream as ps, numpy as np

chunksize = 1024
window = np.hanning(chunksize)
for x_l in ps.chunks(chunksize, overlap=chunksize//2, channels=1):
    X_l = np.fft.rfft(x_l * window)
```

See also the included examples under `/examples`.



## CHAPTER 7

---

### Command Line Application

---

Once installed, the `pastream` application should be callable from your command line. If you're familiar with `sox` you'll notice that some of the command line syntax is quite similar. Here are a few examples to help get you started.

Display the help file:

```
$ pastream -h
```

List available audio devices:

```
$ pastream -l
```

Simultaneous play and record from the default audio device:

```
$ pastream input.wav output.wav
```

Pipe input from `sox` using the AU format:

```
$ sox -n -t au - synth sine 440 | pastream - output.wav
```

Play a RAW file:

```
$ pastream -c1 -r48k -e=pcm_16 -o output.raw
```

Record 10 seconds of audio at 48kHz:

```
$ pastream null output.wav -r48k -n=480k
```



## CHAPTER 8

---

### API Reference

---



#### 0.0.8:

- BUG: fixed possible bad behavior when `pad >= 0 frames < 0` (06881)
- BUG: `pad > 0` can cause too many frame reads (fixed in e917e)
- Receive buffer is no longer automatically flushed when calling `start()` (cd65b)
- BUG: `AttributeError` was not correctly being caught and reraised in stream threads (3bc5e)
- Added sphinx documentation (11c13)
- `frames` attribute changed from `long` to `long long` (ee4ebb)
- `chunks`: eliminated an unnecessary copy when using `overlap` (b0304)

#### 0.0.7:

- add `-loop` option to the CLI to allow looping playback.
- allow empty string as an alternative to null
- Raise exception when attempting to open stream with RAW playback file if any of `samplerate/channels/subtype` are not specified.
- change prebuffering behavior slightly: only wait until the first write, not until the buffer fills up. This should avoid potential long pre-buffer times
- fix formatting errors in `__repr__` when using multiple dtypes and/or devices
- no need to vendor `pa_ringbuffer` anymore, it's available on pip! (Thanks @mgeier !)
- if a `SoundFile inpf` is passed to a `SoundFileInputStream` class, it will be used to set the stream `samplerate/channels`.
- addresses a bug when `BUFFERSIZE < 8192`
- `Stream` and `SoundFileStream` classes renamed to *\*DuplexStream*
- Swapped assignments of input/output in `SoundFileStreams` to make it align with the usage in the rest of the library. The order of input/output arguments from the CLI still stays the same though.

- remove *allow\_drops* parameters. It can be added back at a later point if it proves to be a more useful feature

#### 0.0.6:

- fix 'null' not properly matching on cmdline
- chunks: check that portaudio has not been terminated before trying to close/stop a stream
- drop *allow\_xruns/XRunError*
- *Buffered\*Stream* -> *\*Stream*
- *\*Buffer{Empty,Full}* -> *Buffer{Empty,Full}*
- fix remaining issues with wheel building
- Dropped unused exception classes (*PaStreamError*, *AudioBufferError*)
- Added prebuffer argument to *start()* to bypass filling output buffer before stream starts

#### 0.0.5:

- Redirect *sys.stdout* to *devnull* when '-' is used as the output file stream
- Specifying multiple *--file-type s* at command line fixed
- *--format* now only accepts a single argument
- *ringbuffersize\_t* is of a different type for mac platforms; fixed
- *ps.chunks()* README example fixed
- **frames is now a signed value. The behavior previously reserved for** *frames == 0* now is active whenever *frames < 0*
  - Comma separated arguments are no longer allowed; multiple argument options can only be specified by passing them multiple times
  - dropped support for passing a bool for *pad* parameter
  - *-q* flag for specifying buffersize has been dropped. This is now reserved for the new *--quiet* option.
- add a loopback test for the pastream app using *stdin > stdout*
- improvement: *chunks* function: make sure that stream is closed properly without the performance hit of having an extra yield
- new feature: If both *padding* and *frames* are *< 0*, padding will be added indefinitely
- new feature: *-q/--quiet* option; this drops the deprecated *-q* option for specifying buffersize

#### 0.0.4:

- bugfix: chunks: overlap was (accidentally) not allowed if chunksize was not non-zero. This should be allowed as long as *stream.blocksize > 0*.
- chunks now supports passing a generic ndarray to *out* parameter (without having to cast it to a bytes object)
- *nframes* renamed to *frames*
- *padding* renamed to *pad*
- added *allow\_drops* option to give user the option to ignore *ReceiveBufferEmpty* error in more atypical use cases
- *raise\_on\_xruns* changed to *allow\_xruns*; inverted behavior



- got rid of undocumented `keep_alive` option; the combination of `allow_drops` and `pad` can give the same functionality
- `--pad` now can be specified without an argument which just sets `pad` to `True`
- added autopadding feature: Now if `frames > 0` and `pad == True` or `pad < 0`, playback will be zero padded out to `frames`. This is a nice feature for the `pastream` application and `SoundFileStream` since sometimes you want to add extra padding after the file playback.

**0.0.3:**

- command line options for size parameters now accept `k/K/m/M` suffix
- Backwards compatibility break: multiple argument command line options now accept a comma delimited list
- improved `SoundFileStream` reader writers; nearly zero read/write misses
- bugfix: `__repr__` had a bug for certain cases

**0.0.2:**

- Improved `SoundFileStream` interface: remove `sfkwards`; instead `format`, `endian`, and `subtype` can be passed directly since they don't collide with any of the `sounddevice` parameters
- Updated examples to allow half or full duplex operation. Also accepts `subtype` for RAW files
- `chunks()` updates \* better polling behavior greatly decreases read misses \* now supports generic buffers so `numpy` is not required \* added `out` option to allow user to pass a preallocated buffer \* bugfix: overlap was not overlapping correctly
- MAJOR bugfix: `samplerate` was not being properly passed up the class chain
- MAJOR bugfix: `lastTime` was not being properly copied in `py_pastream.c` so the value returned was garbage
- bugfix: `assert_chunks_equal`: the 'inframes' buffer was not being allocated enough space for when `chunk-size > blocksize` which was causing mismatch hysteria

**0.0.1:**

- First tenable release