pyzmq-wrapper Documentation

Release 0.1.1b

Hari (http://github.com/supercoderz)

Sep 18, 2017

Contents

pyzmq-wrapper is a set of classes that provide wrappers over the zmq code - these aim to serve the 80% cases where the developers just want to start a publisher or subscriber withut having to worry about how the zmq connections are made or what flags are to be used for each message. The publishers and the subscribers created by pyzmq-wrapper use the default values for all flags. The only assumption made is that the subscribers will be in non-blocking mode, so that you can use the subscriber in a thread and can shut it down easily. Also the subscriber provides a callback - freeing you from having to loop through the receiving process.

The sock() method on the publishers and subscribers gives access to the underlying zmq socket object so that you can use that for sending messages which require you to set certain flags.

CHAPTER 1

Installation

You can install this using one of the following:

```
pip install pyzmq-wrapper

easy_install pyzmq-wrapper
```

CHAPTER 2

Getting started

The following entities are provided to abstract the user from the zmq code.

Publisher

A simple pulisher can be started on a port as shown below. This object can then be used to send messages - with or without a topic:

```
from zmqwrapper import *

p=publisher('tcp://127.0.0.1:5555')
p.publish("hello",RAW)
#greeting is the topic
p.publish("hello",MULTIPART,"greeting")
```

Subscriber

A simple subscriber can be created by passing a port, a list of topics, a callback and message type. The callback is executed for every message that is received on the topic:

```
from zmqwrapper import *

#define the callback
def process_greeting(topic,message):
    print message


#create the subscriber
s=subscriber('tcp://127.0.0.1:5555',['greetings'],process_greeting,MULTIPART)
#and start it so that we can process the messages
s.start()
```

Requestor

A simple requestor can be created by passing in the address, callback for replies and the message type. The callback receives the requestor again so that you can make additional requests if needed:

```
from zmqwrapper import *

def foo(message,requestor):
    print message

rq=requestor('tcp://127.0.0.1:5555',foo,JSON)
rq.start()
rq.request('test message',JSON)
```

Replier

A simple replier can be created by passing in the address, callback for requests and the message type. The callback receives the replier so that you can send back he response:

```
from zmqwrapper import *
import time

def foo(message,replier):
    print message
    replier.reply(message)

rp=replier('tcp://127.0.0.1:5555',foo,JSON)
rp.start()
time.sleep(5)
```

Producer

A simple producer can be started on a port as shown below. This object can then be used to push messages:

```
from zmqwrapper import *

p=producer('tcp://127.0.0.1:5555')
p.push("hello",RAW)
```

Consumer

A simple consumer can be created by passing a port,a callback and message type. The callback is executed for every message that is received on the topic:

```
from zmqwrapper import *

#define the callback
def process_greeting(message):
    print message
```

```
#create the consumer
c=consumer('tcp://127.0.0.1:5555',process_greeting,RAW)
#and start it so that we can process the messages
c.start()
```

CHAPTER 3

API Docs

zmqwrapper Package

zmqwrapper Package

constants Module

The message types that are supported by the package are -

- RAW
- PYOBJ
- JSON
- MULTIPART
- STRING
- UNICODE

This map to the same types supported by pyzmq

consumers Module

class zmqwrapper.consumers.**Consumer**(*address*, *callback*, *message_type*)
Bases: *zmqwrapper.sockets.ClientConnection*

Requestor that that can send requests of given type

Args:

- address: the address to bind to
- callback: the callback to invoke for every reply
- message_type: the type of request to send

start()
Start a thread that consumes the replies and invokes the callback

stop()
Stop the consumer thread

zmqwrapper.consumers.**consumer**(*address*, *callback*, *message_type*)
Creates a consumer binding to the given address pull messages. The callback is invoked for every reply received.

Args:

- address: the address to bind the PULL socket to.
- callback: the callback to invoke for every message. Must accept 1 variables - the message
- message_type: the type of message to receive

producers Module

class zmqwrapper.producers.**Producer**(*address*)
Bases: *zmqwrapper.sockets.ServerConnection*

Requestor that that can respond to requests of given type

Args:

- address: the address to bind to

push(*message*, *message_type*)
Send a reply message of the given type

Args:

- message: the message to publish
- message_type: the type of message being sent

zmqwrapper.producers.**producer**(*address*)
Creates a producer binding to the given address push messages. The callback is invoked for every request received.

Args:

- address: the address to bind the PUSH socket to.

publishers Module

class zmqwrapper.publishers.**Publisher**(*address*)
Bases: *zmqwrapper.sockets.ServerConnection*

Publisher that can send messages to ZMQ

Args:

- address: the address to bind to

publish(*message*, *message_type*, *topic=''*)
Publish the message on the PUB socket with the given topic name.

Args:

- message: the message to publish
- message_type: the type of message being sent

- topic: the topic on which to send the message. Defaults to ‘’.

zmqwrapper.publishers.**publisher**(*address*)
Creates a publisher binding to the given port number.

Args:

- address: the address to bind the PUB socket to.

repliers Module

class zmqwrapper.repliers.**Replier**(*address*, *callback*, *message_type*)
Bases: *zmqwrapper.sockets.ServerConnection*

Requestor that that can respond to requests of given type

Args:

- address: the address to bind to
- callback: the callback to invoke for every request
- message_type: the type of reply to send

reply(*message*, *message_type*)
Send a reply message of the given type

Args:

- message: the message to publish
- message_type: the type of message being sent

start()
Start a thread that consumes the requests and invokes the callback

stop()
Stop the consumer thread

zmqwrapper.repliers.**replier**(*address*, *callback*, *message_type*)
Creates a replier binding to the given address send replies. The callback is invoked for every request received.

Args:

- address: the address to bind the REP socket to.
- callback: the callback to invoke for every message. Must accept 2 variables - message and the replier
- message_type: the type of message to receive

requestors Module

class zmqwrapper.requestors.**Requestor**(*address*, *callback*, *message_type*)
Bases: *zmqwrapper.sockets.ClientConnection*

Requestor that that can send requests of given type

Args:

- address: the address to bind to
- callback: the callback to invoke for every reply

- message_type: the type of request to send

request(*message*, *message_type*)
Send a request message of the given type

Args:

- message: the message to publish
- message_type: the type of message being sent

start()
Start a thread that consumes the replies and invokes the callback

stop()
Stop the consumer thread

zmqwrapper.requestors.**requestor**(*address*, *callback*, *message_type*)
Creates a requestor binding to the given address send requests. The callback is invoked for every reply received.

Args:

- address: the address to bind the REQ socket to.
- callback: the callback to invoke for every message. Must accept 2 variables - message and the requestor
- message_type: the type of message to receive

sockets Module

class zmqwrapper.sockets.**ClientConnection**(*address*, *socket_type*)
Bases: *zmqwrapper.sockets.SendReceiveMixin*, object

Creates a client side socket of given type.

Args:

- address: the address to use
- socket_type: the tyoe of socket to open

close()
Close the socket connection.

sock()
Returns the zmq socket object being used for the connection. This can be used to receive messages with additional flags etc.

Returns:

- The underlying zmq socket

class zmqwrapper.sockets.**SendReceiveMixin**
Provides send or receive functionality for the sockets

receive(*message_type*)
Receive the message of the specified type and retun

Args:

- message_type: the type of the message to receive

Returns:

- the topic of the message

- the message received from the socket

send(*message*, *message_type*, *topic=''*)
Send the message on the socket.

Args:

- message: the message to publish
- message_type: the type of message being sent
- topic: the topic on which to send the message. Defaults to ''.

class zmqwrapper.sockets.**ServerConnection**(*address*, *socket_type*)
Bases: *zmqwrapper.sockets.SendReceiveMixin*, object

Creates a server side socket of given type.

Args:

- address: the address to use
- socket_type: the tyoe of socket to open

close()
Close the socket connection.

sock()
Returns the zmq socket object being used for the connection. This can be used to send messages with additional flags etc.

Returns:

- The underlying zmq socket

subscribers Module

class zmqwrapper.subscribers.**Subscriber**(*address*, *topics*, *callback*, *message_type*)
Bases: *zmqwrapper.sockets.ClientConnection*

Subscriber that can read messages from ZMQ

Args:

- address: the address to bind to
- topics: the topics to subscribe
- callback: the callback to invoke for every message
- message_type: the type of message to receive

start()
Start a thread that consumes the messages and invokes the callback

stop()
Stop the consumer thread

zmqwrapper.subscribers.**subscriber**(*address*, *topics*, *callback*, *message_type*)
Creates a subscriber binding to the given address and subscribe the given topics. The callback is invoked for every message received.

Args:

- address: the address to bind the PUB socket to.

- topics: the topics to subscribe
- callback: the callback to invoke for every message. Must accept 2 variables - topic and message
- message_type: the type of message to receive

test_pubsub Module

zmqwrapper.test_pubsub.**test_init_basic_subscribe**()

zmqwrapper.test_pubsub.**test_init_publisher**()

zmqwrapper.test_pubsub.**test_init_subscriber**()

test_pushpull Module

zmqwrapper.test_pushpull.**test_init_basic_consume**()

zmqwrapper.test_pushpull.**test_init_consumer**()

zmqwrapper.test_pushpull.**test_init_producer**()

test_reqrep Module

zmqwrapper.test_reqrep.**test_init_basic_reqrep**()

zmqwrapper.test_reqrep.**test_init_replier**()

zmqwrapper.test_reqrep.**test_init_requestor**()

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

z

Index