socket_zmq Documentation

Release 0.2.9

Lipin Dmitriy

November 11, 2013

Contents

Listen tcp socket, parse Thrift framed protocol and distribute message between workers.

CHAPTER 1

Reference Docs

1.1 API Documentation

The API documentation is organized alphabetically by module name.

1.1.1 thriftworker Package

thriftworker Package

Process incoming request and return it's result.

app Module

ThriftWorker.

Distribute thrift requests between workers.

class thriftworker.app.**ThriftWorker**(*loop=None*, *protocol_factory=None*, *port_range=None*, *pool_size=None*, *shutdown_timeout=None*)

Bases: thriftworker.utils.mixin.SubclassMixin

Store global application state. Also acts as factory for whole application.

Acceptor

Acceptors

Hub

Create bounded Hub class.

Listener

Create bounded Listener class.

Listeners

Create bounded Listeners class.

Services

Create bounded Processor class.

Worker

acceptor_cls = 'thriftworker.transports.framed:FramedAcceptor'

acceptors
Create pool of acceptors.

classmethod default()
Return default application instance.

hub
Instance of bounded `LoopContainer`.

static instance()
Return global instance.

listeners
Create pool of listeners.

loop
Create event loop. Should be running in separate thread.

pool_size
Return default pool size.

port_range
Return range from which we allowed to allocate ports.

protocol_factory
Specify which protocol should be used.

services
Create global request processor instance.

worker
Create some worker routine.

worker_cls

constants Module

All constants for this package.

exceptions Module

All exceptions for this package.

exception `thriftworker.exceptions.`**AllocationError**
Bases: `exceptions.Exception`

Raised if we can't bind to any address from pool.

exception `thriftworker.exceptions.`**BindError**
Bases: `exceptions.Exception`

Error on socket binding.

hub Module

listener Module

class thriftworker.listener.**Listener**(*name*, *address*, *backlog=None*)
Bases: thriftworker.utils.mixin.LoopMixin

channel
Wrapper for socket. Needed to pass descriptor to child process.

host
Return host to which this socket is binded.

port
Return binded port number.

socket
A shortcut to create a TCP socket and bind it.

start
Bind listener to given port.

stop

class thriftworker.listener.**Listeners**
Bases: thriftworker.utils.mixin.LoopMixin

Maintain pool of listeners.

Listener
Shortcut to thriftworker.listener.Listener class.

channels
Return list of registered channels. Useful to pass them to child process.

enumerated
Return enumerated mapping of listeners.

register(*name*, *host*, *port*, *backlog=None*)
Register new listener with given parameters.

services Module

Store processor and protocol for each service.

class thriftworker.services.**Services**
Bases: object

Process new requests and return response. Store processor for each service.

class Service
Bases: tuple

Holder of service processor and protocol factory.

processor
Alias for field number 0

proto_factory
Alias for field number 1

Services.**app = None**

Services.**create_processor**(*service_name*)
Create function that will process incoming request and return payload that we should return.

Parameters **service_name** – name of served service

Services.**register**(*service_name*, *processor*, *proto_factory=None*)
Register new processor for given service.

state Module

Handle global package state.

thriftworker.state.**get_current_app**()

thriftworker.state.**set_current_app**(*app*)

Subpackages

envs Package

_event Module

base Module

green Module

sync Module

tests Package

tests Package

utils Module

class thriftworker.tests.utils.**CustomAppMixin**
Bases: object

Create application with custom loop.

setUp()

wait_for_predicate(*func*, *timeout=5.0*)

class thriftworker.tests.utils.**GreenMeta**
Bases: type

class thriftworker.tests.utils.**GreenTest**(*methodName='runTest'*)
Bases: thriftworker.tests.utils.CustomAppMixin, thriftworker.tests.utils.NewBase

Ensure that all methods will be executed in loop.

setUp()

class thriftworker.tests.utils.**StartStopLoopMixin**
Bases: thriftworker.tests.utils.CustomAppMixin

Ensure that hub will started be started before tests run.

setUp()

wakeup_loop()

class thriftworker.tests.utils.**TestCase**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Default test case that reset current application on each run.

setUp()

thriftworker.tests.utils.**start_stop_ctx**(**args*, ***kwds*)
Start container on enter and stop on exit to and from context.

Subpackages

app Package

test_app Module

class thriftworker.tests.app.test_app.**TestApp**(*methodName='runTest'*)
Bases: thriftworker.tests.utils.TestCase

test_current_app()

test_custom_loop()

test_custom_pool_size()

test_custom_port_range()

test_custom_proto_factory()

test_default_app()

test_negative_pool_size()

test_wrong_port_range()

test_hub Module

exception thriftworker.tests.app.test_hub.**ExpectedError**
Bases: exceptions.Exception

class thriftworker.tests.app.test_hub.**TestGreenlet**(*methodName='runTest'*)
Bases: thriftworker.tests.utils.GreenTest

assertKilled(*g*)

test_error_exit(**args*, ***kwargs*)

test_kill_just_started_block(**args*, ***kwargs*)

test_kill_just_started_noblock(**args*, ***kwargs*)

test_kill_not_started_block(**args*, ***kwargs*)

test_kill_not_started_noblock(**args*, ***kwargs*)

test_kill_running_block(**args*, ***kwargs*)

test_kill_running_noblock(**args*, ***kwargs*)

test_simple_exit(**args*, ***kwargs*)

class thriftworker.tests.app.test_hub.**TestHub**(*methodName='runTest'*)
Bases: thriftworker.tests.utils.CustomAppMixin, thriftworker.tests.utils.TestCase

context()

setUp()

test_start_stop()

test_wakeup()

test_listener Module

class thriftworker.tests.app.test_listener.**ListenerMixin**
Bases: thriftworker.tests.utils.StartStopLoopMixin

Listener = None

setUp()

class thriftworker.tests.app.test_listener.**ListenersMixin**
Bases: thriftworker.tests.utils.StartStopLoopMixin

Listeners = None

setUp()

class thriftworker.tests.app.test_listener.**TestListener**(*methodName='runTest'*)
Bases: thriftworker.tests.app.test_listener.ListenerMixin, thriftworker.tests.utils.TestCase

class Listener(*name*, *address*, *backlog=None*)
Bases: thriftworker.utils.mixin.LoopMixin

channel
Wrapper for socket. Needed to pass descriptor to child process.

host
Return host to which this socket is binded.

port
Return binded port number.

socket
A shortcut to create a TCP socket and bind it.

start
Bind listener to given port.

stop

TestListener.**test_bind_error**()

TestListener.**test_bind_from_pool**()

TestListener.**test_start_stop**()

class thriftworker.tests.app.test_listener.**TestListeners**(*methodName='runTest'*)
Bases: thriftworker.tests.app.test_listener.ListenersMixin, thriftworker.tests.utils.TestCase

class Listeners
Bases: thriftworker.utils.mixin.LoopMixin

Maintain pool of listeners.

Listener
Shortcut to `thriftworker.listener.Listener` class.

channels
Return list of registered channels. Useful to pass them to child process.

enumerated
Return enumerated mapping of listeners.

register(*name*, *host*, *port*, *backlog=None*)
Register new listener with given parameters.

`TestListeners.`**test_register**()

test_services Module

test_state Module

class `thriftworker.tests.app.test_state.`**TestState**(*methodName='runTest'*)
Bases: `thriftworker.tests.utils.TestCase`

test_change_app()

test_current_app()

test_no_app_created()

transports Package

test_base Module

test_framed Module

utils Module

utilities Package

test_atomics Module

class `thriftworker.tests.utilities.test_atomics.`**TestAtomicInteger**(*methodName='runTest'*)
Bases: `thriftworker.tests.utils.TestCase`

setUp()

test_add()

test_cmp()

test_decr()

test_get()

test_incr()

test_props()

test_repr()

test_set()

test_sub()

test_decorators Module

class thriftworker.tests.utilities.test_decorators.**TestOther**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

test_cached_property()

test_cached_property_access()

test_finalize Module

class thriftworker.tests.utilities.test_finalize.**TestOther**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

test_out_of_scope()

test_reset()

test_imports Module

test_loop Module

test_other Module

class thriftworker.tests.utilities.test_other.**TestOther**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

test_addresses_from_pool_auto()

test_addresses_from_pool_digit()

test_addresses_from_pool_wrong()

test_port_from_range()

test_rgetattr()

thriftworker.tests.utilities.test_other.**skip**(*g*, *num=1*)

test_proxy Module

class thriftworker.tests.utilities.test_proxy.**TestPromiseProxy**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

test_maybe_evaluate()

test_only_evaluated_once()

class thriftworker.tests.utilities.test_proxy.**TestProxy**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

test_bool()

test_call()

test_context()

test_dictproxy()

test_dir()

test_getsetdel_attr()

test_hash()

test_int()

test_listproxy()

test_name()

test_reduce()

test_slots()

test_unicode()

Subpackages

stats Package

test_counter Module

class thriftworker.tests.utilities.stats.test_counter.**TestCounter**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

setUp()

test_add()

test_add_sample()

test_count()

test_iadd()

test_timer Module

class thriftworker.tests.utilities.stats.test_timer.**TestTimer**(*methodName='runTest'*)

Bases: thriftworker.tests.utils.TestCase

setUp()

test_add()

test_add_sample()

test_count()

test_iadd()

workers Package

test_base Module

test_green Module

test_sync Module

test_threads Module

utils Module

transports Package

base Module

class thriftworker.transports.base.**Acceptors**

Bases: thriftworker.utils.mixin.StartStopMixin, thriftworker.utils.mixin.LoopMixin

Maintain pool of acceptors. Start them when needed.

Acceptor
Shortcut to thriftworker.acceptor.Acceptor class.

connections_number
Return current connection number across all acceptors.

empty
Are all acceptors empty or not.

register(*fd*, *name*, *backlog=None*)
Register new acceptor in pool.

start_accepting()
Start all registered acceptors if needed.

start_by_name(*name*)
Start acceptor by name.

stop()
Close all registered acceptors.

stop_accepting(*callback=None*)
Stop all registered acceptors if needed.

stop_by_name(*name*)
Stop acceptor by name.

class thriftworker.transports.base.**BaseAcceptor**(*name*, *descriptor*, *backlog=None*)

Bases: abc.NewBase

Accept incoming connections.

Connection
Return connection class. Depends on current implementation of transport.

class Connections

Bases: object

Store existed connections.

callback
Return callback if existed.

close()

register(*connection*)
Register new connection.

remove(*connection*)
Remove registered connection.

BaseAcceptor.**acceptor**
Return function that should accept new connections.

BaseAcceptor.**active**
Is current acceptor active.

BaseAcceptor.**close**
Close all resources.

BaseAcceptor.**connections_number**
Return number of active connections.

BaseAcceptor.**empty**
Is acceptor empty or not.

BaseAcceptor.**start**
Start acceptor if active.

BaseAcceptor.**stop**
Stop acceptor if active.

class thriftworker.transports.base.**Connections**
Bases: object

Store existed connections.

callback
Return callback if existed.

close()

register(*connection*)
Register new connection.

remove(*connection*)
Remove registered connection.

thriftworker.transports.base.**ignore_eagain**(**args*, ***kwds*)
Ignore all *EAGAIN* errors in context.

Subpackages

framed Package

framed Package

class thriftworker.transports.framed.**FramedAcceptor**(*name*, *descriptor*, *backlog=None*)
Bases: thriftworker.transports.base.BaseAcceptor

class Connection
Bases: object

Which connection should we use?

cb_read_done()

cb_write_done()

close()
Closes connection.

is_closed()
Returns True if connection is closed.

is_ready()
Returns `True` if connection is ready.

on_close()

ready()

utils Package

decorators Module Some decorator implementations.

thriftworker.utils.decorators.**cached_property**
Property descriptor that caches the return value of the get function.

Examples

```
@cached_property
def connection(self):
    return Connection()

@connection.setter  # Prepares stored value
def connection(self, value):
    if value is None:
        raise TypeError('Connection must be a connection')
    return value

@connection.deleter
def connection(self, value):
    # Additional action to do at del(self.attr)
    if value is not None:
        print('Connection %r deleted' % (value, ))
```

env Module

finalize Module Execute cleanup handlers when objects go out of scope.

Taken from `multiprocessing.util.Finalize`.

copyright

3. 2009 - 2012 by Ask Solem.

license BSD, see LICENSE for more details.

class thriftworker.utils.finalize.**Finalize**(*obj*, *callback*, *args=()*, *kwargs=None*, *exitpriority=None*)
Bases: `object`

Object finalization using weakrefs.

cancel()
Cancel finalization of the object.

still_active()

imports Module

thriftworker.utils.imports.**cwd_in_path**(**args*, ***kwds*)

thriftworker.utils.imports.**get_real_module**(*name*)
Get the real Python module, regardless of any monkeypatching

thriftworker.utils.imports.**import_from_cwd**(*module*, *imp=None*, *package=None*)
Import module, but make sure it finds modules located in the current directory.

Modules located in the current directory has precedence over modules located in *sys.path*.

thriftworker.utils.imports.**instantiate**(*name*, **args*, ***kwargs*)
Instantiate class by name.

See `symbol_by_name()`.

thriftworker.utils.imports.**qualname**(*obj*)

thriftworker.utils.imports.**symbol_by_name**(*name*, *aliases={}*, *imp=None*, *package=None*, *sep='.'*, *default=None*, ***kwargs*)
Get symbol by qualified name.

The name should be the full dot-separated path to the class:

```
modulename.ClassName
```

Example:

```
celery.concurrency.processes.TaskPool
                            ^- class name
```

or using ':' to separate module and symbol:

```
celery.concurrency.processes:TaskPool
```

If *aliases* is provided, a dict containing short name/long name mappings, the name is looked up in the aliases first.

Examples:

```
>>> symbol_by_name("celery.concurrency.processes.TaskPool")
<class 'celery.concurrency.processes.TaskPool'>

>>> symbol_by_name("default", {
...     "default": "celery.concurrency.processes.TaskPool"})
<class 'celery.concurrency.processes.TaskPool'>
```

Does not try to look up non-string names. >>> from celery.concurrency.processes import TaskPool >>> symbol_by_name(TaskPool) is TaskPool True

loop Module Execute some function in loop and wait for answer.

class thriftworker.utils.loop.**Container**(*func*, *timeout=None*)
Bases: `object`

Mutable that store result.

dispatch()

exception

from_greenlet(*g*)

func

result

timeout

thriftworker.utils.loop.**DELEGATION_TIMEOUT = 5.0**
Specify default timeout for delegation decorators.

thriftworker.utils.loop.**in_loop**
Schedule execution of given function in main event loop. Wait for function execution.

mixin Module Some useful mixins here.

class thriftworker.utils.mixin.**LoopMixin**
Bases: object

Move loop closer to instance.

app = None

loop
Shortcut to loop.

class thriftworker.utils.mixin.**PropertyMixin**
Bases: object

class thriftworker.utils.mixin.**StartStopMixin**
Bases: object

Implement some stubs method for object that can be started / stopped.

start()

stop()

class thriftworker.utils.mixin.**SubclassMixin**
Bases: object

subclass_with_self(*Class*, *name=None*, *attribute='app'*, *reverse=None*, ***kw*)
Subclass an app-compatible class by setting its app attribute to be this app instance.

App-compatible means that the class has a class attribute that provides the default app it should use, e.g. `class Foo: app = None`.

Parameters

- **Class** – The app-compatible class to subclass.
- **name** – Custom name for the target class.
- **attribute** – Name of the attribute holding the app, default is 'app'.

other Module Some other useful tools.

thriftworker.utils.other.**get_addresses_from_pool**(*name*, *address*, *port_range=None*)
Get addresses from pool.

thriftworker.utils.other.**get_port_from_range**(*name*, *range_start*, *range_end*)
Get random port from given range [range_start, range_end].

thriftworker.utils.other.**rgetattr**(*obj*, *path*)
Get nested attribute from object.

Parameters

- **obj** – object
- **path** – path to attribute

proxy Module Parts of this module is Copyright by Werkzeug Team.

class thriftworker.utils.proxy.**PromiseProxy**(*local*, *args=None*, *kwargs=None*, *name=None*)
Bases: <property object at 0x2687e68>.Proxy

This is a proxy to an object that has not yet been evaluated.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

thriftworker.utils.proxy.**maybe_evaluate**(*obj*)

Subpackages

atomics Package

atomics Package

stats Package

stats Package

counters Module
class thriftworker.utils.stats.counters.**Counters**
Bases: collections.defaultdict

Store counters here.

to_dict()
Convert all counters to dict.

timers Module
class thriftworker.utils.stats.timers.**Timers**
Bases: collections.defaultdict

to_dict()
Convert all timers to dict.

workers Package

base Module
class thriftworker.workers.base.**BaseWorker**(*pool_size=None*)
Bases: thriftworker.utils.mixin.StartStopMixin, abc.NewBase

class Request(*loop*, *connection*, *message_buffer*, *request_id*, *service*)
Bases: object

Describe a request.

connection

dispatch()
Notify connection that request was processed.

dispatch_time

dispatching_timers

end_time

exception

execute(*processor*)
Process our request.

execution_time

loop

message_buffer

method

method_name

receipt_time

request_id

response

service

start_time

successful

BaseWorker.**concurrency**
How many tasks executed in parallel?

BaseWorker.**create_callback**()
Create callback that should be called after request was done.

BaseWorker.**create_consumer**()

BaseWorker.**create_producer**(*service*)
Create producer for connections.

BaseWorker.**create_task**(*processor*)
Create new task for given processor.

class thriftworker.workers.base.**Request**(*loop*, *connection*, *message_buffer*, *request_id*, *service*)
Bases: object

Describe a request.

connection

dispatch()
Notify connection that request was processed.

dispatch_time

dispatching_timers

end_time

exception

execute(*processor*)
Process our request.

execution_time

loop

message_buffer

method

method_name

receipt_time

request_id

response

service

start_time

successful

green Module

sync Module

class thriftworker.workers.sync.**Promise**(*func*, *callback*)

Bases: object

Used to enqueue task execution in thread pool.

callback

cb(**args*)

exception

func

result

class thriftworker.workers.sync.**SyncWorker**(*pool_size=None*)

Bases: thriftworker.workers.base.BaseWorker

Process all request in separate thread.

create_consumer()

threads Module

class thriftworker.workers.threads.**Pool**(*app*, *size=None*)

Bases: object

Orchestrate workers. Start and stop them, provide new tasks.

class **Worker**(*app*, *queue*, *shutdown_timeout=None*)

Bases: threading.Thread

Simple threaded worker.

body()

Consume messages from queue and execute them until empty message sent.

run()

wait()

Pool.**put**(*task*)

Pool.**start**()

Pool.**stop**()

class thriftworker.workers.threads.**ThreadsWorker**(*pool_size=None*)

Bases: thriftworker.workers.base.BaseWorker

Process all request in thread-pool.

class **Message**

Bases: tuple

Message(task, callback)

callback

Alias for field number 1

task

Alias for field number 0

ThreadsWorker.**create_consumer**()

ThreadsWorker.**start**()

ThreadsWorker.**stop**()

class thriftworker.workers.threads.**Worker**(*app*, *queue*, *shutdown_timeout=None*)

Bases: threading.Thread

Simple threaded worker.

body()

Consume messages from queue and execute them until empty message sent.

run()

wait()

1.2 Changelog

1.2.1 0.1.11 - 0.1.14

- Fix bugs;

1.2.2 0.1.1 - 0.1.10

- Add support for prefork model;

1.2.3 0.1.0 (initial release)

- Base prototype;

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index

t