vertx-eventbus-python

Release 3.8.5

Feb 05, 2020

Contents:

1 Links **3**

2 API **5**
2.1 Vertx.eventbus module 5

3 Unit Tests **9**
3.1 test_eventbus module 9

4 Indices and tables **13**

Index **15**

This is a python client for the Vert.x Vert.x-tcp-eventbus-bridge

CHAPTER 1

Links

- Github
- Wiki
- Chat

CHAPTER 2

API

2.1 Vertx.eventbus module

class Vertx.eventbus.**Eventbus**(*host='localhost'*, *port=7000*, *options=None*, *onError=None*, *timeOut=None*, *connect=True*, *debug=False*)

Bases: object

Vert.x TCP eventbus client for python

Variables

- **headers** – any headers to be sent as per the vertx-tcp-eventbus-bridge specification
- **state** (*State.CONNECTING: State*) – the state of the the eventbus
- **host** (*str*) – 'localhost' the host the eventbus is connected to
- **port** (*int*) – 7000 : the port to be used for the socket connection

:ivar pingInterval:5000:the ping interval in millisecs :vartype pingInterval: int

:ivar pongCount:0:the number of pongs received :vartype pongCount: int

Variables

- **timeOut** (*float*) – DEFAULT_TIMEOUT:time in secs to be used as the socket timeout
- **debug** (*bool*) – False: True if debugging should be enabled

:ivar onError:onError:the function to handle errors messages with no address :vartype onError: function

:ivar handlers:{}: the dict of handlers for incoming messages :vartype handlers: dict

:ivar replyHandler:{}: the dict of handlers for reply messages :vartype replyHandlers: dict

__init__(*host='localhost'*, *port=7000*, *options=None*, *onError=None*, *timeOut=None*, *connect=True*, *debug=False*)

constructor

Parameters

- **host** (*str*) – the host to connect to - default: 'localhost'
- **port** (*int*) – the port to use - default: 7000
- **options** (*dict*) – e.g. { vertxbus_ping_interval=5000 }
- **onError** (*function*) – the handler to use for erromessages with no address- default: None will be replaced by default onError
- **timeOut** (*float*) – time in secs to be used as the socket timeout - default: 60 secs - the minimium timeOut is 10 msecs and will be enforced
- **connect** (*bool*) – True if the eventbus should automatically be opened - default: True
- **debug** (*bool*) – True if debugging should be enabled - default: False

Raise

IOError

- the socket could not be opened

Exception

- some other issue e.g. with starting the listening thread

addHeader(*header*, *value*)
add a header with the given header key and value

Parameters

- **header** (*str*) – the key of the header value to add
- **value** (*object*) – the value of the header value to add

close()
close the eventbus connection after staying in the CLOSING state for the given timeInterval

Parameters **timeInterval** (*float*) – the number of seconds to sleep before actually closing the eventbus - default: 30 seconds

isOpen()
Checks if the eventbus state is OPEN.

Returns True if State is OPEN else False

Return type bool

onErrorHandler(*message*)
default onError Handler - only gives debug output

open()
open the eventbus by connecting the eventbus socket and starting a listening thread by default the connection is opened on construction of an Eventbus instance

Raise

IOError

- the socket could not be opened

Exception

- some other issue e.g. with starting the listening thread

ping()
send a ping

Raise

Exception

- eventbus is not open

pongHandler()
default pong Handler - counts the number of pongs Received

publish(*address*, *body=None*, *headers=None*)
publish a message

Parameters

- **address** (*str*) – the target address to send the message to
- **body** (*str*) – the body of the message e.g. a JSON object
- **headers** (*dict*) – headers to be added - default: None

Raise

Exception

- eventbus is not open

registerHandler(*address*, *callback*, *headers=None*)
register a handler

Parameters

- **address** (*str*) – the address to register a handler for
- **callback** (*function*) – a callback for the address
- **headers** (*dict*) – headers to be added - default: None

Raise

Exception

- eventbus is not open
- callback not callable

send(*address*, *body=None*, *callback=None*, *headers=None*)
send a message

Parameters

- **address** (*str*) – the target address to send the message to
- **body** (*str*) – the body of the message e.g. a JSON object- default: None
- **headers** (*dict*) – headers to be added - default: None

Raise

Exception

- eventbus is not open

unregisterHandler(*address*, *callback*, *headers=None*)
unregister a callback for a given address if there is more than one callback for the address it will be remove from the handler list if there is only one callback left an unregister message will be sent over the bus and then the address is fully removed

Parameters

- **address** (*str*) – the address to unregister the handler for
- **callback** (*function*) – the callback to unregister
- **headers** (*dict*) – headers to be added - default: None

Raise

Exception

- eventbus is not open
- address not registered
- callback not registered

wait(*state=<State.OPEN: 1>*, *timeOut=5.0*, *timeStep=0.01*)
wait for the eventbus to reach the given state

Parameters

- **state** (State) – the state to wait for - default: State.OPEN
- **timeOut** (*float*) – the timeOut in secs after which the wait fails with an Exception
- **timeStep** (*float*) – the timeStep in secs in which the state should be regularly checked

Raise

Exception wait timed out

class Vertx.eventbus.**State**
Bases: enum.IntEnum

Eventbus state see https://github.com/vert-x3/vertx-bus-bower/blob/master/vertx-eventbus.js

CLOSED = 3

CLOSING = 2

CONNECTING = 0

OPEN = 1

CHAPTER 3

Unit Tests

3.1 test_eventbus module

Created on 2020-02-01

@author: wf

class tests.test_eventbus.**EchoCommand**(*cmd*, *msgType*, *address*)
Bases: dict

an Echo Command object

__init__(*cmd*, *msgType*, *address*)
construct me

Parameters

- **cmd** (*str*) – a command either “time” or “counter”
- **msgType** (*str*) – a message type either “send” or “publish”
- **address** (*str*) – an address to be used for the echo

asJson()
return me as a json String

Returns the json representation of the EchoCommand

Return type str

class tests.test_eventbus.**Handler**(*debug=False*)
Bases: object

a Handler for messages

__init__(*debug=False*)
construct me

Parameters **debug** (*bool*) – if True show debug messages - default: True

handle(*err*, *message=None*)
handle the given vert.x tcp-event bus message

Parameters

- **err** (*dict*) – potential error message
- **message** (*dict*) – the message dict to handle
- **may contain a body and headers** (*it*) –

class tests.test_eventbus.**TestEventbus**(**args*, ***kwargs*)
Bases: unittest.case.TestCase

test the Eventbus for the vert.x tcp eventbus bridge

__init__(**args*, ***kwargs*)
construct me

classmethod setUpClass()
Hook method for setting up class fixture before running tests in the class.

classmethod tearDownClass()
Hook method for deconstructing the class fixture after running all tests in the class.

testCmd()
test json encoding of a Cmd

testCreateWithInvalidPort()
test creating an event bus for an invalid port

testHandler()
test handler

testMergeHeaders()
test merging headers

testRegisterHandler()
test a successful handler registration

testRegisterWithClosedBus()
try registering an event bus for a closed port

testSocketDirect()
test direct socket communication with echo server

testSocketOfEventBus()
send a message using the private function of the event bus

testTcpEventBusBridgeStarter()
test the TcpEventBusBridgeStarter

testWait()
test waiting for the eventbus to open and close

test_ping()
test sending a ping

test_publish()
test publishing a message to the echo server

test_publishWithHeader()
test publishing a message with headers

test_publishWithMultipleHandlers()
test publishing a message to be handle by multiple handlers

test_registerHandler()
test registering a handler

test_reply()
test sending a message with a reply handler

test_send()
test sending a message

test_sendInvalidAddress()
test trying to send to an invalid address

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

Symbols

__init__() (*Vertx.eventbus.Eventbus method*), 5
__init__() (*tests.test_eventbus.EchoCommand method*), 9
__init__() (*tests.test_eventbus.Handler method*), 9
__init__() (*tests.test_eventbus.TestEventbus method*), 10

A

addHeader() (*Vertx.eventbus.Eventbus method*), 6
asJson() (*tests.test_eventbus.EchoCommand method*), 9

C

close() (*Vertx.eventbus.Eventbus method*), 6
CLOSED (*Vertx.eventbus.State attribute*), 8
CLOSING (*Vertx.eventbus.State attribute*), 8
CONNECTING (*Vertx.eventbus.State attribute*), 8

E

EchoCommand (*class in tests.test_eventbus*), 9
Eventbus (*class in Vertx.eventbus*), 5

H

handle() (*tests.test_eventbus.Handler method*), 9
Handler (*class in tests.test_eventbus*), 9

I

isOpen() (*Vertx.eventbus.Eventbus method*), 6

O

onErrorHandler() (*Vertx.eventbus.Eventbus method*), 6
OPEN (*Vertx.eventbus.State attribute*), 8
open() (*Vertx.eventbus.Eventbus method*), 6

P

ping() (*Vertx.eventbus.Eventbus method*), 6
pongHandler() (*Vertx.eventbus.Eventbus method*), 7
publish() (*Vertx.eventbus.Eventbus method*), 7

R

registerHandler() (*Vertx.eventbus.Eventbus method*), 7

S

send() (*Vertx.eventbus.Eventbus method*), 7
setUpClass() (*tests.test_eventbus.TestEventbus class method*), 10
State (*class in Vertx.eventbus*), 8

T

tearDownClass() (*tests.test_eventbus.TestEventbus class method*), 10
test_ping() (*tests.test_eventbus.TestEventbus method*), 10
test_publish() (*tests.test_eventbus.TestEventbus method*), 10
test_publishWithHeader() (*tests.test_eventbus.TestEventbus method*), 10
test_publishWithMultipleHandlers() (*tests.test_eventbus.TestEventbus method*), 10
test_registerHandler() (*tests.test_eventbus.TestEventbus method*), 11
test_reply() (*tests.test_eventbus.TestEventbus method*), 11
test_send() (*tests.test_eventbus.TestEventbus method*), 11
test_sendInvalidAddress() (*tests.test_eventbus.TestEventbus method*), 11
testCmd() (*tests.test_eventbus.TestEventbus method*), 10
testCreateWithInvalidPort() (*tests.test_eventbus.TestEventbus method*), 10

TestEventbus (*class in tests.test_eventbus*), 10
testHandler() (*tests.test_eventbus.TestEventbus method*), 10
testMergeHeaders() (*tests.test_eventbus.TestEventbus method*), 10
testRegisterHandler() (*tests.test_eventbus.TestEventbus method*), 10
testRegisterWithClosedBus() (*tests.test_eventbus.TestEventbus method*), 10
tests.test_eventbus (*module*), 9
testSocketDirect() (*tests.test_eventbus.TestEventbus method*), 10
testSocketOfEventBus() (*tests.test_eventbus.TestEventbus method*), 10
testTcpEventBusBridgeStarter() (*tests.test_eventbus.TestEventbus method*), 10
testWait() (*tests.test_eventbus.TestEventbus method*), 10

U

unregisterHandler() (*Vertx.eventbus.Eventbus method*), 7

W

wait() (*Vertx.eventbus.Eventbus method*), 8