cli-proton-python Documentation

Release 1.0.0

Petr Matousek

Feb 20, 2018

API

CHAPTER 1

CLI-PROTON-PYTHON

Proton Python clients is a collection of reactive messaging test clients built on python-qpid-proton AMQP1.0 based messaging library. cli-proton-python is a part of Unified Messaging Test Clients that offers equivalent functionality when using different programing languages or APIs.

current related projects:

- cli-java
- cli-rhea
- cli-netlite
- cli-proton-ruby

1.1 Installation

cli-proton-python requires Python v2.6+ to run.

```
$ pip install cli-proton-python
```

1.2 Using

Using the command line clients (please refer to –help to discover the available options)

```
$ cli-proton-python-sender --broker-url "username:password@localhost:5672/queue_test"␣
↪--count 1 --msg-content "text message" --log-msgs dict
$ cli-proton-python-receiver --broker-url "username:password@localhost:5672/queue_test
↪" --count 1 --log-msgs dict
```

Using in script

```
import proton
from cli_proton_python import sender

parser = sender.options.SenderOptions()

opts, _ = parser.parse_args()
opts.broker_url = 'username:password@localhost:5672/examples'
opts.count = 1
opts.msg_content = 'text message'
opts.log_msgs = 'dict'

container = proton.reactor.Container(sender.Send(opts))
container.run()
```

1.3 Documentation

Documentation may be found on readthedocs.io: read the documentation

1.4 License

1.4.1 cli_proton_python.connector module

Proton reactive API python connector client

class cli_proton_python.connector.**Connector**(*opts*)

Bases: *cli_proton_python.coreclient.CoreClient*

Proton reactive API python connector client

close_objects()

closes all the open objects (after given close-sleep time)

get_conn_result()

returns the connection statistics triplet

connection statistictriplets are:

* connections opened, connections errors, connection requests

* connection requests stat is ignored, set to 1 for backwards compatibility

Returns connector statistics triplet

Return type tuple

get_result()

called when the reactor's exit

Returns error message

Return type str

on_connection_opened(*event*)

called when the connection is opened

Parameters **event** (*proton.Event*) – reactor event

on_connection_remote_open(*event*)
called when the remote connection is opening

Parameters **event** (*proton.Event*) – reactor event

on_link_opened(*event*)
called when the link is opened

Parameters **event** (*proton.Event*) – reactor event

on_session_opened(*event*)
called when the session is opened

Parameters **event** (*proton.Event*) – reactor event

on_start(*event*)
called when the event loop starts

Parameters **event** (*proton.Event*) – reactor event

on_transport_error(*event*)
called when the connection can't be opened due to transport error

Parameters **event** (*proton.Event*) – reactor event

cli_proton_python.connector.**main**()
main loop

cli_proton_python.connector.**run_connectors**(*opts*, *results*, *errors*, *stats=None*)
thread worker function

Parameters

- **opts** (*optparse.Values instance*) – connector client options
- **results** (*list*) – list of connection results
- **errors** (*int*) – number of connection errors
- **stats** (*list*) – list containing statistics dictionary (default: None)

1.4.2 cli_proton_python.coreclient module

Proton reactive API python core client module

exception cli_proton_python.coreclient.**ClientException**(*message*)
Bases: exceptions.Exception

custom client exception class (just to know source of exception)

class cli_proton_python.coreclient.**CoreClient**(*opts*, *reactor_opts=None*)
Bases: proton.handlers.MessagingHandler

Proton reactive API python core client

implements various support methods for sender, recevier and connector

static calculate_delay(*in_count*, *in_duration*)
calculates delay between sending/receiving messages used by shecduler when requested by duration option (uses logic from deprecated utils.sleep4next)

Parameters

- **in_count** (*int*) – number of messages to be sent/received

- **in_duration** (*int*) – send/receive process total execution time

Returns computed time to wait before or after message is sent/received

Return type float

clear_messages()
clears list of stored messages

get_messages()
returns list of stored messages

on_transport_error(*event*)
handler called when any error related to transport occurs

See also:

MessagingHandler::on_transport_error

Parameters **event** (*proton.Event*) – reactor event

parse_connection_options()
Prepare options passed to container connect

Returns connection options

Return type dict

parse_link_options()
Prepare link options passed

Returns link options

Return type list

print_message(*message*)
prints or store a message

utils.print_message wrapper

Parameters

- **msg** (*proton.Message*) – message
- **msg_format** (*str*) – pre-defined message format

set_delay_after()
returns delay duration after sending a message (in seconds)

Returns time to wait after message is sent/received

Return type float

set_delay_before()
returns delay duration before sending a message (in seconds)

Returns time to wait before message is sent/received

Return type float

set_up_ssl(*event*)
sets-up SSLDomain

Parameters **event** (*proton.Event*) – reactor event

set_up_ssl_client_auth(*event*)
sets-up SSLDomain for the client authentication

Parameters **event** (*proton.Event*) – reactor event

set_up_ssl_server_auth(*event*)
set-up SSLDomain for the server verification

VERIFY_PEER: Require peer to provide a valid identifying certificate
VERIFY_PEER_NAME: Require valid certificate and matching name

Parameters **event** (*proton.Event*) – reactor event

tear_down(*event*, *settled=False*)
tears down and closes the connection

Parameters

- **event** (*proton.Event*) – reactor event
- **settled** (*bool*) – indicates whether all messages has been explicitly settled

class cli_proton_python.coreclient.**CustomBackoff**(*interval=None*, *limit=None*, *timeout=None*)
Bases: proton.reactor.Backoff

a reconnect strategy involving an increasing delay between retries, up to a maximum or 60 seconds. This is a modified version supporting limit to the number of reconnect attempts before giving up.

next()
implements the reconnect attempt action

Returns next reconnect attempt delay time

Return type float

reset()
resets the reconnect attempts counters

class cli_proton_python.coreclient.**DelayedNoop**
Bases: object

callback object that does nothing.

on_timer_task(*_*)
empty event handler method

class cli_proton_python.coreclient.**ErrorsHandler**(*conn_reconnect*)
Bases: object

class to be used as universal errors handler for clients

on_unhandled(*name*, *event*)
Universal handler which sees all events when added as global handler to reactor.

See also:

node_data/clients/python/receiver.py and node_data/clients/python/sender.py

Parameters

- **name** (*str*) – event name
- **event** (*proton.Event*) – event object

1.4.3 cli_proton_python.formatter module

Clients output formatter module

class cli_proton_python.formatter.**Formatter**(*message*)
Bases: object

Output formatter class for clients

static format_dict(*in_data*)
formats dictionary

Parameters **in_data** (*dict*) – input data

Returns input data string formated as dict

Return type str

static format_float(*in_data*)
formats float value

Parameters **in_data** (*float*) – input data

Returns input data string formated as float

Return type str

static format_int(*in_data*)
formats integer value

Parameters **in_data** (*int, long*) – input data

Returns input data string formated as int

Return type str

static format_list(*in_data*)
formats list

Parameters **in_data** (*list*) – input data

Returns input data string formated as list

Return type str

static format_object(*in_data*)
formats general object

Parameters **in_data** (*None, bool, int, long, float, dict, list, str, unicode, bytes*) – input data

Returns input data converted to string

Return type str

static format_string(*in_data*)
formats string

Parameters **in_data** (*str, unicode, bytes*) – input data

Returns input data string formated as string

Return type str

print_error()
print error information

Returns prefix message with string indicating error

Return type str

print_message()
prints message in default upstream format

Returns message to be printed in upstream format

Return type str

print_message_as_dict()
prints message in python dictionary form

Returns message to be printed in dictionary format

Return type str

print_message_as_interop()
Print message in AMQP interoperable format

Returns message to be printed in interoperable format

Return type str

print_message_as_json()
Print message in JSON form

Returns message to be printed in JSON form

Return type str

print_stats()
print statistics information

Returns prefix message with string indicating statistics

Return type str

static quote_string_escape(*in_data*)
escapes quotes in given string

Parameters **in_data** (*str, unicode*) – input string

Returns input string with quotes escaped

Return type str, unicode

1.4.4 cli_proton_python.options module

Proton reactive API python client options module

class cli_proton_python.options.**ConnectorOptions**
Bases: *cli_proton_python.options.CoreOptions*

Proton reactive API python connector specific client options

add_connector_options()
add the connector options

class cli_proton_python.options.**CoreOptions**
Bases: optparse.OptionParser, object

Proton reactive API python core client options

add_connection_options()
add the connection options

add_control_options()
add the control options

add_logging_options()
add the logging options

class cli_proton_python.options.**ReceiverOptions**
Bases: *cli_proton_python.options.SRCoreOptions*

Proton reactive API python receiver specific client options

add_control_options()
add the control options

add_link_options()
add the control options

add_reactor_options()
add receiver's options

add_receiver_options()
add receiver's options

class cli_proton_python.options.**SRCoreOptions**
Bases: *cli_proton_python.options.CoreOptions*

Proton reactive API python sender/receiver client options

add_control_options()
add the control options

add_link_options()
add the link options

add_logging_options()
add the logging options

add_transaction_options()
add the transaction options

class cli_proton_python.options.**SenderOptions**
Bases: *cli_proton_python.options.SRCoreOptions*

Proton reactive API python sender specific client options

add_message_options()
add the message options

add_reactor_options()
add receiver's options

cli_proton_python.options.**convert_to_unicode**(*value*)
Python 2.x: converts value to unicode

Parameters **value** (*str*) – value to be converted to unicode

Returns unicode string

Return type str (unicode)

cli_proton_python.options.**str_to_unicode**(*option*, *_*, *value*, *parser*)
Python 2.x: stores cmdline string, converts to unicode for Python 2.x

Parameters

- **option** – option object

- **value** (*related Option object from cli_proton_python.options*) – option value
- **value** – option parser

cli_proton_python.options.**to_unicode**(*option*, *_*, *value*, *parser*)
stores values of multi-value cmdline string, converts to unicode for Python 2.x

Parameters

- **option** – option object
- **value** (*related Option object from cli_proton_python.options*) – option value
- **value** – option parser

1.4.5 cli_proton_python.receiver module

Proton reactive API python reciever client

class cli_proton_python.receiver.**Recv**(*opts*, *prefetch=None*)
Bases: *cli_proton_python.coreclient.CoreClient*

Proton reactive API python receiver client

implements various handler methods for reactor events triggered by proton.reactor

check_empty(*event*)
checks whether there are messages to dispatch

Parameters **event** (*proton.Event*) – reactor event

do_message_action(*delivery*)
performs requested action on received message

Parameters **delivery** (*proton.Delivery*) – delivery disposition

on_connection_opened(*event*)
called when connection is opened

Parameters **event** (*proton.Event*) – reactor event

on_delivery(*event*)
called when a message is delivered

Parameters **event** (*proton.Event*) – reactor event

on_link_flow(*event*)
called on link flow

Parameters **event** (*proton.Event*) – reactor event

on_link_opened(*event*)
called when link is opened

Parameters **event** (*proton.Event*) – reactor event

on_message(*event*)
called when a message is received

Parameters **event** (*proton.Event*) – reactor event

on_settled(*event*)
called when the remote peer has settled the outgoing message this is the point at which it should never be retransmitted

Parameters **event** (*proton.Event*) – reactor event

on_start(*event*)
called when the event loop starts, creates a receiver for given url

Parameters **event** (*proton.Event*) – reactor event

parse_link_options()
Prepare link options

Returns list of link options

Return type list

process_reply_to(*event*)
sends received message to reply to address

Parameters **event** (*proton.Event*) – reactor event

tear_down(*event*, *settled=False*)
tears down and closes the connection

Parameters

- **event** (*proton.Event*) – reactor event
- **settled** (*bool*) – indicates whether all messages has been explicitly settled

class cli_proton_python.receiver.**Timeout**(*parent*, *event*)
Bases: object

Scheduler object for timeout control

on_timer_task(*_*)
on_timer_task action handler

if it's time to close, so no new messages arrived in time, close connection

class cli_proton_python.receiver.**TxRecv**(*opts*)
Bases: *cli_proton_python.receiver.Recv*, proton.handlers.TransactionHandler

Proton reactive API python transactional receiver client

implements various handler methods for reactor events triggered by proton.reactor

is_empty(*event*)
check if queue is empty

Parameters **event** (*proton.Event*) – reactor event

Returns True if the source is empty, False otherwise

Return type bool

on_delivery(*event*)
called when a message is delivered

Parameters **event** (*proton.Event*) – reactor event

on_disconnected(*_*)
called when the transaction is disconnected

on_message(*event*)
called when a message is received

Parameters **event** (*proton.Event*) – reactor event

on_start(*event*)
called when the event loop starts, creates a transactional receiver for given url

Parameters **event** (*proton.Event*) – reactor event

on_transaction_aborted(*event*)
called when the transaction is aborted

Parameters **event** (*proton.Event*) – reactor event

on_transaction_committed(*event*)
called when the transaction is committed

Parameters **event** (*proton.Event*) – reactor event

on_transaction_declared(*event*)
called when the transaction is declared

Parameters **event** (*proton.Event*) – reactor event

transaction_finish(*event*)
finish transaction, do tranaction action, process reporting and control options

Parameters **event** (*proton.Event*) – reactor event

transaction_process(*event*)
transactionally receive a message, process reporting and control options

Parameters **event** (*proton.Event*) – reactor event

cli_proton_python.receiver.**main**()
main loop

1.4.6 cli_proton_python.sender module

Proton reactive API python sender client

class cli_proton_python.sender.**Send**(*opts*)
Bases: *cli_proton_python.coreclient.CoreClient*

Proton reactive API python sender client

implements various handler methods for reactor events triggered by proton.reactor

on_accepted(*event*)
called when the remote peer accepts an outgoing message

Parameters **event** (*proton.Event*) – reactor event

on_disconnected(*event*)
called when the socket is disconnected

Parameters **event** (*proton.Event*) – reactor event

on_rejected(*event*)
called when the remote peer rejects an outgoing message

Parameters **event** (*proton.Event*) – reactor event

on_sendable(*event*)
called when sending can proceed

Parameters **event** (*proton.Event*) – reactor event

on_settled(*event*)
called when the remote peer has settled the outgoing message, this is the point at which it should never be retransmitted

Parameters **event** (*proton.Event*) – reactor event

on_start(*event*)
called when the event loop starts, creates a sender for given url

Parameters **event** (*proton.Event*) – reactor event

on_timer_task(*_*)
next send action scheduler

the Send object itself is shecduled to perform next send action, that is why it contains timer_task method method

incoming event object does not contain many fields, that is why self.event is used

prepare_content()
prepares the content depending on type

Note:

- if self.opts.msg_list_items are set amqp/map content is constructed,
- elif self.opts.msg_map_items are set amqp/list content is constructed,
- else the content is considered as text/plain

Returns string message content

Return type str (unicode) or list or dict

static prepare_content_from_file(*filename*)
reads and returns file contents

Parameters **filename** (*str*) – path to file to be opened and read

Returns contents of filename as unicode string

Return type str (unicode) or None

prepare_list_content()
prepares list content

Returns list constructed from options list items

Return type list

prepare_map_content()
prepares map content

Returns flat map constructed from options map items

Return type dict

prepare_message()
compose and return the message

Returns message to be sent

Return type proton.Message

prepare_string_content(*content*)
prepares string content

re-types content accoding content-type given, enables message sequence numbering if formatting string (%[0-9]*d) is found

Parameters **content** (*str (unicode)*) – message content string

Returns string message content

Return type str (unicode)

send_message()
sends a message

class cli_proton_python.sender.**TxSend**(*opts*)
Bases: *cli_proton_python.sender.Send*, proton.handlers.TransactionHandler

Proton reactive API python transactional sender client

implements various handler methods for reactor events triggered by proton.reactor

on_accepted(*event*)
suppressed in transactional, no actions are performed

Parameters **event** (*proton.Event*) – reactor event

on_disconnected(*event*)
suppressed in transactional, no actions are performed

Parameters **event** (*proton.Event*) – reactor event

on_sendable(*event*)
suppressed in transactional, no actions are performed

Parameters **event** (*proton.Event*) – reactor event

on_settled(*event*)
suppressed in transactional, no actions are performed

Parameters **event** (*proton.Event*) – reactor event

on_start(*event*)
called when the event loop starts, creates a transactional sender for given url

Parameters **event** (*proton.Event*) – reactor event

on_timer_task(*_*)
suppressed in transactional, no actions are performed

Parameters **event** (*proton.Event*) – reactor event

on_transaction_aborted(*event*)
called when the transaction is aborted

Parameters **event** (*proton.Event*) – reactor event

on_transaction_committed(*event*)
called when the transaction is committed

Parameters **event** (*proton.Event*) – reactor event

on_transaction_declared(*event*)
called when the transaction is declared

Parameters **event** (*proton.Event*) – reactor event

transaction_finish(*event*)
finish transaction, do tranaction action, process reporting and control options

Parameters **event** (*proton.Event*) – reactor event

transaction_process(*event*)
transactionally send a message, process reporting and control options

Parameters **event** (*proton.Event*) – reactor event

cli_proton_python.sender.**main**()
main loop

1.4.7 cli_proton_python.utils module

Various client's common utils functions

cli_proton_python.utils.**dump_error**(*err_message*)
dump error message in parsable format

Parameters **err_message** (*str*) – error massage to be logged

cli_proton_python.utils.**dump_event**(*event*)
dumps proton event object

Parameters **event** (*proton.Event*) – reactor event to be dumped

cli_proton_python.utils.**hard_retype**(*value*)
tries to converts value to relevant type by re-typing

Parameters **value** (*str (unicode)*) – value to be converted

Returns re-typed value

Return type int, float, bool, str

cli_proton_python.utils.**nameval**(*in_string*)
converts given string to key, value and separator triplets

Parameters **in_string** (*str (unicode)*) – key/value pair

Returns key, value and separator triplet

Return type tuple

cli_proton_python.utils.**prepare_flat_map**(*entries*, *e_type=None*)
prepares map content from multiple key, value pairs

Note: only flat map is currently supported

Parameters

- **entries** (*list*) – list of key, separator, value triplets
- **e_type** (*str*) – map entries desired content type (default: None)

Returns flat map containing given entries of given type

Return type dict

cli_proton_python.utils.**print_message**(*msg*, *msg_format*)
prints a message in coresponding format

Parameters

- **msg** (*proton.Message*) – message
- **msg_format** (*str*) – pre-defined message format

cli_proton_python.utils.**retype_content**(*content*, *content_type*)
converts the content depending on type

Parameters

- **content** (*str (unicode)*) – message content
- **content_type** (*str*) – message content type

Returns re-typed content according to given content_type

Return type int, float, long, bool, str

cli_proton_python.utils.**set_up_client_logging**(*level*)
sets up the client library logging

Parameters **level** (*int, string*) – log level number or proton logging type

cli_proton_python.utils.**sleep4next**(*in_ts*, *in_count*, *in_duration*, *in_indx*)
custom sleep for checkpoints

Deprecated since version 1.0.0: use scheduler instead

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

Changelog

v1.0.0 – November 20, 2017

v1.0.1 – November 24, 2017

- json message output format support added
- minor documentation and packaging changes
- API documentation generation

v1.0.2 – February 02, 2018 * address / amqp-to message field support * authentication mechanisms option support * existing bug references added to tests

v1.0.3 – February 05, 2018 * message numbering fix * use anonymous authentication mechanism in p2p tests

v1.0.4 – TBD, not yet live * message group sequence option support

Python Module Index

C

Index