
message-queue Documentation

Release 0.1.1

Ingress

March 24, 2016

| | | |
|----------|------------------------------------|-----------|
| 1 | Current supported adapters: | 3 |
| 2 | Installing message-queue | 5 |
| 3 | Documentation | 7 |
| 4 | Indices and tables | 11 |
| | Python Module Index | 13 |

Message Queue python library to publish and subscribe to queues with diferent types of adapters.

Current supported adapters:

- RabbitMQ - AMQP 0.9.1

Installing message-queue

message-queue is available to install via pip:

```
pip install message-queue
```


3.1 Module Documentation

Message Queue python library to publish and subscribe to queues with diferent types of adapters.

3.1.1 Documentation

Adapters

Current available adapters.

BaseAdapter

Message Queue Base Adapter Interface.

```
class message_queue.adapters.base_adapter.BaseAdapter
```

```
    Bases: object
```

```
    close ()
```

```
        Close connection.
```

```
    configure_queue ()
```

```
        Define the queue configuration.
```

```
    connect ()
```

```
        Connect to queue.
```

```
    consume ()
```

```
        Consume message from the queue.
```

```
    consume_callback ()
```

```
        Callback method to execute in the consume.
```

```
    format_message ()
```

```
        Format message to send to the queue.
```

```
    send ()
```

```
        Publish a message to the queue.
```

AMQPAdapter

Example:

```
from message_queue import AMQPAdapter

adapter = AMQPAdapter(host='0.0.0.0')
```

AMQP 0.9.1 Adapter to connect to RabbitMQ using pika library.

Publish and subscribe to queues and exchanges in RabbitMQ

```
class message_queue.adapters.amqp_adapter.AMQPAdapter (host='localhost', port=5672,
                                                         user='guest', password='guest',
                                                         vhost='/')
```

Bases: *message_queue.adapters.base_adapter.BaseAdapter*

```
__init__ (host='localhost', port=5672, user='guest', password='guest', vhost='/')
```

Create the connection credentials and parameters then connect.

Parameters

- **host** (*string*) – Server host
- **port** (*int*) – Server port
- **user** (*string*) – Server server user
- **password** (*string*) – Server server password
- **vhost** (*string*) – Server virtual host

close ()

Close connection and channel.

configure_queue (**kwargs)

Configure the queue.

Parameters

- **prefetch_count** (*int*) – Specifies a prefetch window in terms of whole messages
- **queue** (*string*) – Queue name to connect
- **passive** (*bool*) – Only check to see if the queue exists
- **durable** (*bool*) – Survive reboots of the broker
- **exclusive** (*bool*) – Only allow access by the current connection
- **auto_delete** (*bool*) – Delete after consumer cancels or disconnects
- **arguments** (*bool*) – Custom key/value arguments for the queue

connect ()

Connect to AMQP server usgin BlockingConnection.

consume (*worker*)

Consume message from the queue.

Parameters **worker** (*function*) – Method that consume the message

consume_callback (*worker*)

Decorate worker to exectue on consume callback.

Parameters **worker** (*function*) – Worker to exectue in the consume callback

format_message (*message*)

Format message to AMQP format.

Parameters **message** (*dict*) – Message to format

send (*message*)

Publish a message in the queue.

Parameters **message** (*Message*) – Message to publish in the channel

Indices and tables

- [genindex](#)

Message

Example:

```
from message_queue import Message
message = Message({ 'id': 1 })
```

Create messages to publish in the queue.

class `message_queue.message.Message` (*content*, ****kwargs**)

__init__ (*content*, ****kwargs**)

Create a new message.

Parameters

- **content** (*dict*) – Content of the message
- **kwargs** (*dict*) – Extra parameters for the message

get_content ()

Get the message content.

Return type `dict`

to_json (*content*)

Convert content to json.

Parameters **content** (*dict*) – Content to encode in json format

Returns type `string`

Publisher

Example:

```
from message_queue import AMQPAdapter
from message_queue import Publisher
from message_queue import Message

adapter = AMQPAdapter(host='0.0.0.0')
publisher = Publisher(adapter)
message = Message({ 'id': 1 })
```

```
publisher.publish(message)
```

Message Queue Publisher.

class `message_queue.publisher.Publisher` (*adapter*)

__init__ (*adapter*)

Create a new publisher with an Adapter instance.

Parameters **adapter** (`BaseAdapter`) – Connection Adapter

publish (*message*)

Publish a message message.

Parameters **message** (`Message`) – Message to publish in the channel

Subscriber

Example:

```
from message_queue import AMQPAdapter
from message_queue import Subscriber

def worker(channel, method, properties, body):
    print body

adapter = AMQPAdapter(host='0.0.0.0')
subscriber = Subscriber(adapter)

subscriber.consume(worker)
```

Subscribe to a specific queue and consume the messages.

class `message_queue.subscriber.Subscriber` (*adapter*)

__init__ (*adapter*)

Create a new subscriber with an Adapter instance.

Parameters **adapter** (`BaseAdapter`) – Connection Adapter

consume (*worker*)

Consume a queued message.

Parameters **worker** (*function*) – Worker to execute when consuming the message

3.1.2 Indices and tables

- [genindex](#)

Indices and tables

- `genindex`

m

`message_queue.adapters.amqp_adapter`, 8
`message_queue.adapters.base_adapter`, 7
`message_queue.message`, 9
`message_queue.publisher`, 10
`message_queue.subscriber`, 10

Symbols

- [__init__\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[__init__\(\)](#) (message_queue.message.Message method), 9
[__init__\(\)](#) (message_queue.publisher.Publisher method), 10
[__init__\(\)](#) (message_queue.subscriber.Subscriber method), 10
- A**
 AMQPAdapter (class in message_queue.adapters.amqp_adapter), 8
- B**
 BaseAdapter (class in message_queue.adapters.base_adapter), 7
- C**
[close\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[close\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
[configure_queue\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[configure_queue\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
[connect\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[connect\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
[consume\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[consume\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
[consume\(\)](#) (message_queue.subscriber.Subscriber method), 10
[consume_callback\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[consume_callback\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
- F**
[format_message\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 8
[format_message\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
- G**
[get_content\(\)](#) (message_queue.message.Message method), 9
- M**
 Message (class in message_queue.message), 9
 message_queue.adapters.amqp_adapter (module), 8
 message_queue.adapters.base_adapter (module), 7
 message_queue.message (module), 9
 message_queue.publisher (module), 10
 message_queue.subscriber (module), 10
- P**
[publish\(\)](#) (message_queue.publisher.Publisher method), 10
 Publisher (class in message_queue.publisher), 10
- S**
[send\(\)](#) (message_queue.adapters.amqp_adapter.AMQPAdapter method), 9
[send\(\)](#) (message_queue.adapters.base_adapter.BaseAdapter method), 7
 Subscriber (class in message_queue.subscriber), 10
- T**
[to_json\(\)](#) (message_queue.message.Message method), 9