Python Binary Memcached (bmemached) Documentation

Release 0.17

Jayson Reis

April 15, 2013

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION TO BMEMCACHED

A pure python module to access memcached via it's binary with SASL auth support.

The main purpose of this module it to be able to communicate with memcached using binary protocol and support authentication, so it can work with Heroku for example.

1.1 Installing

Use pip or easy_install.

```
pip install python-binary-memcached
```

1.2 Using

```
import bmemcached
client = bmemcached.Client(('127.0.0.1:11211', ), 'user',
                           'password')
client.set('key', 'value')
print client.get('key')
```

1.3 Running the tests

First run memcached with:

```
memcached -S -vvv
memcached -p5000 -S -vvv
memcached -S -s/tmp/memcached.sock -vvv
```

This is to cover all tests with socket, standard port and non standard port.

Then, run the tests.

```
cd src_dir/
py.test
```

1.4 Using with Django

If you want to use it with Django, go to django-bmemcached to get a Django backend.

CHAPTER

TWO

BMEMCACHED PACKAGE

2.1 bmemcached Package

class bmemcached.__init__.**Client**(*servers=['127.0.0.1:11211'], username=None, password=None*)

Bases: object

This is intended to be a client class which implement standard cache interface that common libs do.

add(*key*, *value*, *time=100*)

Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

decr(*key*, *value*)

Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented

Returns Actual value of the key on server

Return type int

delete(*key*)

Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

disconnect_all()

Disconnect all servers.

Returns Nothing

Return type None

flush_all(*time=0*)
Send a command to server flushldelete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time=100*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

set(*key*, *value*, *time=100*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi (*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_servers (*servers*)
Iter to a list of servers and instantiate Server class.

Parameters **servers** (*list*) – A list of servers

Returns Returns nothing

Return type None

stats (*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

2.2 client Module

class bmemcached.client.**Client** (*servers=['127.0.0.1:11211'], username=None, password=None*)
Bases: object

This is intended to be a client class which implement standard cache interface that common libs do.

add (*key*, *value*, *time=100*)
Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

decr (*key*, *value*)
Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented

Returns Actual value of the key on server

Return type int

delete(*key*)
Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

disconnect_all()
Disconnect all servers.

Returns Nothing

Return type None

flush_all(*time=0*)
Send a command to server flush|delete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time=100*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.

- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

set (*key*, *value*, *time=100*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi (*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_servers (*servers*)
Iter to a list of servers and instantiate Server class.

Parameters **servers** (*list*) – A list of servers

Returns Returns nothing

Return type None

stats (*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

2.3 exceptions Module

exception bmemcached.exceptions.**AuthenticationNotSupported**
Bases: bmemcached.exceptions.MemcachedException

exception bmemcached.exceptions.**InvalidCredentials**
Bases: bmemcached.exceptions.MemcachedException

exception bmemcached.exceptions.**MemcachedException**
Bases: exceptions.Exception

2.4 server Module

class bmemcached.server.**Server**(*server*, *username=None*, *password=None*)

Bases: object

This class is used by Client class to communicate with server.

COMMANDS = {'auth_negotiation': {'command': 32}, 'getkq': {'command': 13, 'struct': '%ds'}, 'stat': {'command': 16},

COMPRESSION_THRESHOLD = 128

FLAGS = {'integer': 2, 'pickle': 1, 'compressed': 8, 'long': 4}

HEADER_SIZE = 24

HEADER_STRUCT = '!BBHBBHLLQ'

MAGIC = {'request': 128, 'response': 129}

STATUS = {'key_exists': 2, 'auth_error': 8, 'unknown_command': 129, 'success': 0, 'key_not_found': 1}

add(*key*, *value*, *time*)

Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

authenticate(*username*, *password*)

Authenticate user on server.

Parameters

- **username** (*basestring*) – Username used to be authenticated.
- **password** (*basestring*) – Password used to be authenticated.

Returns True if successful.

Raises InvalidCredentials, AuthenticationNotSupported, MemcachedException

Return type bool

decr(*key*, *value*, *default=0*, *time=100*)

Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented
- **default** (*int*) – Default value if key does not exist.
- **time** (*int*) – Time in seconds to expire key.

Returns Actual value of the key on server

Return type int

delete(*key*)
Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

deserialize(*value*, *flags*)
Deserialized values based on flags or just return it if it is not serialized.

Parameters

- **value** (*basestring, int*) – Serialized or not value.
- **flags** (*int*) – Value flags

Returns Deserialized value

Return type basestringlint

disconnect()
Disconnects from server.

Returns Nothing

Return type None

flush_all(*time*)
Send a command to server flushldelete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*, *default=0*, *time=1000000*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented
- **default** (*int*) – Default value if key does not exist.
- **time** (*int*) – Time in seconds to expire key.

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

serialize(*value*)
Serializes a value based on it's type.

Parameters **value** (*basestring, int, long, object*) – Something to be serialized

Returns Serialized type

Return type str

set(*key*, *value*, *time*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi(*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

split_host_port(*server*)
Return (host, port) from server.

Port defaults to 11211.

```
>>> split_host_port('127.0.0.1:11211')
('127.0.0.1', 11211)
>>> split_host_port('127.0.0.1')
('127.0.0.1', 11211)
```

stats(*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

CHAPTER

THREE

BMEMCACHED

3.1 bmemcached Package

3.1.1 bmemcached Package

class bmemcached.__init__.**Client**(*servers=['127.0.0.1:11211'], username=None, password=None*)

Bases: object

This is intended to be a client class which implement standard cache interface that common libs do.

add(*key*, *value*, *time=100*)

Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

decr(*key*, *value*)

Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented

Returns Actual value of the key on server

Return type int

delete(*key*)

Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

disconnect_all()
Disconnect all servers.

Returns Nothing

Return type None

flush_all(*time=0*)
Send a command to server flush|delete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time=100*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

set(*key*, *value*, *time=100*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name

- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi (*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_servers (*servers*)
Iter to a list of servers and instantiate Server class.

Parameters **servers** (*list*) – A list of servers

Returns Returns nothing

Return type None

stats (*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

3.1.2 client Module

class bmemcached.client.**Client** (*servers=['127.0.0.1:11211'], username=None, password=None*)
Bases: object

This is intended to be a client class which implement standard cache interface that common libs do.

add (*key*, *value*, *time=100*)
Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

decr (*key*, *value*)
Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented

Returns Actual value of the key on server

Return type int

delete(*key*)
Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

disconnect_all()
Disconnect all servers.

Returns Nothing

Return type None

flush_all(*time=0*)
Send a command to server flush|delete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time=100*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

set (*key*, *value*, *time=100*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi (*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_servers (*servers*)
Iter to a list of servers and instantiate Server class.

Parameters **servers** (*list*) – A list of servers

Returns Returns nothing

Return type None

stats (*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

3.1.3 exceptions Module

exception bmemcached.exceptions.**AuthenticationNotSupported**
Bases: bmemcached.exceptions.MemcachedException

exception bmemcached.exceptions.**InvalidCredentials**
Bases: bmemcached.exceptions.MemcachedException

exception bmemcached.exceptions.**MemcachedException**
Bases: exceptions.Exception

3.1.4 server Module

class bmemcached.server.**Server**(*server*, *username=None*, *password=None*)

Bases: object

This class is used by Client class to communicate with server.

COMMANDS = {'auth_negotiation': {'command': 32}, 'getkq': {'command': 13, 'struct': '%ds'}, 'stat': {'command': 16},

COMPRESSION_THRESHOLD = 128

FLAGS = {'integer': 2, 'pickle': 1, 'compressed': 8, 'long': 4}

HEADER_SIZE = 24

HEADER_STRUCT = '!BBHBBHLLQ'

MAGIC = {'request': 128, 'response': 129}

STATUS = {'key_exists': 2, 'auth_error': 8, 'unknown_command': 129, 'success': 0, 'key_not_found': 1}

add(*key*, *value*, *time*)

Add a key/value to server ony if it does not exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is added False if key already exists

Return type bool

authenticate(*username*, *password*)

Authenticate user on server.

Parameters

- **username** (*basestring*) – Username used to be authenticated.
- **password** (*basestring*) – Password used to be authenticated.

Returns True if successful.

Raises InvalidCredentials, AuthenticationNotSupported, MemcachedException

Return type bool

decr(*key*, *value*, *default=0*, *time=100*)

Decrement a key, if it exists, returns it's actual value, if it don't, return 0. Minimum value of decrement return is 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be decremented
- **default** (*int*) – Default value if key does not exist.
- **time** (*int*) – Time in seconds to expire key.

Returns Actual value of the key on server

Return type int

delete(*key*)
Delete a key/value from server. If key does not exist, it returns True.

Parameters **key** (*basestring*) – Key's name to be deleted

Returns True in case o success and False in case of failure.

Return type bool

deserialize(*value*, *flags*)
Deserialized values based on flags or just return it if it is not serialized.

Parameters

- **value** (*basestring, int*) – Serialized or not value.
- **flags** (*int*) – Value flags

Returns Deserialized value

Return type basestringlint

disconnect()
Disconnects from server.

Returns Nothing

Return type None

flush_all(*time*)
Send a command to server flushldelete all keys.

Parameters **time** (*int*) – Time to wait until flush in seconds.

Returns True in case of success, False in case of failure

Return type bool

get(*key*)
Get a key from server.

Parameters **key** (*basestring*) – Key's name

Returns Returns a key data from server.

Return type object

get_multi(*keys*)
Get multiple keys from server.

Parameters **keys** (*list*) – A list of keys to from server.

Returns A dict with all requested keys.

Return type dict

incr(*key*, *value*, *default=0*, *time=1000000*)
Increment a key, if it exists, returns it's actual value, if it don't, return 0.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*int*) – Number to be incremented
- **default** (*int*) – Default value if key does not exist.
- **time** (*int*) – Time in seconds to expire key.

Returns Actual value of the key on server

Return type int

replace(*key*, *value*, *time*)
Replace a key/value to server ony if it does exist.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True if key is replace False if key does not exists

Return type bool

serialize(*value*)
Serializes a value based on it's type.

Parameters **value** (*basestring, int, long, object*) – Something to be serialized

Returns Serialized type

Return type str

set(*key*, *value*, *time*)
Set a value for a key on server.

Parameters

- **key** (*basestring*) – Key's name
- **value** (*object*) – A value to be stored on server.
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

set_multi(*mappings*, *time=100*)
Set multiple keys with it's values on server.

Parameters

- **mappings** (*dict*) – A dict with keys/values
- **time** (*int*) – Time in seconds that your key will expire.

Returns True in case of success and False in case of failure

Return type bool

split_host_port(*server*)
Return (host, port) from server.

Port defaults to 11211.

```
>>> split_host_port('127.0.0.1:11211')
('127.0.0.1', 11211)
>>> split_host_port('127.0.0.1')
('127.0.0.1', 11211)
```

stats(*key=None*)
Return server stats.

Parameters **key** (*basestring*) – Optional if you want status from a key.

Returns A dict with server stats

Return type dict

CHAPTER

FOUR

INDICES AND TABLES

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

PYTHON MODULE INDEX

b