
trollius_redis Documentation

Release 0.1.4

Ben Jolitz

April 20, 2015

1	Features	3
2	Installation	5
3	Author and License	7
4	Indices and tables	9
4.1	Examples	9
4.2	Reference	13

Asynchronous Redis client for Python trollius.

Port of .._asyncio-redis: <https://github.com/jonathanslenders/asyncio-redis>

This Redis library is a completely asynchronous, non-blocking client for a Redis server.

It depends on the asyncio way of doing things (PEP 3156) but uses trollius so you can use Python 2.

If you're new to asyncio, it can be helpful to check out [the asyncio documentation](#) first.

Features

- Works for the trollius (PEP3156) event loop
- No dependencies except trollius
- Connection pooling and pipelining
- Automatic conversion from native Python types (unicode or bytes) to Redis types (bytes).
- Blocking calls and transactions supported
- Pubsub support
- Streaming of multi bulk replies
- Completely tested

Installation

```
pip install trollius-redis
```

Start by taking a look at *some examples*.

Author and License

The `trollius_redis` package is a port done by Ben Jolitz of the `asyncio_redis` package originally written by Jonathan Slenders. It's BSD licensed and freely available. Feel free to improve this package and [send a pull request](#).

Indices and tables

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

4.1 Examples

4.1.1 The Connection class

A `Connection` instance will take care of the connection and will automatically reconnect, using a new transport when the connection drops. This connection class also acts as a proxy to a `RedisProtocol` instance; any Redis command of the protocol can be called directly at the connection.

```
import trollius
from trollius import From
import trollius_redis

@trollius.coroutine
def example():
    # Create Redis connection
    connection = yield From(trollius_redis.Connection.create(host=u'localhost', port=6379))

    # Set a key
    yield From(connection.set(u'my_key', u'my_value'))

    # When finished, close the connection.
    connection.close()

if __name__ == '__main__':
    loop = trollius.get_event_loop()
    loop.run_until_complete(example())
```

See *the reference* to learn more about the other Redis commands.

4.1.2 Connection pooling

Requests will automatically be distributed among all connections in a `Pool`. If a connection is blocking because of—for instance—a blocking `rpop`, another connection will be used for new commands.

Note: This is the recommended way to connect to the Redis server.

```
import trollius
from trollius import From
import trollius_redis

@trollius.coroutine
def example():
    # Create Redis connection
    connection = yield From(trollius_redis.Pool.create(
        host='localhost', port=6379, poolsize=10))

    # Set a key
    yield From(connection.set(u'my_key', u'my_value'))

    # When finished, close the connection pool.
    connection.close()
```

4.1.3 Transactions

A transaction can be started by calling `multi`. This returns a `Transaction` instance which is in fact just a proxy to the `RedisProtocol`, except that every Redis method of the protocol now became a coroutine that returns a future. The results of these futures can be retrieved after the transaction is committed with `exec`.

```
import trollius
from trollius import From
import trollius_redis

@trollius.coroutine
def example(loop):
    # Create Redis connection
    connection = yield From(trollius_redis.Pool.create(
        host='localhost', port=6379, poolsize=10))

    # Create transaction
    transaction = yield From(connection.multi())

    # Run commands in transaction (they return future objects)
    f1 = yield From(transaction.set(u'key', u'value'))
    f2 = yield From(transaction.set(u'another_key', u'another_value'))

    # Commit transaction
    yield From(transaction.exec())

    # Retrieve results
    result1 = yield From(f1)
    result2 = yield From(f2)

    # When finished, close the connection pool.
    connection.close()
```

It's recommended to use a large enough poolsize. A connection will be occupied as long as there's a transaction running in there.

4.1.4 Pubsub

By calling `start_subscribe` (either on the protocol, through the `Connection` class or through the `Pool` class), you can start a pubsub listener.

```
import trollius
from trollius import From
import trollius_redis

@trollius.coroutine
def example():
    # Create connection
    connection = yield From(trollius_redis.Connection.create(host=u'localhost', port=6379))

    # Create subscriber.
    subscriber = yield From(connection.start_subscribe())

    # Subscribe to channel.
    yield From(subscriber.subscribe([ u'our-channel' ]))

    # Inside a while loop, wait for incoming events.
    while True:
        reply = yield From(subscriber.next_published())
        print(u'Received: ', repr(reply.value), u'on channel', reply.channel)

    # When finished, close the connection.
    connection.close()
```

4.1.5 LUA Scripting

The `register_script` function – which can be used to register a LUA script – returns a `Script` instance. You can call its `run` method to execute this script.

```
import trollius
from trollius import From
import trollius_redis

code = \
u"""
local value = redis.call('GET', KEYS[1])
value = tonumber(value)
return value * ARGV[1]
"""

@trollius.coroutine
def example():
    connection = yield From(trollius_redis.Connection.create(
        host=u'localhost', port=6379))

    # Set a key
    yield From(connection.set(u'my_key', u'2'))

    # Register script
    multiply = yield From(connection.register_script(code))

    # Run script
    script_reply = yield From(multiply.run(keys=[u'my_key'], args=[u'5']))
```

```
result = yield From(script_reply.return_value())
print(result) # prints 2 * 5

# When finished, close the connection.
connection.close()
```

4.1.6 Raw bytes or UTF-8

The redis protocol only knows about bytes, but normally you want to use strings in your Python code. `trollius_redis` is helpful and installs an encoder that does this conversion automatically, using the UTF-8 codec. However, sometimes you want to access raw bytes. This is possible by passing a `BytesEncoder` instance to the connection, pool or protocol.

```
import trollius
from trollius import From
import trollius_redis

from trollius_redis.encoders import BytesEncoder

@trollius.coroutine
def example():
    # Create Redis connection
    connection = yield From(trollius_redis.Connection.create(
        host='localhost', port=6379, encoder=BytesEncoder()))

    # Set a key
    yield From(connection.set(b'my_key', b'my_value'))

    # When finished, close the connection.
    connection.close()
```

4.1.7 Scanning for keys

Redis has a few nice scanning utilities to discover keys in the database. They are rather low-level, but `trollius_redis` exposes a simple `Cursor` class that allows you to iterate over all the keys matching a certain pattern. Each call of the `fetchone()` coroutine will return the next match. You don't have to worry about accessing the server every `x` pages.

The following example will print all the keys in the database:

```
import trollius
from trollius import From
import trollius_redis

from trollius_redis.encoders import BytesEncoder

@trollius.coroutine
def example():
    cursor = yield From(protocol.scan(match=u'*'))
    while True:
        item = yield From(cursor.fetchone())
        if item is None:
            break
        else:
            print(item)
```


See the scanning utilities: `scan()`, `sscan()`, `hscan()` and `zscan()`

4.1.8 The RedisProtocol class

The most low level way of accessing the redis server through this library is probably by creating a connection with the `RedisProtocol` yourself. You can do it as follows:

```
import trollius
from trollius import From
import trollius_redis

@trollius.coroutine
def example():
    loop = trollius.get_event_loop()

    # Create Redis connection
    transport, protocol = yield From(loop.create_connection(
        trollius_redis.RedisProtocol, u'localhost', 6379))

    # Set a key
    yield From(protocol.set(u'my_key', u'my_value'))

    # Get a key
    result = yield From(protocol.get(u'my_key'))
    print(result)

if __name__ == '__main__':
    trollius.get_event_loop().run_until_complete(example())
```

Note: It is not recommended to use the Protocol class directly, because the low-level Redis implementation could change. Prefer the `Connection` or `Pool` class as demonstrated above if possible.

4.2 Reference

You can either use the `RedisProtocol` class directly, use the `Connection` class, or use the `Pool` wrapper which also offers connection pooling.

4.2.1 The Protocol

```
class trollius_redis.RedisProtocol(password=None, db=0, encoder=None, connection_lost_callback=None, enable_typechecking=True, loop=None)
```

The Redis Protocol implementation.

```
self.loop = trollius.get_event_loop()
transport, protocol = yield From(loop.create_connection(
    RedisProtocol, 'localhost', 6379))
```

Parameters

- **password** (Native Python type as defined by the `encoder` parameter) – Redis database password

- **encoder** (`BaseEncoder` instance.) – Encoder to use for encoding to or decoding from redis bytes to a native type. (Defaults to `UTF8Encoder`)
- **db** (*int*) – Redis database
- **enable_typechecking** (*bool*) – When `True`, check argument types for all redis commands. Normally you want to have this enabled.

append (*self, key, value*)
Append a value to a key

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) `int`

auth (*self, password*)
Authenticate to the server

Parameters **password** – Native Python type, as defined by `native_type`

Returns (Future of) `StatusReply`

bgrewriteaof (*self*)
Asynchronously rewrite the append-only file

Returns (Future of) `StatusReply`

bgsave (*self*)
Asynchronously save the dataset to disk

Returns (Future of) `StatusReply`

bitcount (*self, key, start=0, end=-1*)
Count the number of set bits (population counting) in a string.

Parameters

- **key** – Native Python type, as defined by `native_type`
- **start** – `int`
- **end** – `int`

Returns (Future of) `int`

bitop_and (*self, destkey, srckeys*)
Perform a bitwise AND operation between multiple keys.

Parameters

- **destkey** – Native Python type, as defined by `native_type`
- **srckeys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `int`

bitop_not (*self, destkey, key*)
Perform a bitwise NOT operation between multiple keys.

Parameters

- **destkey** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

bitop_or (*self*, *destkey*, *srckeys*)

Perform a bitwise OR operation between multiple keys.

Parameters

- **destkey** – Native Python type, as defined by `native_type`
- **srckeys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

bitop_xor (*self*, *destkey*, *srckeys*)

Perform a bitwise XOR operation between multiple keys.

Parameters

- **destkey** – Native Python type, as defined by `native_type`
- **srckeys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

blpop (*self*, *keys*, *timeout=0*)

Remove and get the first element in a list, or block until one is available. This will raise `TimeoutError` when the timeout was exceeded and Redis returns `None`.

Parameters

- **timeout** – int
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `BlockingPopReply`

brpop (*self*, *keys*, *timeout=0*)

Remove and get the last element in a list, or block until one is available. This will raise `TimeoutError` when the timeout was exceeded and Redis returns `None`.

Parameters

- **timeout** – int
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `BlockingPopReply`

brpoplpush (*self*, *source*, *destination*, *timeout=0*)

Pop a value from a list, push it to another list and return it; or block until one is available

Parameters

- **source** – Native Python type, as defined by `native_type`
- **destination** – Native Python type, as defined by `native_type`
- **timeout** – int

Returns (Future of) Native Python type, as defined by `native_type`

client_getname (*self*)

Get the current connection name

Returns (Future of) Native Python type, as defined by `native_type`

client_kill (*self*, *address*)

Kill the connection of a client *address* should be an “ip:port” string.

Parameters *address* – str

Returns (Future of) `StatusReply`

client_list (*self*)

Get the list of client connections

Returns (Future of) `InfoReply`

client_setname (*self*, *name*)

Set the current connection name

Returns (Future of) `StatusReply`

config_get (*self*, *parameter*)

Get the value of a configuration parameter

Parameters *parameter* – str

Returns (Future of) `ConfigPairReply`

config_resetstat (*self*)

Reset the stats returned by INFO

Returns (Future of) `StatusReply`

config_rewrite (*self*)

Rewrite the configuration file with the in memory configuration

Returns (Future of) `StatusReply`

config_set (*self*, *parameter*, *value*)

Set a configuration parameter to the given value

Parameters

- *value* – str
- *parameter* – str

Returns (Future of) `StatusReply`

connection_made (*transport*)

dbsize (*self*)

Return the number of keys in the currently-selected database.

Returns (Future of) int

decr (*self*, *key*)

Decrement the integer value of a key by one

Parameters *key* – Native Python type, as defined by `native_type`

Returns (Future of) int

decrby (*self*, *key*, *increment*)

Decrement the integer value of a key by the given number

Parameters

- **increment** – int
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

delete (*self, keys*)

Delete a key

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

echo (*self, string*)

Echo the given string

Parameters **string** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type`

evalsha (*self, sha, keys=None, args=None*)

Evaluates a script cached on the server side by its SHA1 digest. Scripts are cached on the server side using the SCRIPT LOAD command.

The return type/value depends on the script.

This will raise a `ScriptKilledError` exception if the script was killed.

Parameters

- **sha** – str
- **args** – List or iterable of Native Python type, as defined by `native_type` or None
- **keys** – List or iterable of Native Python type, as defined by `native_type` or None

Returns (Future of) `EvalScriptReply`

exists (*self, key*)

Determine if a key exists

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) bool

expire (*self, key, seconds*)

Set a key's time to live in seconds

Parameters

- **seconds** – int
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

expireat (*self, key, timestamp*)

Set the expiration for a key as a UNIX timestamp

Parameters

- **key** – Native Python type, as defined by `native_type`
- **timestamp** – int

Returns (Future of) int

flushall (*self*)

Remove all keys from all databases

Returns (Future of) `StatusReply`

flushdb (*self*)

Delete all the keys of the currently selected DB. This command never fails.

Returns (Future of) `StatusReply`

get (*self, key*)

Get the value of a key

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or `None`

getbit (*self, key, offset*)

Returns the bit value at offset in the string value stored at key

Parameters

- **key** – Native Python type, as defined by `native_type`
- **offset** – int

Returns (Future of) `bool`

getset (*self, key, value*)

Set the string value of a key and return its old value

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or `None`

hdel (*self, key, fields*)

Delete one or more hash fields

Parameters

- **key** – Native Python type, as defined by `native_type`
- **fields** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `int`

hexists (*self, key, field*)

Returns if field is an existing field in the hash stored at key.

Parameters

- **key** – Native Python type, as defined by `native_type`
- **field** – Native Python type, as defined by `native_type`

Returns (Future of) `bool`

hget (*self, key, field*)

Get the value of a hash field

Parameters

- **key** – Native Python type, as defined by `native_type`

- **field** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or None

hgetall (*self, key*)

Get the value of a hash field

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) `DictReply`

hgetall_asdict (*self, key*)

Get the value of a hash field

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) dict

hincrby (*self, key, field, increment*)

Increment the integer value of a hash field by the given number Returns: the value at field after the increment operation.

Parameters

- **key** – Native Python type, as defined by `native_type`
- **field** – Native Python type, as defined by `native_type`

Returns (Future of) int

hincrbyfloat (*self, key, field, increment*)

Increment the float value of a hash field by the given amount Returns: the value at field after the increment operation.

Parameters

- **increment** – int or float
- **key** – Native Python type, as defined by `native_type`
- **field** – Native Python type, as defined by `native_type`

Returns (Future of) float

hkeys (*self, key*)

Get all the keys in a hash. (Returns a set)

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) `SetReply`

hkeys_asset (*self, key*)

Get all the keys in a hash. (Returns a set)

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) set

hlen (*self, key*)

Returns the number of fields contained in the hash stored at key.

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

hmget (*self, key, fields*)

Get the values of all the given hash fields

Parameters

- **key** – Native Python type, as defined by `native_type`
- **fields** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `ListReply`

hmget_aslist (*self, key, fields*)

Get the values of all the given hash fields

Parameters

- **key** – Native Python type, as defined by `native_type`
- **fields** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) list

hmset (*self, key, values*)

Set multiple hash fields to multiple values

Parameters

- **key** – Native Python type, as defined by `native_type`
- **values** – dict

Returns (Future of) `StatusReply`

hscan (*self, key, match='*'*)

Incrementally iterate hash fields and associated values Also see: `scan()`

Parameters

- **key** – Native Python type, as defined by `native_type`
- **match** – Native Python type, as defined by `native_type`

Returns (Future of) `DictCursor`

hset (*self, key, field, value*)

Set the string value of a hash field

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`
- **field** – Native Python type, as defined by `native_type`

Returns (Future of) int

hsetnx (*self, key, field, value*)

Set the value of a hash field, only if the field does not exist

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`
- **field** – Native Python type, as defined by `native_type`

Returns (Future of) int

hvals (*self*, *key*)

Get all the values in a hash. (Returns a list)

Parameters *key* – Native Python type, as defined by `native_type`

Returns (Future of) `ListReply`

hvals_aslist (*self*, *key*)

Get all the values in a hash. (Returns a list)

Parameters *key* – Native Python type, as defined by `native_type`

Returns (Future of) list

in_blocking_call

True when waiting for answer to blocking command.

in_pubsub

True when the protocol is in pubsub mode.

in_transaction

True when we're inside a transaction.

in_use

True when this protocol is in use.

incr (*self*, *key*)

Increment the integer value of a key by one

Parameters *key* – Native Python type, as defined by `native_type`

Returns (Future of) int

incrby (*self*, *key*, *increment*)

Increment the integer value of a key by the given amount

Parameters

- **increment** – int
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

info (*self*, *section=None*)

Get information and statistics about the server

Parameters *section* – Native Python type, as defined by `native_type` or None

Returns (Future of) `InfoReply`

is_connected

True when the underlying transport is connected.

keys (*self*, *pattern*)

Find all keys matching the given pattern.

Note: Also take a look at

`scan()`.

Parameters *pattern* – Native Python type, as defined by `native_type`

Returns (Future of) `ListReply`

keys_aslist (*self*, *pattern*)

Find all keys matching the given pattern.

Note: Also take a look at

`scan()`.

Parameters **pattern** – Native Python type, as defined by `native_type`

Returns (Future of) list

lastsave (*self*)

Get the UNIX time stamp of the last successful save to disk

Returns (Future of) int

lindex (*self*, *key*, *index*)

Get an element from a list by its index

Parameters

- **index** – int
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or None

linsert (*self*, *key*, *pivot*, *value*, *before=False*)

Insert an element before or after another element in a list

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`
- **pivot** – Native Python type, as defined by `native_type`

Returns (Future of) int

llen (*self*, *key*)

Returns the length of the list stored at key.

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

lpop (*self*, *key*)

Remove and get the first element in a list

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or None

lpush (*self*, *key*, *values*)

Prepend one or multiple values to a list

Parameters

- **key** – Native Python type, as defined by `native_type`
- **values** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

lpushx (*self, key, value*)

Prepend a value to a list, only if the list exists

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

lrange (*self, key, start=0, stop=-1*)

Get a range of elements from a list.

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) `ListReply`

lrange_aslist (*self, key, start=0, stop=-1*)

Get a range of elements from a list.

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) list

lrem (*self, key, count=0, value=''*)

Remove elements from a list

Parameters

- **key** – Native Python type, as defined by `native_type`
- **count** – int

Returns (Future of) int

lset (*self, key, index, value*)

Set the value of an element in a list by its index.

Parameters

- **index** – int
- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) `StatusReply`

ltrim (*self, key, start=0, stop=-1*)

Trim a list to the specified range

Parameters

- **stop** – int

- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) `StatusReply`

mget (*self*, *keys*)

Returns the values of all specified keys.

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `ListReply`

mget_aslist (*self*, *keys*)

Returns the values of all specified keys.

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) list

move (*self*, *key*, *database*)

Move a key to another database

Parameters

- **database** – int
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

multi (*self*, *watch=None*)

Start of transaction.

```
transaction = yield From(protocol.multi())

# Run commands in transaction
f1 = yield From(transaction.set(u'key', u'value'))
f2 = yield From(transaction.set(u'another_key', u'another_value'))

# Commit transaction
yield From(transaction.execute())

# Retrieve results (you can also use asyncio.tasks.gather)
result1 = yield From(f1)
result2 = yield From(f2)
```

returns A `trollius_redis.Transaction` instance.

Parameters **watch** – List or iterable of Native Python type, as defined by `native_type` or `None`

Returns (Future of) `trollius_redis.Transaction`

persist (*self*, *key*)

Remove the expiration from a key

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

pexpire (*self*, *key*, *milliseconds*)

Set a key's time to live in milliseconds

Parameters

- **key** – Native Python type, as defined by `native_type`
- **milliseconds** – int

Returns (Future of) int

pexpireat (*self*, *key*, *milliseconds_timestamp*)

Set the expiration for a key as a UNIX timestamp specified in milliseconds

Parameters

- **key** – Native Python type, as defined by `native_type`
- **milliseconds_timestamp** – int

Returns (Future of) int

ping (*self*)

Ping the server (Returns PONG)

Returns (Future of) `StatusReply`

pttl (*self*, *key*)

Get the time to live for a key in milliseconds

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

publish (*self*, *channel*, *message*)

Post a message to a channel (Returns the number of clients that received this message.)

Parameters

- **message** – Native Python type, as defined by `native_type`
- **channel** – Native Python type, as defined by `native_type`

Returns (Future of) int

pubsub_channels (*self*, *pattern=None*)

Lists the currently active channels. An active channel is a Pub/Sub channel with one ore more subscribers (not including clients subscribed to patterns).

Parameters **pattern** – Native Python type, as defined by `native_type` or None

Returns (Future of) `ListReply`

pubsub_channels_aslist (*self*, *pattern=None*)

Lists the currently active channels. An active channel is a Pub/Sub channel with one ore more subscribers (not including clients subscribed to patterns).

Parameters **pattern** – Native Python type, as defined by `native_type` or None

Returns (Future of) list

pubsub_numpat (*self*)

Returns the number of subscriptions to patterns (that are performed using the PSUBSCRIBE command). Note that this is not just the count of clients subscribed to patterns but the total number of patterns all the clients are subscribed to.

Returns (Future of) int

pubsub_numsub (*self*, *channels*)

Returns the number of subscribers (not counting clients subscribed to patterns) for the specified channels.

Parameters channels – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `DictReply`

pubsub_numsub_asdict (*self*, *channels*)

Returns the number of subscribers (not counting clients subscribed to patterns) for the specified channels.

Parameters channels – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) dict

randomkey (*self*)

Return a random key from the keyspace

Returns (Future of) Native Python type, as defined by `native_type`

register_script (*self*, *script*)

Register a LUA script.

```
script = yield From(protocol.register_script(lua_code))
result = yield From(script.run(keys=[...], args=[...]))
```

Parameters script – str

Returns (Future of) `Script`

rename (*self*, *key*, *newkey*)

Rename a key

Parameters

- **newkey** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) `StatusReply`

renamenx (*self*, *key*, *newkey*)

Rename a key, only if the new key does not exist (Returns 1 if the key was successfully renamed.)

Parameters

- **newkey** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

rpop (*self, key*)

Remove and get the last element in a list

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or None

rpoplpush (*self, source, destination*)

Remove the last element in a list, append it to another list and return it

Parameters

- **source** – Native Python type, as defined by `native_type`
- **destination** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type` or None

rpush (*self, key, values*)

Append one or multiple values to a list

Parameters

- **key** – Native Python type, as defined by `native_type`
- **values** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

rpushx (*self, key, value*)

Append a value to a list, only if the list exists

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

sadd (*self, key, members*)

Add one or more members to a set

Parameters

- **key** – Native Python type, as defined by `native_type`
- **members** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

save (*self*)

Synchronously save the dataset to disk

Returns (Future of) `StatusReply`

scan (*self, match='*'*)

Walk through the keys space. You can either fetch the items one by one or in bulk.

```
cursor = yield From(protocol.scan(match=u' *'))
while True:
    item = yield From(cursor.fetchone())
    if item is None:
        break
```

```
    else:
        print (item)
```

```
cursor = yield From(protocol.scan(match=u'*'))
items = yield From(cursor.fetchall())
```

Also see: `sscan()`, `hscan()` and `zscan()`

Redis reference: <http://redis.io/commands/scan>

Parameters `match` – Native Python type, as defined by `native_type`

Returns (Future of) `Cursor`

scard (*self*, *key*)

Get the number of members in a set

Parameters `key` – Native Python type, as defined by `native_type`

Returns (Future of) `int`

script_exists (*self*, *shas*)

Check existence of scripts in the script cache.

Parameters `shas` – List or iterable of `str`

Returns (Future of) List or iterable of `bool`

script_flush (*self*)

Remove all the scripts from the script cache.

Returns (Future of) `StatusReply`

script_kill (*self*)

Kill the script currently in execution. This raises `NoRunningScriptError` when there are no scripts running.

Returns (Future of) `StatusReply`

script_load (*self*, *script*)

Load script, returns sha1

Parameters `script` – `str`

Returns (Future of) `str`

sdiff (*self*, *keys*)

Subtract multiple sets

Parameters `keys` – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `SetReply`

sdiff_asset (*self*, *keys*)

Subtract multiple sets

Parameters `keys` – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `set`

sdiffstore (*self*, *destination*, *keys*)

Subtract multiple sets and store the resulting set in a key

Parameters

- **destination** – Native Python type, as defined by `native_type`
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

select (*self*, *db*)

Change the selected database for the current connection

Parameters *db* – int

Returns (Future of) `StatusReply`

set (*self*, *key*, *value*, *expire=None*, *pexpire=None*, *only_if_not_exists=False*, *only_if_exists=False*)

Set the string value of a key

```
yield From(protocol.set(u'key', u'value'))
result = yield From(protocol.get(u'key'))
assert result == 'value'
```

To set a value and its expiration, only if key not exists, do:

```
yield From(protocol.set(
    u'key', u'value', expire=1, only_if_not_exists=True))
```

This will send: SET key value EX 1 NX at the network. To set value and its expiration in milliseconds, but only if

key already exists:

```
yield From(protocol.set(
    u'key', u'value', pexpire=1000, only_if_exists=True))
```

Parameters

- **value** – Native Python type, as defined by `native_type`
- **pexpire** – int or None
- **expire** – int or None
- **only_if_not_exists** – bool
- **key** – Native Python type, as defined by `native_type`
- **only_if_exists** – bool

Returns (Future of) `StatusReply` or None

setbit (*self*, *key*, *offset*, *value*)

Sets or clears the bit at offset in the string value stored at key

Parameters

- **value** – bool
- **key** – Native Python type, as defined by `native_type`
- **offset** – int

Returns (Future of) bool

setex (*self, key, seconds, value*)

Set the string value of a key with expire

Parameters

- **seconds** – int
- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) `StatusReply`

setnx (*self, key, value*)

Set the string value of a key if it does not exist. Returns True if value is successfully set

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) bool

shutdown (*self, save=False*)

Synchronously save the dataset to disk and then shut down the server

Returns (Future of) `StatusReply`

sinter (*self, keys*)

Intersect multiple sets

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `SetReply`

sinter_asset (*self, keys*)

Intersect multiple sets

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) set

sinterstore (*self, destination, keys*)

Intersect multiple sets and store the resulting set in a key

Parameters

- **destination** – Native Python type, as defined by `native_type`
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

sismember (*self, key, value*)

Determine if a given value is a member of a set

Parameters

- **value** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) bool

smembers (*self, key*)

Get all the members in a set

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) `SetReply`

smembers_asset (*self, key*)

Get all the members in a set

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) set

smove (*self, source, destination, value*)

Move a member from one set to another

Parameters

- **value** – Native Python type, as defined by `native_type`
- **source** – Native Python type, as defined by `native_type`
- **destination** – Native Python type, as defined by `native_type`

Returns (Future of) int

spop (*self, key*)

Removes and returns a random element from the set value stored at key.

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) Native Python type, as defined by `native_type`

srandmember (*self, key, count=1*)

Get one or multiple random members from a set (Returns a list of members, even when count==1)

Parameters

- **key** – Native Python type, as defined by `native_type`
- **count** – int

Returns (Future of) `SetReply`

srandmember_asset (*self, key, count=1*)

Get one or multiple random members from a set (Returns a list of members, even when count==1)

Parameters

- **key** – Native Python type, as defined by `native_type`
- **count** – int

Returns (Future of) set

srem (*self, key, members*)

Remove one or more members from a set

Parameters

- **key** – Native Python type, as defined by `native_type`

- **members** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

sscan (*self*, *key*, *match*='*')

Incrementally iterate set elements

Also see: `scan()`

Parameters

- **key** – Native Python type, as defined by `native_type`
- **match** – Native Python type, as defined by `native_type`

Returns (Future of) `SetCursor`

start_subscribe (*self*, **a*)

Start a pubsub listener.

```
# Create subscription
subscription = yield From(protocol.start_subscribe())
yield From(subscription.subscribe(['key']))
yield From(subscription.psubscribe(['pattern*']))

while True:
    result = yield From(subscription.next_published())
    print(result)

returns Subscription
```

Returns (Future of) `trollius_redis.Subscription`

strlen (*self*, *key*)

Returns the length of the string value stored at key. An error is returned when key holds a non-string value.

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

sunion (*self*, *keys*)

Add multiple sets

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) `SetReply`

sunion_asset (*self*, *keys*)

Add multiple sets

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) set

sunionstore (*self*, *destination*, *keys*)

Add multiple sets and store the resulting set in a key

Parameters

- **destination** – Native Python type, as defined by `native_type`
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

ttl (*self*, *key*)

Get the time to live for a key

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

type (*self*, *key*)

Determine the type stored at key

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) `StatusReply`

watch (*self*, *keys*)

Watch keys (pulled in from `asyncio-redis` ac9467ef381523da1aeee293fac71e443b3c018f)

```
# Watch keys for concurrent updates
yield From(protocol.watch([u'key', u'other_key']))

value = yield From(protocol.get(u'key'))
another_value = yield From(protocol.get(u'another_key'))

transaction = yield From(protocol.multi())

f1 = yield From(transaction.set(u'key', another_value))
f2 = yield From(transaction.set(u'another_key', value))

# Commit transaction
yield From(transaction.exec())

# Retrieve results
yield From(f1)
yield From(f2)
```

Parameters **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) None

zadd (*self*, *key*, *values*)

Add one or more members to a sorted set, or update its score if it already exists

```
yield protocol.zadd('myszset', { 'key': 4, 'key2': 5 })
```

Parameters

- **key** – Native Python type, as defined by `native_type`
- **values** – dict

Returns (Future of) int

zcard (*self*, *key*)

Get the number of members in a sorted set

Parameters **key** – Native Python type, as defined by `native_type`

Returns (Future of) int

zcount (*self, key, min, max*)

Count the members in a sorted set with scores within the given values

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – ZScoreBoundary
- **max** – ZScoreBoundary

Returns (Future of) int

zincrby (*self, key, increment, member*)

Increment the score of a member in a sorted set

Parameters

- **increment** – float
- **member** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) float

zinterstore (*self, destination, keys, weights=None, aggregate='SUM'*)

Intersect multiple sorted sets and store the resulting sorted set in a new key

Parameters

- **destination** – Native Python type, as defined by `native_type`
- **weights** – None or List or iterable of float
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

zrange (*self, key, start=0, stop=-1*)

Return a range of members in a sorted set, by index.

You can do the following to receive the slice of the sorted set as a python dict (mapping the keys to their scores):

```
result = yield protocol.zrange('myzset', start=10, stop=20)
my_dict = yield result.asdict()
```

or the following to retrieve it as a list of keys:

```
result = yield protocol.zrange('myzset', start=10, stop=20)
my_dict = yield result.aslist()
```

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`

- **start** – int

Returns (Future of) `ZRangeReply`

zrange_asdict (*self*, *key*, *start=0*, *stop=-1*)

Return a range of members in a sorted set, by index.

You can do the following to receive the slice of the sorted set as a python dict (mapping the keys to their scores):

```
result = yield protocol.zrange('myzset', start=10, stop=20)
my_dict = yield result.asdict()
```

or the following to retrieve it as a list of keys:

```
result = yield protocol.zrange('myzset', start=10, stop=20)
my_dict = yield result.aslist()
```

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) dict

zrangebyscore (*self*, *key*, *min=ZScoreBoundary(value='-inf', exclude_boundary=False)*,
max=ZScoreBoundary(value='+inf', exclude_boundary=False))

Return a range of members in a sorted set, by score

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – `ZScoreBoundary`
- **max** – `ZScoreBoundary`

Returns (Future of) `ZRangeReply`

zrangebyscore_asdict (*self*, *key*, *min=ZScoreBoundary(value='-inf', exclude_boundary=False)*,
max=ZScoreBoundary(value='+inf', exclude_boundary=False))

Return a range of members in a sorted set, by score

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – `ZScoreBoundary`
- **max** – `ZScoreBoundary`

Returns (Future of) dict

zrank (*self*, *key*, *member*)

Determine the index of a member in a sorted set

Parameters

- **member** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int or None

zrem (*self*, *key*, *members*)

Remove one or more members from a sorted set

Parameters

- **key** – Native Python type, as defined by `native_type`
- **members** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

zremrangebyrank (*self*, *key*, *min=0*, *max=-1*)

Remove all members in a sorted set within the given indexes

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – int
- **max** – int

Returns (Future of) int

zremrangebyscore (*self*, *key*, *min=ZScoreBoundary(value='-inf', exclude_boundary=False)*,
max=ZScoreBoundary(value='+inf', exclude_boundary=False))

Remove all members in a sorted set within the given scores

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – `ZScoreBoundary`
- **max** – `ZScoreBoundary`

Returns (Future of) int

zrevrange (*self*, *key*, *start=0*, *stop=-1*)

Return a range of members in a reversed sorted set, by index.

You can do the following to receive the slice of the sorted set as a python dict (mapping the keys to their scores):

```
my_dict = yield protocol.zrevrange_asdict(  
    'myzset', start=10, stop=20)
```

or the following to retrieve it as a list of keys:

```
zrange_reply = yield protocol.zrevrange(  
    'myzset', start=10, stop=20)  
my_dict = yield zrange_reply.aslist()
```

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) `ZRangeReply`

zrevrange_asdict (*self*, *key*, *start=0*, *stop=-1*)

Return a range of members in a reversed sorted set, by index.

You can do the following to receive the slice of the sorted set as a python dict (mapping the keys to their scores):

```
my_dict = yield protocol.zrevrange_asdict(
    'myzset', start=10, stop=20)
```

or the following to retrieve it as a list of keys:

```
zrange_reply = yield protocol.zrevrange(
    'myzset', start=10, stop=20)
my_dict = yield zrange_reply.aslist()
```

Parameters

- **stop** – int
- **key** – Native Python type, as defined by `native_type`
- **start** – int

Returns (Future of) dict

zrevrangebyscore (*self*, *key*, *max=ZScoreBoundary(value='+inf', exclude_boundary=False)*,
min=ZScoreBoundary(value='-inf', exclude_boundary=False))

Return a range of members in a sorted set, by score, with scores ordered from high to low

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – `ZScoreBoundary`
- **max** – `ZScoreBoundary`

Returns (Future of) `ZRangeReply`

zrevrangebyscore_asdict (*self*, *key*, *max=ZScoreBoundary(value='+inf', exclude_boundary=False)*,
min=ZScoreBoundary(value='-inf', exclude_boundary=False))

Return a range of members in a sorted set, by score, with scores ordered from high to low

Parameters

- **key** – Native Python type, as defined by `native_type`
- **min** – `ZScoreBoundary`
- **max** – `ZScoreBoundary`

Returns (Future of) dict

zrevrank (*self*, *key*, *member*)

Determine the index of a member in a sorted set, with scores ordered from high to low

Parameters

- **member** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) int or None

zscan (*self*, *key*, *match='*'*)

Incrementally iterate sorted sets elements and associated scores Also see: `scan()`

Parameters

- **key** – Native Python type, as defined by `native_type`
- **match** – Native Python type, as defined by `native_type`

Returns (Future of) `DictCursor`

zscore (*self*, *key*, *member*)

Get the score associated with the given member in a sorted set

Parameters

- **member** – Native Python type, as defined by `native_type`
- **key** – Native Python type, as defined by `native_type`

Returns (Future of) float or None

zunionstore (*self*, *destination*, *keys*, *weights=None*, *aggregate='SUM'*)

Add multiple sorted sets and store the resulting sorted set in a new key

Parameters

- **destination** – Native Python type, as defined by `native_type`
- **weights** – None or List or iterable of float
- **keys** – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) int

class `trollius_redis.HiRedisProtocol` (*password=None*, *db=0*, *encoder=None*, *connection_lost_callback=None*, *enable_typechecking=True*, *loop=None*)

Protocol implementation that uses the `hiredis` library for parsing the incoming data. This will be faster in many cases, but not necessarily always.

It does not (yet) support streaming of multibulk replies, which means that you won't see the first item of a multi bulk reply, before the whole response has been parsed.

4.2.2 Encoders

class `trollius_redis.encoders.BaseEncoder`

Abstract base class for all encoders.

decode_to_native (*data*)

Decodes network bytes to a Python native type. It should always be the reverse operation of `encode_from_native`.

encode_from_native (*data*)

Encodes the native Python type to network bytes. Usually this will encode a string object to bytes using the UTF-8 encoding. You can either override this function, or set the *encoding* attribute.

native_type = None

class `trollius_redis.encoders.UTF8Encoder`

Encode strings to and from utf-8 bytes.

class `trollius_redis.encoders.BytesEncoder`

For raw access to the Redis database.

native_type

alias of `bytes`

4.2.3 Connection

class `trollius_redis.Connection`

Wrapper around the protocol and transport which takes care of establishing the connection and reconnecting it.

```
connection = yield From(Connection.create(host='localhost', port=6379))
```

```
result = yield From(connection.set('key', 'value'))
```

close ()

Close the connection transport.

classmethod **create** (*host*='localhost', *port*=6379, *password*=None, *db*=0, *encoder*=None, *auto_reconnect*=True, *loop*=None, *enable_typechecking*=True, *protocol_class*=<class 'trollius_redis.protocol.RedisProtocol'>, ***unused_args*)

Parameters

- **host** (*str*) – Address, either host or unix domain socket path
- **port** (*int*) – TCP port. If port is 0 then host assumed to be unix socket path
- **password** (*bytes*) – Redis database password
- **db** (*int*) – Redis database
- **encoder** (`BaseEncoder` instance.) – Encoder to use for encoding to or decoding from redis bytes to a native type.
- **auto_reconnect** (*bool*) – Enable auto reconnect
- **loop** – (optional) asyncio event loop.
- **protocol_class** (`RedisProtocol`) – (optional) redis protocol implementation

static **from_uri** (*uri*, ***additional_args*)

Convert a redis:// or unix:// URI as supported in the redis-py driver.

Parameters

- **uri** (*str*) – Address, either host or unix domain socket path
- **encoder** (`BaseEncoder` instance.) – Encoder to use for encoding to or decoding from redis bytes to a native type.
- **auto_reconnect** (*bool*) – Enable auto reconnect
- **loop** – (optional) asyncio event loop.
- **protocol_class** (`RedisProtocol`) – (optional) redis protocol implementation

transport

The transport instance that the protocol is currently using.

4.2.4 Connection pool

class `trollius_redis.Pool`

Pool of connections. Each Takes care of setting up the connection and connection pooling.

When poolsize > 1 and some connections are in use because of transactions or blocking requests, the other are preferred.

```
pool = yield From(
    Pool.create(host='localhost', port=6379, poolsize=10))
result = yield From(connection.set('key', 'value'))
```

close()

Close all the connections in the pool.

connections_connected

The amount of open TCP connections.

connections_in_use

Return how many protocols are in use.

```
classmethod create(host='localhost', port=6379, password=None, db=0, encoder=None, pool-
    size=1, auto_reconnect=True, loop=None, protocol_class=<class 'trol-
    lius_redis.protocol.RedisProtocol'>)
```

Create a new connection pool instance.

Parameters

- **host** (*str*) – Address, either host or unix domain socket path
- **port** (*int*) – TCP port. If port is 0 then host assumed to be unix socket path
- **password** (*bytes*) – Redis database password
- **db** (*int*) – Redis database
- **encoder** (`BaseEncoder` instance.) – Encoder to use for encoding to or decoding from redis bytes to a native type.
- **poolsize** (*int*) – The number of parallel connections.
- **auto_reconnect** (*bool*) – Enable auto reconnect
- **loop** – (optional) asyncio event loop.
- **protocol_class** (`RedisProtocol`) – (optional) redis protocol implementation

poolsize

Number of parallel connections in the pool.

register_script (*self, script*)

Register a LUA script.

```
script = yield From(protocol.register_script(lua_code))
result = yield From(script.run(keys=[...], args=[...]))
```

Parameters `script` – str

Returns (Future of) `Script`

4.2.5 Command replies

class `trollius_redis.replies.StatusReply` (*status*)
 Wrapper for Redis status replies. (for messages like OK, QUEUED, etc...)

class `trollius_redis.replies.DictReply` (*multibulk_reply*)
 Container for a dict reply.

The content can be retrieved by calling `asdict()` which returns a Python dictionary. Or by iterating over it:

```
for f in dict_reply:
    key, value = yield from f
    print(key, value)
```

asdict()
 Return the result as a Python dictionary.

class `trollius_redis.replies.ListReply` (*multibulk_reply*)
 Redis list result. The content can be retrieved by calling `aslist()` or by iterating over it or by iterating over it

```
for f in list_reply:
    item = yield from f
    print(item)
```

aslist()
 Return the result as a Python list.

class `trollius_redis.replies.SetReply` (*multibulk_reply*)
 Redis set result. The content can be retrieved by calling `asset()` or by iterating over it

```
for f in set_reply:
    item = yield from f
    print(item)
```

asset()
 Return the result as a Python set.

class `trollius_redis.replies.ZRangeReply` (*multibulk_reply*)
 Container for a zrange query result.

class `trollius_redis.replies.PubSubReply` (*channel, value, pattern=None*)
 Received pubsub message.

channel
 Channel name

pattern
 The pattern to which we subscribed or *None* otherwise

value
 Received PubSub value

class `trollius_redis.replies.BlockingPopReply` (*list_name, value*)
`blpop()` or `brpop()` reply

list_name
 List name.

value
 Popped value

class `trollius_redis.replies.InfoReply` (*data*)
`info()` reply.

```
class trollius_redis.replies.ClientListReply (data)
    client_list () reply.
```

4.2.6 Cursors

```
class trollius_redis.cursors.Cursor (name, scanfunc)
    Cursor for walking through the results of a scan query.
```

```
    fetchall ()
        Coroutine that reads all the items in one list.
```

```
    fetchone ()
        Coroutines that returns the next item. It returns None after the last item.
```

```
class trollius_redis.cursors.SetCursor (name, scanfunc)
    Cursor for walking through the results of a sscan query.
```

```
class trollius_redis.cursors.DictCursor (name, scanfunc)
    Cursor for walking through the results of a hscan query.
```

```
    fetchall ()
        Coroutine that reads all the items in one dictionary.
```

```
    fetchone ()
        Get next { key: value } tuple It returns None after the last item.
```

```
class trollius_redis.cursors.ZCursor (name, scanfunc)
    Cursor for walking through the results of a zscan query.
```

4.2.7 Utils

```
class trollius_redis.ZScoreBoundary (value, exclude_boundary=False)
    Score boundary for a sorted set. for queries like zrangebyscore and similar
```

Parameters

- **value** (*float*) – Value for the boundary.
- **exclude_boundary** (*bool*) – Exclude the boundary.

```
class trollius_redis.Transaction (protocol)
    Transaction context. This is a proxy to a RedisProtocol instance. Every redis command called on this
    object will run inside the transaction. The transaction can be finished by calling either discard or exec.
```

More info: <http://redis.io/topics/transactions>

```
    discard ()
        Discard all commands issued after MULTI
```

```
    execute ()
        Execute transaction.

        This can raise a TransactionError when the transaction fails.
```

```
    unwatch ()
        Forget about all watched keys
```

```
class trollius_redis.Subscription (protocol)
    Pubsub subscription
```

next_published()

Coroutine which waits for next pubsub message to be received and returns it.

Returns instance of :class:`PubSubReply`

<trollius_redis.replies.PubSubReply>

psubscribe(patterns)

`_psubscribe(self, patterns)` Listen for messages published to channels matching the given

patterns

Parameters patterns – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) None

punsubscribe(patterns)

`_punsubscribe(self, patterns)`

Stop listening for messages posted to channels matching the given patterns

Parameters patterns – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) None

subscribe(channels)

`_subscribe(self, channels)` Listen for messages published to the given channels

Parameters channels – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) None

unsubscribe(channels)

`_unsubscribe(self, channels)` Stop listening for messages posted to the given channels

Parameters channels – List or iterable of Native Python type, as defined by `native_type`

Returns (Future of) None

class `trollius_redis.Script` (*sha, code, get_evalsha_func*)

Lua script.

run (*keys=[], args=[]*)

Returns a coroutine that executes the script.

```
script_reply = yield From(script.run(keys=[], args=[]))
```

```
# If the LUA script returns something, retrieve the return value
```

```
result = yield From(script_reply.return_value())
```

This will raise a `ScriptKilledError` exception if the script was killed.

class `trollius_redis.ZAggregate`

Aggregation method for zinterstore and zunionstore.

4.2.8 Exceptions

class `trollius_redis.exceptions.TransactionError`
Transaction failed.

class `trollius_redis.exceptions.NotConnectedError` (*message='Not connected'*)
Protocol is not connected.

class `trollius_redis.exceptions.TimeoutError`
Timeout during blocking pop.

class `trollius_redis.exceptions.ConnectionLostError` (*exc*)
Connection lost during query. (Special case of `NotConnectedError`.)

class `trollius_redis.exceptions.NoAvailableConnectionsInPoolError` (*message='Not connected'*)
When the connection pool has no available connections.

class `trollius_redis.exceptions.ScriptKilledError`
Script was killed during an `evalsha` call.

class `trollius_redis.exceptions.NoRunningScriptError`
`script_kill` was called while no script was running.

A

append() (trollius_redis.RedisProtocol method), 14
 asdict() (trollius_redis.replies.DictReply method), 41
 aslist() (trollius_redis.replies.ListReply method), 41
 asset() (trollius_redis.replies.SetReply method), 41
 auth() (trollius_redis.RedisProtocol method), 14

B

BaseEncoder (class in trollius_redis.encoders), 38
 bgrewriteof() (trollius_redis.RedisProtocol method), 14
 bgsave() (trollius_redis.RedisProtocol method), 14
 bitcount() (trollius_redis.RedisProtocol method), 14
 bitop_and() (trollius_redis.RedisProtocol method), 14
 bitop_not() (trollius_redis.RedisProtocol method), 14
 bitop_or() (trollius_redis.RedisProtocol method), 15
 bitop_xor() (trollius_redis.RedisProtocol method), 15
 BlockingPopReply (class in trollius_redis.replies), 41
 blpop() (trollius_redis.RedisProtocol method), 15
 brpop() (trollius_redis.RedisProtocol method), 15
 brpoplpush() (trollius_redis.RedisProtocol method), 15
 BytesEncoder (class in trollius_redis.encoders), 39

C

channel (trollius_redis.replies.PubSubReply attribute), 41
 client_getname() (trollius_redis.RedisProtocol method), 15
 client_kill() (trollius_redis.RedisProtocol method), 16
 client_list() (trollius_redis.RedisProtocol method), 16
 client_setname() (trollius_redis.RedisProtocol method), 16
 ClientListReply (class in trollius_redis.replies), 42
 close() (trollius_redis.Connection method), 39
 close() (trollius_redis.Pool method), 40
 config_get() (trollius_redis.RedisProtocol method), 16
 config_resetstat() (trollius_redis.RedisProtocol method), 16
 config_rewrite() (trollius_redis.RedisProtocol method), 16
 config_set() (trollius_redis.RedisProtocol method), 16
 Connection (class in trollius_redis), 39

connection_made() (trollius_redis.RedisProtocol method), 16
 ConnectionLostError (class in trollius_redis.exceptions), 44
 connections_connected (trollius_redis.Pool attribute), 40
 connections_in_use (trollius_redis.Pool attribute), 40
 create() (trollius_redis.Connection class method), 39
 create() (trollius_redis.Pool class method), 40
 Cursor (class in trollius_redis.cursors), 42

D

dbsize() (trollius_redis.RedisProtocol method), 16
 decode_to_native() (trollius_redis.encoders.BaseEncoder method), 38
 DECR() (trollius_redis.RedisProtocol method), 16
 DECRBY() (trollius_redis.RedisProtocol method), 16
 delete() (trollius_redis.RedisProtocol method), 17
 DictCursor (class in trollius_redis.cursors), 42
 DictReply (class in trollius_redis.replies), 41
 discard() (trollius_redis.Transaction method), 42

E

echo() (trollius_redis.RedisProtocol method), 17
 encode_from_native() (trollius_redis.encoders.BaseEncoder method), 38
 evalsha() (trollius_redis.RedisProtocol method), 17
 execute() (trollius_redis.Transaction method), 42
 exists() (trollius_redis.RedisProtocol method), 17
 expire() (trollius_redis.RedisProtocol method), 17
 expireat() (trollius_redis.RedisProtocol method), 17

F

fetchall() (trollius_redis.cursors.Cursor method), 42
 fetchall() (trollius_redis.cursors.DictCursor method), 42
 fetchone() (trollius_redis.cursors.Cursor method), 42
 fetchone() (trollius_redis.cursors.DictCursor method), 42
 flushall() (trollius_redis.RedisProtocol method), 17
 flushdb() (trollius_redis.RedisProtocol method), 18
 from_uri() (trollius_redis.Connection static method), 39

G

get() (trollius_redis.RedisProtocol method), 18
 getbit() (trollius_redis.RedisProtocol method), 18
 getset() (trollius_redis.RedisProtocol method), 18

H

hdel() (trollius_redis.RedisProtocol method), 18
 hexists() (trollius_redis.RedisProtocol method), 18
 hget() (trollius_redis.RedisProtocol method), 18
 hgetall() (trollius_redis.RedisProtocol method), 19
 hgetall_asdict() (trollius_redis.RedisProtocol method), 19
 hincrby() (trollius_redis.RedisProtocol method), 19
 hincrbyfloat() (trollius_redis.RedisProtocol method), 19
 HiRedisProtocol (class in trollius_redis), 38
 hkeys() (trollius_redis.RedisProtocol method), 19
 hkeys_asset() (trollius_redis.RedisProtocol method), 19
 hlen() (trollius_redis.RedisProtocol method), 19
 hmget() (trollius_redis.RedisProtocol method), 19
 hmget_aslist() (trollius_redis.RedisProtocol method), 20
 hmset() (trollius_redis.RedisProtocol method), 20
 hscan() (trollius_redis.RedisProtocol method), 20
 hset() (trollius_redis.RedisProtocol method), 20
 hsetnx() (trollius_redis.RedisProtocol method), 20
 hvals() (trollius_redis.RedisProtocol method), 21
 hvals_aslist() (trollius_redis.RedisProtocol method), 21

I

in_blocking_call (trollius_redis.RedisProtocol attribute), 21
 in_pubsub (trollius_redis.RedisProtocol attribute), 21
 in_transaction (trollius_redis.RedisProtocol attribute), 21
 in_use (trollius_redis.RedisProtocol attribute), 21
 incr() (trollius_redis.RedisProtocol method), 21
 incrby() (trollius_redis.RedisProtocol method), 21
 info() (trollius_redis.RedisProtocol method), 21
 InfoReply (class in trollius_redis.replies), 41
 is_connected (trollius_redis.RedisProtocol attribute), 21

K

keys() (trollius_redis.RedisProtocol method), 21
 keys_aslist() (trollius_redis.RedisProtocol method), 22

L

lastsave() (trollius_redis.RedisProtocol method), 22
 lindex() (trollius_redis.RedisProtocol method), 22
 linsert() (trollius_redis.RedisProtocol method), 22
 list_name (trollius_redis.replies.BlockingPopReply attribute), 41
 ListReply (class in trollius_redis.replies), 41
 llen() (trollius_redis.RedisProtocol method), 22
 lpop() (trollius_redis.RedisProtocol method), 22
 lpush() (trollius_redis.RedisProtocol method), 22
 lpushx() (trollius_redis.RedisProtocol method), 23

lrange() (trollius_redis.RedisProtocol method), 23
 lrange_aslist() (trollius_redis.RedisProtocol method), 23
 lrem() (trollius_redis.RedisProtocol method), 23
 lset() (trollius_redis.RedisProtocol method), 23
 ltrim() (trollius_redis.RedisProtocol method), 23

M

mget() (trollius_redis.RedisProtocol method), 24
 mget_aslist() (trollius_redis.RedisProtocol method), 24
 move() (trollius_redis.RedisProtocol method), 24
 multi() (trollius_redis.RedisProtocol method), 24

N

native_type (trollius_redis.encoders.BaseEncoder attribute), 39
 native_type (trollius_redis.encoders.BytesEncoder attribute), 39
 next_published() (trollius_redis.Subscription method), 42
 NoAvailableConnectionsInPoolError (class in trollius_redis.exceptions), 44
 NoRunningScriptError (class in trollius_redis.exceptions), 44
 NotConnectedError (class in trollius_redis.exceptions), 44

P

pattern (trollius_redis.replies.PubSubReply attribute), 41
 persist() (trollius_redis.RedisProtocol method), 24
 pexpire() (trollius_redis.RedisProtocol method), 24
 pexpireat() (trollius_redis.RedisProtocol method), 25
 ping() (trollius_redis.RedisProtocol method), 25
 Pool (class in trollius_redis), 40
 poolsize (trollius_redis.Pool attribute), 40
 psubscribe() (trollius_redis.Subscription method), 43
 pttl() (trollius_redis.RedisProtocol method), 25
 publish() (trollius_redis.RedisProtocol method), 25
 pubsub_channels() (trollius_redis.RedisProtocol method), 25
 pubsub_channels_aslist() (trollius_redis.RedisProtocol method), 25
 pubsub_numpat() (trollius_redis.RedisProtocol method), 25
 pubsub_numsub() (trollius_redis.RedisProtocol method), 26
 pubsub_numsub_asdict() (trollius_redis.RedisProtocol method), 26
 PubSubReply (class in trollius_redis.replies), 41
 punsubscribe() (trollius_redis.Subscription method), 43

R

randomkey() (trollius_redis.RedisProtocol method), 26
 RedisProtocol (class in trollius_redis), 13
 register_script() (trollius_redis.Pool method), 40

register_script() (trollius_redis.RedisProtocol method), 26
 rename() (trollius_redis.RedisProtocol method), 26
 renamenx() (trollius_redis.RedisProtocol method), 26
 rpop() (trollius_redis.RedisProtocol method), 27
 rpoplpush() (trollius_redis.RedisProtocol method), 27
 rpush() (trollius_redis.RedisProtocol method), 27
 rpushx() (trollius_redis.RedisProtocol method), 27
 run() (trollius_redis.Script method), 43

S

sadd() (trollius_redis.RedisProtocol method), 27
 save() (trollius_redis.RedisProtocol method), 27
 scan() (trollius_redis.RedisProtocol method), 27
 scard() (trollius_redis.RedisProtocol method), 28
 Script (class in trollius_redis), 43
 script_exists() (trollius_redis.RedisProtocol method), 28
 script_flush() (trollius_redis.RedisProtocol method), 28
 script_kill() (trollius_redis.RedisProtocol method), 28
 script_load() (trollius_redis.RedisProtocol method), 28
 ScriptKilledError (class in trollius_redis.exceptions), 44
 sdiff() (trollius_redis.RedisProtocol method), 28
 sdiff_asset() (trollius_redis.RedisProtocol method), 28
 sdiffstore() (trollius_redis.RedisProtocol method), 28
 select() (trollius_redis.RedisProtocol method), 29
 set() (trollius_redis.RedisProtocol method), 29
 setbit() (trollius_redis.RedisProtocol method), 29
 SetCursor (class in trollius_redis.cursors), 42
 setex() (trollius_redis.RedisProtocol method), 29
 setnx() (trollius_redis.RedisProtocol method), 30
 SetReply (class in trollius_redis.replies), 41
 shutdown() (trollius_redis.RedisProtocol method), 30
 sinter() (trollius_redis.RedisProtocol method), 30
 sinter_asset() (trollius_redis.RedisProtocol method), 30
 sinterstore() (trollius_redis.RedisProtocol method), 30
 sismember() (trollius_redis.RedisProtocol method), 30
 smembers() (trollius_redis.RedisProtocol method), 30
 smembers_asset() (trollius_redis.RedisProtocol method), 31
 smove() (trollius_redis.RedisProtocol method), 31
 spop() (trollius_redis.RedisProtocol method), 31
 srandmember() (trollius_redis.RedisProtocol method), 31
 srandmember_asset() (trollius_redis.RedisProtocol method), 31
 srem() (trollius_redis.RedisProtocol method), 31
 sscan() (trollius_redis.RedisProtocol method), 32
 start_subscribe() (trollius_redis.RedisProtocol method), 32
 StatusReply (class in trollius_redis.replies), 41
 strlen() (trollius_redis.RedisProtocol method), 32
 subscribe() (trollius_redis.Subscription method), 43
 Subscription (class in trollius_redis), 42
 sunion() (trollius_redis.RedisProtocol method), 32
 sunion_asset() (trollius_redis.RedisProtocol method), 32

sunionstore() (trollius_redis.RedisProtocol method), 32

T

TimeoutError (class in trollius_redis.exceptions), 44
 Transaction (class in trollius_redis), 42
 TransactionError (class in trollius_redis.exceptions), 44
 transport (trollius_redis.Connection attribute), 39
 ttl() (trollius_redis.RedisProtocol method), 33
 type() (trollius_redis.RedisProtocol method), 33

U

unsubscribe() (trollius_redis.Subscription method), 43
 unwatch() (trollius_redis.Transaction method), 42
 UTF8Encoder (class in trollius_redis.encoders), 39

V

value (trollius_redis.replies.BlockingPopReply attribute), 41
 value (trollius_redis.replies.PubSubReply attribute), 41

W

watch() (trollius_redis.RedisProtocol method), 33

Z

zadd() (trollius_redis.RedisProtocol method), 33
 ZAggregate (class in trollius_redis), 43
 zcard() (trollius_redis.RedisProtocol method), 33
 zcount() (trollius_redis.RedisProtocol method), 34
 ZCursor (class in trollius_redis.cursors), 42
 zincrby() (trollius_redis.RedisProtocol method), 34
 zinterstore() (trollius_redis.RedisProtocol method), 34
 zrange() (trollius_redis.RedisProtocol method), 34
 zrange_asdict() (trollius_redis.RedisProtocol method), 35
 zrangebyscore() (trollius_redis.RedisProtocol method), 35
 zrangebyscore_asdict() (trollius_redis.RedisProtocol method), 35
 ZRangeReply (class in trollius_redis.replies), 41
 zrank() (trollius_redis.RedisProtocol method), 35
 zrem() (trollius_redis.RedisProtocol method), 36
 zremrangebyrank() (trollius_redis.RedisProtocol method), 36
 zremrangebyscore() (trollius_redis.RedisProtocol method), 36
 zrevrange() (trollius_redis.RedisProtocol method), 36
 zrevrange_asdict() (trollius_redis.RedisProtocol method), 36
 zrevrangebyscore() (trollius_redis.RedisProtocol method), 37
 zrevrangebyscore_asdict() (trollius_redis.RedisProtocol method), 37
 zrevrank() (trollius_redis.RedisProtocol method), 37
 zscan() (trollius_redis.RedisProtocol method), 38

[zscore\(\)](#) (trollius_redis.RedisProtocol method), 38
[ZScoreBoundary](#) (class in trollius_redis), 42
[zunionstore\(\)](#) (trollius_redis.RedisProtocol method), 38