WiPy Tools Documentation

Release 0.0.57

Dwight Hubbard

Jul 29, 2017

Contents

micropython-redis provides a client for the Redis Key-Value store in a Python module.

CHAPTER 1

Table of Contents

Installation Instructions

Installing on CPython 3

Although micropython-redis is designed to function with micropython, it is supported on most python 3 interpreters.Use pip to install on Python3 or PyPy3.

```
$ pip install micropython-redis[all]
```

Installing on micropython

The installation process differs depending on the version of micropython being used. However the **upip** module is used to do the installation from the Python package repositories.

Installing on micropython unix

Use the micropython **upip** module to install on micropython. Different redis functionalities for the redis client are built into different modules. This allows for the installation of specific redis functionality without taaking up space for functionality that is not used. The following Will install the **uredis** module with all the component featues in the default micropython lib directory:

```
$ micropython -m upip install micropython-redis
$ micropython -m upip install micropython-redis.connection
$ micropython -m upip install micropython-redis.geo
$ micropython -m upip install micropython-redis.hash
$ micropython -m upip install micropython-redis.key
$ micropython -m upip install micropython-redis.list
$ micropython -m upip install micropython-redis.pubsub
```

Installing on micropython embedded platforms

To install on micropython embedded platforms:

Step 1. Create a lib directory to hold the python code for the platform

If you don't already have a library directory on the local system to hold the micropython packages, create one.

```
$ mkdir lib
```

Step 2. Set the MICROPATH environment variable to the full path of the lib directory.

Set the MICROPYPATH environment variable to point to the library directory. If you created the directory in the current directory as shown in **Step 1** you could run:

```
$ export MICROPYPATH="`pwd`/lib"
```

Step 3. Use the upip module to install micropython-redis into the lib directory.

Use the **upip** module to install the **micropython-redis** package.

```
$ micropython -m upip install micropython-redis
```

Install the redis packages with the desired redis functionality.

```
$ micropython -m upip install micropython-redis.connection
$ micropython -m upip install micropython-redis.geo
$ micropython -m upip install micropython-redis.hash
$ micropython -m upip install micropython-redis.key
$ micropython -m upip install micropython-redis.list
$ micropython -m upip install micropython-redis.pubsub
```

Step 4. Copy the lib directory to the embedded device.

Finally copy the lib directory you created to the root of the device filesystem. This varies depending on the method being used to put files on the device.

Usage

The micropython-redis module provides 3 different redis client interfaces.Each of which has different benefits and tradeoffs.

In the following section, the resource estimates are based on usage using the micropython unix port. The numbers will be different on other platforms, although the relative amounts used will remain roughly the same.

The uredis.Redis()/uredis.StrictRedis() classes.

This class is mostly compatible with the redis-py redis.Redis()/redis.StrictRedis() classes. Which allows a lot of existing code to work with little or no modifications.

The tradeoff is this requires the most resources. Currently importing this module uses currently ~20kb of memory.

The uredis_modular.* classes

The uredis_modular python module contains python modules that each implement a subset of the redis server functionality. Each of these modules can be used individully or they can be combined as mixins to create a Redis class with the desired functionality. The more functionality used, the more resources the resulting class will use.

All of these modules share a common redis.client which currently uses about ~6.5kb. Then each functionality module increases the resource usage by 1kb to 6kb depending on the compexity of the functinality submodule.

For example using the uredis_modular.list.List() submodule provides all of the redis server List functionality but uses 10kb to import.

Low level access using the uredis_modular.client.Client() class

The uredis.modular.client.Client() implements the redis protocol and can be used to communicate to the redis server direcctly without pulling in any of the funtionality submodules. This method uses the least resources, requiring ~6.5kb to import.

This method is not compatible with the redis-py bindings in any way.Also all communications will need to be encoded/decode to from byte strings prior to sending.

Code Documentation

Module

Redis Client for embedded python environments

class `uredis_modular.list.`**`List`**(*host=None*, *port=6379*, *password=None*)

Bases: `uredis_modular.client.Client`

Redis Client with support for all Redis List operations

`blpop`(**keys*, ***kwargs*)

Remove and get the first element of a list or block until one is available

Parameters

- **`*keys`** – Key or keys to get the first element from
- **`timeout`** (*`int, optional`*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

`brpop`(**keys*, ***kwargs*)

Remove and get the last element of a list or block until one is available

Parameters

- **`*keys`** – Key or keys to get the first element from
- **`timeout`** (*`int, optional`*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

`brpoplpush`(*src*, *dst*, *timeout=0*)
Remove and get the last element of a list and push it to the front of another list, blocking if there is no value to available.

Parameters

- **`src`** (*`str`*) – Key to pop the value from
- **`dst`** (*`str`*) – Key to prepend the value to
- **`timeout`** (*`int, optional`*) – Maximum time to block waiting for a key, if not specified or the value is 0, will wait forever.

Returns The bytestring of the value retrievied from the src

Return type bytes

`convert_to_bytestream`(*value*)
Return a bytestream of the value

Parameters **`value`** –

Returns A bytestream of the value

Return type bytes

`rpop`(*name*)
Remove and get the last element of a list

Parameters **`name`** (*`str`*) – Key to pop the value from

Returns The bytestring of the value retrievied from the src

Return type bytes

`send_redis_array_string`(*items*)
Send a redis array string

Parameters **`items`** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis.Redis() Class

class `uredis.`**`Redis`**(*host=None*, *port=6379*, *password=None*)
Bases: *`uredis_modular.connection.Connection`*, *`uredis_modular.geo.Geo`*, *`uredis_modular.hash.Hash`*, *`uredis_modular.key.Key`*, *`uredis_modular.list.List`*, *`uredis_modular.pubsub.PubSub`*, *`uredis_modular.set.Set`*, *`uredis_modular.sortedset.SortedSet`*, *`uredis_modular.string.String`*

Primary Redis Client Class.

This class provides a Redis Client with all the functionality of the supported subclasses.

This class is intended to be mostly compatible with the redis-py redis.Redis()/redis.StrictRedis() classes.

auth(*password*)
Authenticate to the server

Parameters **password** (*str*) – The password to authenticate with

blpop(**keys*, ***kwargs*)
Remove and get the first element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpop(**keys*, ***kwargs*)
Remove and get the last element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpoplpush(*src*, *dst*, *timeout=0*)
Remove and get the last element of a list and push it to the front of another list, blocking if there is no value to available.

Parameters

- **src** (*str*) – Key to pop the value from
- **dst** (*str*) – Key to prepend the value to
- **timeout** (*int, optional*) – Maximum time to block waiting for a key, if not specified or the value is 0, will wait forever.

Returns The bytestring of the value retrievied from the src

Return type bytes

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

echo(**args*)
Echo the given string

Parameters **message** (*str*) – The string to echo

geoadd(**args*)
Add one or more geospatial items in the geospatial index represented using a sorted set

Parameters ***args** – key longitude latitude member [longitude latitude member ...]

Returns The number of elements added to the sorted set, not including elements already existing for which the score was updated.

Return type int

geodist (**args*)
Returns the distance between two members of a geospatial index

Parameters ***args** – key member1 member2 [unit]

geohash (**args*)
Members of a geospatial index as geohash strings

Parameters ***args** – key member [member ...]

Returns Returns members of a geospatial index as standard geohash strings

Return type dict

geopos (**args*)
Return longitude and latitude of members of a geospatial index

Parameters ***args** – key member [key member ...]

Returns Returns members of a geospatial index as standard geohash strings

Return type dict

georadius (**args*)
Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a point

Parameters ***args** – key longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

georadiusbymember (**args*)
Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a member

Parameters ***args** – key member radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

hgetall (**args*)
" Returns all fields and values of the hash stored at key.

Returns Dictionary of all key/values from the field

Return type dict

hincrby (*key*, *field*, *increment*)
Increments the number stored at field in the hash stored at key by increment. If key does not exist, a new key holding a hash is created. If field does not exist the value is set to 0 before the operation is performed.

The range of values supported by HINCRBY is limited to 64 bit signed integers.

Parameters

- **key** (*str*) – Hash key to increment
- **field** (*str*) – Hash field to increment
- **increment** (*int*) – Amount to increment

Returns The value at field after the increment operation.

Return type int

ping(**args*)
Ping the server

quit(**args*)
Close the connection

rpop(*name*)
Remove and get the last element of a list

Parameters **name** (*str*) – Key to pop the value from

Returns The bytestring of the value retrievied from the src

Return type bytes

select(**args*)
Change the selected database

Parameters **index** (*int*) – The redis database number to switch to

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (list) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

zadd(*name*, **args*, ***kwargs*)
Set any number of score, element-name pairs to the key name. Pairs can be specified in two ways:

As *args, in the form of: score1, name1, score2, name2

Parameters

- **name** (*str*) – Keyname of the list
- ***args** – Sequency of name,score values

uredis.StrictRedis() Class

class uredis.**StrictRedis**(*host=None*, *port=6379*, *password=None*)
Bases: `uredis.uredis.Redis`

auth(*password*)
Authenticate to the server

Parameters **password** (*str*) – The password to authenticate with

blpop(**keys*, ***kwargs*)
Remove and get the first element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpop(**keys*, ***kwargs*)
Remove and get the last element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpoplpush(*src*, *dst*, *timeout=0*)

Remove and get the last element of a list and push it to the front of another list, blocking if there is no value to available.

Parameters

- **src** (*str*) – Key to pop the value from
- **dst** (*str*) – Key to prepend the value to
- **timeout** (*int, optional*) – Maximum time to block waiting for a key, if not specified or the value is 0, will wait forever.

Returns The bytestring of the value retrievied from the src

Return type bytes

convert_to_bytestream(*value*)

Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

echo(**args*)

Echo the given string

Parameters **message** (*str*) – The string to echo

geoadd(**args*)

Add one or more geospatial items in the geospatial index represented using a sorted set

Parameters ***args** – key longitude latitude member [longitude latitude member ...]

Returns The number of elements added to the sorted set, not including elements already existing for which the score was updated.

Return type int

geodist(**args*)

Returns the distance between two members of a geospatial index

Parameters ***args** – key member1 member2 [unit]

geohash(**args*)

Members of a geospatial index as geohash strings

Parameters ***args** – key member [member ...]

Returns Returns members of a geospatial index as standard geohash strings

Return type dict

geopos(**args*)

Return longitude and latitude of members of a geospatial index

> **Parameters** `*args` – key member [key member ...]
>
> **Returns** Returns members of a geospatial index as standard geohash strings
>
> **Return type** dict

georadius(**args*)
Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a point

> **Parameters** `*args` – key longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

georadiusbymember(**args*)
Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a member

> **Parameters** `*args` – key member radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

hgetall(**args*)
" Returns all fields and values of the hash stored at key.

> **Returns** Dictionary of all key/values from the field
>
> **Return type** dict

hincrby(*key*, *field*, *increment*)
Increments the number stored at field in the hash stored at key by increment. If key does not exist, a new key holding a hash is created. If field does not exist the value is set to 0 before the operation is performed.

The range of values supported by HINCRBY is limited to 64 bit signed integers.

> **Parameters**
>
> - **key** (*str*) – Hash key to increment
> - **field** (*str*) – Hash field to increment
> - **increment** (*int*) – Amount to increment
>
> **Returns** The value at field after the increment operation.
>
> **Return type** int

ping(**args*)
Ping the server

quit(**args*)
Close the connection

rpop(*name*)
Remove and get the last element of a list

> **Parameters** **name** (*str*) – Key to pop the value from
>
> **Returns** The bytestring of the value retrievied from the src
>
> **Return type** bytes

select(**args*)
Change the selected database

> **Parameters** **index** (*int*) – The redis database number to switch to

send_redis_array_string(*items*)
Send a redis array string

Parameters **`items`** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

`zadd`(*name*, **args*, ***kwargs*)

Set any number of score, element-name pairs to the key name. Pairs can be specified in two ways:

As *args, in the form of: score1, name1, score2, name2

Parameters

- **`name`** (*`str`*) – Keyname of the list
- **`*args`** – Sequency of name,score values

uredis_modular.connection.Connection() Class

class `uredis_modular.connection.`**`Connection`**(*host=None*, *port=6379*, *password=None*)

Bases: `uredis_modular.client.Client`

`auth`(*password*)

Authenticate to the server

Parameters **`password`** (*`str`*) – The password to authenticate with

`convert_to_bytestream`(*value*)

Return a bytestream of the value

Parameters **`value`** –

Returns A bytestream of the value

Return type bytes

`echo`(**args*)

Echo the given string

Parameters **`message`** (*`str`*) – The string to echo

`ping`(**args*)

Ping the server

`quit`(**args*)

Close the connection

`select`(**args*)

Change the selected database

Parameters **`index`** (*`int`*) – The redis database number to switch to

`send_redis_array_string`(*items*)

Send a redis array string

Parameters **`items`** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.geo.Geo() Class

class `uredis_modular.geo.`**`Geo`**(*host=None*, *port=6379*, *password=None*)

Bases: `uredis_modular.client.Client`

`convert_to_bytestream`(*value*)

Return a bytestream of the value

Parameters **`value`** –

Returns A bytestream of the value

Return type bytes

`geoadd`(**args*)

Add one or more geospatial items in the geospatial index represented using a sorted set

Parameters **`*args`** – key longitude latitude member [longitude latitude member ...]

Returns The number of elements added to the sorted set, not including elements already existing for which the score was updated.

Return type int

`geodist`(**args*)

Returns the distance between two members of a geospatial index

Parameters **`*args`** – key member1 member2 [unit]

`geohash`(**args*)

Members of a geospatial index as geohash strings

Parameters **`*args`** – key member [member ...]

Returns Returns members of a geospatial index as standard geohash strings

Return type dict

`geopos`(**args*)

Return longitude and latitude of members of a geospatial index

Parameters **`*args`** – key member [key member ...]

Returns Returns members of a geospatial index as standard geohash strings

Return type dict

`georadius`(**args*)

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a point

Parameters **`*args`** – key longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

`georadiusbymember`(**args*)

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a member

Parameters **`*args`** – key member radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITH-HASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]

`send_redis_array_string`(*items*)

Send a redis array string

Parameters **`items`** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.hash.Hash() Class

class `uredis_modular.hash.`**`Hash`**(*host=None*, *port=6379*, *password=None*)

Bases: `uredis_modular.client.Client`

`convert_to_bytestream`(*value*)

Return a bytestream of the value

Parameters **`value`** –

Returns A bytestream of the value

Return type bytes

`hgetall`(**args*)

” Returns all fields and values of the hash stored at key.

Returns Dictionary of all key/values from the field

Return type dict

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

Increments the number stored at field in the hash stored at key by increment. If key does not exist, a new key holding a hash is created. If field does not exist the value is set to 0 before the operation is performed.

The range of values supported by HINCRBY is limited to 64 bit signed integers.

Parameters

- **`key`** (*`str`*) – Hash key to increment
- **`field`** (*`str`*) – Hash field to increment
- **`increment`** (*`int`*) – Amount to increment

Returns The value at field after the increment operation.

Return type int

`send_redis_array_string`(*items*)

Send a redis array string

Parameters **`items`** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.hyperloglog.HyperLogLog() Class

uredis_modular.key.Key() Class

class `uredis_modular.key.`**`Key`**(*host=None*, *port=6379*, *password=None*)

Bases: `uredis_modular.client.Client`

`convert_to_bytestream`(*value*)

Return a bytestream of the value

Parameters **`value`** –

Returns A bytestream of the value

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (list) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.list.List() Class

class uredis_modular.list.**List**(*host=None*, *port=6379*, *password=None*)
Bases: uredis_modular.client.Client

Redis Client with support for all Redis List operations

blpop(**keys*, ***kwargs*)
Remove and get the first element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpop(**keys*, ***kwargs*)
Remove and get the last element of a list or block until one is available

Parameters

- ***keys** – Key or keys to get the first element from
- **timeout** (*int, optional*) – Maximum time to block waiting for the key, if not specified will wait forever.

Returns

Return type First element from the list, or None

brpoplpush(*src*, *dst*, *timeout=0*)
Remove and get the last element of a list and push it to the front of another list, blocking if there is no value to available.

Parameters

- **src** (*str*) – Key to pop the value from
- **dst** (*str*) – Key to prepend the value to
- **timeout** (*int, optional*) – Maximum time to block waiting for a key, if not specified or the value is 0, will wait forever.

Returns The bytestring of the value retrievied from the src

Return type bytes

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

rpop(*name*)
Remove and get the last element of a list

Parameters **name** (*str*) – Key to pop the value from

Returns The bytestring of the value retrievied from the src

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.pubsub.PubSub() Class

class `uredis_modular.pubsub.`**PubSub**(*host=None*, *port=6379*, *password=None*)
Bases: `uredis_modular.client.Client`

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (`list`) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.server.Server() Class

uredis_modular.set.Set() Class

class `uredis_modular.set.`**Set**(*host=None*, *port=6379*, *password=None*)
Bases: `uredis_modular.client.Client`

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (list) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.sortedset.SortedSet() Class

class uredis_modular.sortedset.**SortedSet**(*host=None*, *port=6379*, *password=None*)
Bases: uredis_modular.client.Client

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (list) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

zadd(*name*, **args*, ***kwargs*)
Set any number of score, element-name pairs to the key name. Pairs can be specified in two ways:

As *args, in the form of: score1, name1, score2, name2

Parameters

- **name** (*str*) – Keyname of the list
- ***args** – Sequency of name,score values

uredis_modular.string.String() Class

class uredis_modular.string.**String**(*host=None*, *port=6379*, *password=None*)
Bases: uredis_modular.client.Client

convert_to_bytestream(*value*)
Return a bytestream of the value

Parameters **value** –

Returns A bytestream of the value

Return type bytes

send_redis_array_string(*items*)
Send a redis array string

Parameters **items** (list) – The items to be in the redis array string

Returns Redis RESP bytestream representation of the array

Return type bytes

uredis_modular.transaction.Transaction() Class

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index