

php-redisCOPY

Redis~~~

QQ:104732401

EMAIL:cniiliuqi@126.com

Blog:<http://my.oschina.net/cniiliuqi>

RedisRedisRedis

Redis::__construct

Description

Creates a Redis client

Redis

Example

```
$redis = new Redis();
```

connect, open

Description

Connects to a Redis instance.

Redis

Parameters

host: string. can be a host, or the path to a unix domain socket

host: **HOST IPUNIX DOMAIN SOCKET**

port: int, optional

port: **,Redis**

timeout: float, value in seconds (optional, default is 0 meaning unlimited)

timeout: **.,0**

Return Value

BOOL: TRUE on success, FALSE on error.

Example

```
$redis->connect('127.0.0.1', 6379);  
$redis->connect('127.0.0.1'); // port 6379 by default  
$redis->connect('127.0.0.1', 6379, 2.5); // 2.5 sec timeout.  
$redis->connect('/tmp/redis.sock'); // unix domain socket.
```

pconnect, popen

Description

Connects to a Redis instance or reuse a connection already established with pconnect/popen.

pconnect/popenRedis

The connection will not be closed on close or end of request until the php process ends. So be patient on to many open FD's (specially on redis server side) when using persistent connections on many servers connecting to one redis server.

close()PHPconnectredis

Also more than one persistent connection can be made identified by either host + port + timeout or host + persistent_id or unix socket + timeout.

HOST+PORT+TIMEOUTHOST+persistent_id SOCKET+TIMEOUT

This feature is not available in threaded versions. pconnect and popen then working like their non persistent equivalents.

pconnectpopen

Parameters

host: string. can be a host, or the path to a unix domain socket

port: int, optional

timeout: float, value in seconds (optional, default is 0 meaning unlimited)

persistent_id: string. identity for the requested persistent connection

Return Value

BOOL: TRUE on success, FALSE on error.

Example

```
$redis->pconnect('127.0.0.1', 6379);  
$redis->pconnect('127.0.0.1'); // port 6379 by default - same connection like bef  
$redis->pconnect('127.0.0.1', 6379, 2.5); // 2.5 sec timeout and would be anothe  
$redis->pconnect('127.0.0.1', 6379, 2.5, 'x'); // x is sent as persistent_id and wou  
$redis->pconnect('/tmp/redis.sock'); // unix domain socket - would be another co
```

close

Description

Disconnects from the Redis instance, except when `pconnect` is used.

Redis, pconnect

```
$redis->close();
```

setOption

Description

Set client option.

Parameters

parameter name

parameter value

Return value

BOOL: TRUE on success, FALSE on error.

Example

```
$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_NONE);  
$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_PHP); //  
$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_IGBINAR  
  
$redis->setOption(Redis::OPT_PREFIX, 'myAppName:'); // use custom prefix c
```


getOption

Description

Get client option.

Parameters

parameter name

Return value

Parameter value.

Example

```
$redis->getOption(Redis::OPT_SERIALIZER); // return Redis::SERIALIZER
```

ping

Description

Check the current connection status

Parameters

(none)

Return Value

STRING: +PONG on success. Throws a RedisException object on connectivity error, as described above.

ThrowsRedisException

echo

Description

Sends a string to Redis, which replies with the same string

Redis,

Parameters

STRING: The message to send.

Return Value

STRING: the same message.

randomKey

Description

Returns a random key.

KEY

Parameters

None.

Return value

STRING: an existing key in redis.

STRINGREDISKEY

Example

```
$key = $redis->randomKey();  
$surprise = $redis->get($key); // who knows what's in there.
```

select

Description

Switches to a given database.

Parameters

INTEGER: dbindex, the database number to switch to.

ID

Return value

TRUE in case of success, FALSE in case of failure.

Example

(See following function)

move

Description

Moves a key to a different database.

KEY-VALUEDB

Parameters

Key: key, the key to move.

key:key

INTEGER: dbindex, the database number to move the key to.

:ID

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->select(0); // switch to DB 0
$redis->set('x', '42'); // write 42 to x
$redis->move('x', 1); // move to DB 1
$redis->select(1); // switch to DB 1
$redis->get('x'); // will return 42
```

rename, renameKey

Description

Renames a key.

KEY

Parameters

STRING: srckey, the key to rename.

STRING:KEYKEY

STRING: dstkey, the new name for the key.

STRING:KEYKEY

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->set('x', '42');  
$redis->rename('x', 'y');  
$redis->get('y'); // → 42  
$redis->get('x'); // → `FALSE`
```

renameNx

Description

Same as rename, but will not replace a key if the destination already exists. This is the same behaviour as setNx.

renamereNAMENxKEYKEYVALUEKEYKEY-VALUE
VALUEsrcKEYVALUEPHP

Example

```
$redis->set('x', '42');  
$redis->renameNx('x', 'y');  
$redis->get('y'); // → 42  
$redis->get('x'); // → 42
```

```
$redis->set('x','39');
```

```
$redis->get('x'); //->39
```

```
$redis->get('y'); //->42
```


setTimeout, expire, pexpire

Description

Sets an expiration date (a timeout) on an item. pexpire requires a TTL in milliseconds.

KEYpexpire

Parameters

Key: key. The key that will disappear.

Key:KEY

Integer: ttl. The key's remaining Time To Live, in seconds.

integer:

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->set('x', '42');  
$redis->setTimeout('x', 3); // x will disappear in 3 seconds.  
sleep(5);                // wait 5 seconds  
$redis->get('x');         // will return `FALSE`, as 'x' has expired.
```

expireAt, pexpireAt

Description

Sets an expiration date (a timestamp) on an item. pexpireAt requires a timestamp in milliseconds.

KEYUNIX

Parameters

Key: key. The key that will disappear.

Key:key

Integer: Unix timestamp. The key's date of death, in seconds from Epoch time.

integer:Unix

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->set('x', '42');  
$now = time(NULL); // current timestamp  
$redis->expireAt('x', $now + 3); // x will disappear in 3 seconds.  
sleep(5); // wait 5 seconds  
$redis->get('x'); // will return `FALSE`, as 'x' has expired.
```

keys, getKeys

Description

Returns the keys that match a certain pattern.

KEYS

Description

Parameters

STRING: pattern, using '*' as a wildcard.

STRING:

Return value

Array of STRING: The keys that match a certain pattern.

Example

```
$allKeys = $redis->keys('*'); // all keys will match this.  
$keyWithUserPrefix = $redis->keys('user*');
```

dbSize

Description

Returns the current database's size.

DBKEY

Parameters

None.

Return value

INTEGER: DB size, in number of keys.

Example

```
$count = $redis->dbSize();  
echo "Redis has $count keys\n";
```

auth

Description

Authenticate the connection using a password. *Warning:* The password is sent in plain-text over the network.

PASSWORDPASSWORD

Parameters

STRING: password

Return value

BOOL: TRUE if the connection is authenticated, FALSE otherwise.

Example

```
$redis->auth('foobared');
```

bgrewriteaof

Description

Starts the background rewrite of AOF (Append-Only File)

aof

Parameters

None.

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->bgrewriteaof();
```

slaveof

Description

Changes the slave status

Parameters

Either host (string) and port (int), or no parameter to stop being a slave.

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->slaveof('10.0.1.7', 6379);  
/* ... */  
$redis->slaveof();
```

object

Description

Describes the object pointed to by a key.

KEY

Parameters

The information to retrieve (string) and the key (string).

Info can be one of the following:

- "encoding"
- "refcount"
- "idletime"

Return value

STRING for "encoding", *LONG* for "refcount" and "idletime", *FALSE* if the key doesn't exist.

Example

```
$redis->object("encoding", "l"); // → ziplist  
$redis->object("refcount", "l"); // → 1  
$redis->object("idletime", "l"); // → 400 (in seconds, with a precision of 10 seconds)
```


save

Description

Performs a synchronous save.

Parameters

None.

Return value

BOOL: TRUE in case of success, FALSE in case of failure. If a save is already running, this command will fail and return FALSE.

Example

```
$redis->save();
```

bgsave

Description

Performs a background save.

Parameters

None.

Return value

BOOL: TRUE in case of success, FALSE in case of failure. If a save is already running, this command will fail and return FALSE.

Example

```
$redis->bgSave();
```

lastSave

Description

Returns the timestamp of the last disk save.

Parameters

None.

Return value

INT: timestamp.

Example

```
$redis->lastSave();
```

type

Description

Returns the type of data pointed by a given key.

KEYVALUE

Parameters

Key: key

Return value

Depending on the type of the data pointed by the key, this method will return the following value:

string: Redis::REDIS_STRING

set: Redis::REDIS_SET

list: Redis::REDIS_LIST

zset: Redis::REDIS_ZSET

hash: Redis::REDIS_HASH

other: Redis::REDIS_NOT_FOUND

Example

```
$redis->type('key');
```

flushDB

Description

Removes all entries from the current database.

DB

Parameters

None.

Return value

BOOL: Always TRUE.

Example

```
$redis->flushDB();
```

flushAll

Description

Removes all entries from all databases.

DB

Parameters

None.

Return value

BOOL: Always TRUE.

Example

```
$redis->flushAll();
```

sort

Description

Parameters

Key: key *Options:* array(key => value, ...) - optional, with the following keys and values:

```
'by' => 'some_pattern_*',  
'limit' => array(0, 1),  
'get' => 'some_other_pattern_*' or an array of patterns,  
'sort' => 'asc' or 'desc',  
'alpha' => TRUE,  
'store' => 'external-key'
```

Return value

An array of values, or a number corresponding to the number of elements stored if that was used.

Example

```
$redis->delete('s');  
$redis->sadd('s', 5);  
$redis->sadd('s', 4);  
$redis->sadd('s', 2);  
$redis->sadd('s', 1);  
$redis->sadd('s', 3);  
  
var_dump($redis->sort('s')); // 1,2,3,4,5  
var_dump($redis->sort('s', array('sort' => 'desc'))); // 5,4,3,2,1  
var_dump($redis->sort('s', array('sort' => 'desc', 'store' => 'out'))); // (int)5
```

info

Description

Returns an associative array from REDIS that provides information about the server. Passing no arguments to INFO will call the standard REDIS INFO command, which returns information such as the following:

redisPHPINFO();

- redis_version
- arch_bits
- uptime_in_seconds
- uptime_in_days
- connected_clients
- connected_slaves
- used_memory
- changes_since_last_save
- bgsave_in_progress
- last_save_time
- total_connections_received
- total_commands_processed
- role

You can pass a variety of options to INFO (per the Redis documentation), which will modify what is returned.

Parameters

option: The option to provide redis (e.g. "COMMANDSTATS", "CPU")

Example

```
$redis->info(); /* standard redis INFO command */  
$redis->info("COMMANDSTATS"); /* Information on the commands that have  
$redis->info("CPU"); /* just CPU information from Redis INFO */
```

resetStat

Description

Resets the statistics reported by Redis using the INFO command (info() function).

info()

These are the counters that are reset:

- Keyspace hits
- Keyspace misses
- Number of commands processed
- Number of connections received
- Number of expired keys

Parameters

None.

Return value

BOOL: TRUE in case of success, FALSE in case of failure.

Example

```
$redis->resetStat();
```

ttl, pttl

Description

Returns the time to live left for a given key, in seconds. If the key doesn't exist, FALSE is returned. pttl returns a time in milliseconds.

KEYKEYFLASEPTTL

Parameters

Key: key

Return value

Long, the time left to live in seconds.

Example

```
$redis->ttl('key');
```

persist

Description

Remove the expiration timer from a key.

KEY

Parameters

Key: key

Return value

BOOL: TRUE if a timeout was removed, FALSE if the key didn't exist or didn't have an expiration timer.

TRUEKEYKEYFLASE

Example

```
$redis->persist('key');
```

config

Description

Get or Set the redis config keys.

REIDSKEYS

Parameters

operation (string) either GET or SET

key string for SET, glob-pattern for GET.

See <http://redis.io/commands/config-get> for examples.

value optional string (only for SET)

Return value

Associative array for GET, key -> value

bool for SET

Examples

```
$redis->config("GET", "*max-*-entries*");  
$redis->config("SET", "dir", "/var/run/redis/dumps/");
```

eval

Description

Evaluate a LUA script serverside

LUA

Parameters

script string.

args array, optional.

num_keys int, optional.

Return value

Mixed. What is returned depends on what the LUA script itself returns, which could be a scalar value (int/string), or an array. Arrays that are returned can also contain other arrays, if that's how it was set up in your LUA script. If there is an error executing the LUA script, the `getLastError()` function can tell you the message that came back from Redis (e.g. compile error).

LUALUALUAgetLastError()REDIS

Examples

```
$redis->eval("return 1"); // Returns an integer: 1
$redis->eval("return {1,2,3}"); // Returns Array(1,2,3)
$redis->del('mylist');
$redis->rpush('mylist','a');
$redis->rpush('mylist','b');
$redis->rpush('mylist','c');
// Nested response: Array(1,2,3,Array('a','b','c'));
$redis->eval("return {1,2,3,redis.call('lrange','mylist',0,-1)}");
```


evalSha

Description

Evaluate a LUA script serverside, from the SHA1 hash of the script instead of the script itself. In order to run this command Redis will have to have already loaded the script, either by running it or via the SCRIPT LOAD command.

Parameters

script_sha string. The sha1 encoded hash of the script you want to run.

args array, optional. Arguments to pass to the LUA script.

num_keys int, optional. The number of arguments that should go into the KEYS array, vs. the ARGV array when Redis spins the script

Return value

Mixed. See EVAL

Examples

```
$script = 'return 1';  
$sha = $redis->script('load', $script);  
$redis->evalSha($sha); // Returns 1
```


script

Description

Execute the Redis SCRIPT command to perform various operations on the scripting subsystem.

Redis

Usage

```
$redis->script('load', $script);  
$redis->script('flush');  
$redis->script('kill');  
$redis->script('exists', $script1, [$script2, $script3, ...]);
```

Return value

- SCRIPT LOAD will return the SHA1 hash of the passed script on success, and FALSE on failure.
- SCRIPT FLUSH should always return TRUE
- SCRIPT KILL will return true if a script was able to be killed and false if not
- SCRIPT EXISTS will return an array with TRUE or FALSE for each passed script

getLastError

Description

The last error message (if any) returned from a SCRIPT call

Parameters

none

Return Value

A string with the last returned script based error message, or NULL if there is no error

Examples

```
$redis->eval('this-is-not-lua');  
$err = $redis->getLastError();  
// "ERR Error compiling script (new function): user_script:1: '=' expected near '-"
```

`_prefix`

Description

A utility method to prefix the value with the prefix setting for phpredis.

VALUE

Parameters

value string. The value you wish to prefix

Return value

If a prefix is set up, the value now prefixed. If there is no prefix, the value will be returned unchanged.

Examples

```
$redis->setOpt(Redis::OPT_PREFIX, 'my-prefix:');  
$redis->_prefix('my-value'); // Will return 'my-prefix:my-value'
```

`_unserialize`

Description

A utility method to unserialize data with whatever serializer is set up. If there is no serializer set, the value will be returned unchanged. If there is a serializer set up, and the data passed in is malformed, an exception will be thrown. This can be useful if phpredis is serializing values, and you return something from redis in a LUA script that is serialized.

SETSETSETPHP-REDIS

Parameters

value string. The value to be unserialized

Examples

```
$redis->setOpt(Redis::OPT_SERIALIZER, Redis::SERIALIZER_PHP);  
$redis->_unserialize('a:3:{i:0;i:1;i:1;i:2;i:2;i:3;}'); // Will return Array(1,2,3)
```

dump

Description

Dump a key out of a redis database, the value of which can later be passed into redis using the RESTORE command. The data that comes out of DUMP is a binary representation of the key as Redis stores it.

**KEYREIDS(),VALUERESTOREDUMP
VALUEREIDS**

Parameters

key string

Return value

The Redis encoded value of the key, or FALSE if the key doesn't exist

Examples

```
$redis->set('foo', 'bar');  
$val = $redis->dump('foo'); // $val will be the Redis encoded key value
```

restore

Description

Restore a key from the result of a DUMP operation.

DUMPVALUEKEY

Parameters

key string. The key name

tvl integer. How long the key should live (if zero, no expire will be set on the key)

value string (binary). The Redis encoded key value (from DUMP)

Examples

```
$redis->set('foo', 'bar');  
$val = $redis->dump('foo');  
$redis->restore('bar', 0, $val); // The key 'bar', will now be equal to the key 'foo'
```

migrate

Description

Migrates a key to a different Redis instance.

KEYREIDS

Parameters

host string. The destination host

port integer. The TCP port to connect to.

key string. The key to migrate.

destination-db integer. The target DB.

timeout integer. The maximum amount of time given to this transfer.

Examples

```
$redis->migrate('backup', 6379, 'foo', 0, 3600);
```

time

Description

Return the current Redis server time.

REDIS

Parameters

(none)

Return value

If successful, the time will come back as an associative array with element zero being the unix timestamp, and element one being microseconds.

Examples

```
$redis->time();
```


stringredisstringredisstringjpg

get

Description

Get the value related to the specified key

Parameters

key

Return Value

String or *Bool*: If key didn't exist, FALSE is returned. Otherwise, the value related to this key is returned.

BOOLKEYFALSEKEY

Examples

```
$redis->get('key');
```

set

Description

Set the string value in argument as value of the key.

KEY

Parameters

Key

Value

Timeout (optional). Calling SETEX is preferred if you want a timeout.

Return value

Bool TRUE if the command is successful.

Examples

```
$redis->set('key', 'value');
```

setex, psetex

Description

Set the string value in argument as value of the key, with a time to live. PSETEX uses a TTL in milliseconds.

KEY-VALUE,psetex()

Parameters

Key TTL Value

Return value

Bool TRUE if the command is successful.

Examples

```
$redis->setex('key', 3600, 'value'); // sets key → value, with 1h TTL.  
$redis->psetex('key', 100, 'value'); // sets key → value, with 0.1 sec TTL.
```

setnx

Description

Set the string value in argument as value of the key if the key doesn't already exist in the database.

setnxKEY-VALUERedisKEYSETFalse

Parameters

key value

Return value

Bool TRUE in case of success, FALSE in case of failure.

Examples

```
$redis->setnx('key', 'value'); /* return TRUE */  
$redis->setnx('key', 'value'); /* return FALSE */
```

del, delete

Description

Remove specified keys.

KEYS

Parameters

An array of keys, or an undefined number of parameters, each a key: *key1 key2 key3 ... keyN*

KEYSKEY

Return value

Long Number of keys deleted.

KEY-VALUE

Examples

```
$redis->set('key1', 'val1');  
$redis->set('key2', 'val2');  
$redis->set('key3', 'val3');  
$redis->set('key4', 'val4');  
  
$redis->delete('key1', 'key2'); /* return 2 */  
$redis->delete(array('key3', 'key4')); /* return 2 */
```

getSet

Description

Sets a value and returns the previous entry at that key.

VALUE KEYVALUE

Parameters

Key: key

STRING: value

Return value

A string, the previous value located at this key.

Example

```
$redis->set('x', '42');  
$exValue = $redis->getSet('x', 'lol'); // return '42', replaces x by 'lol'  
$newValue = $redis->get('x') // return 'lol'
```

multi, exec, discard.

Description

Enter and exit transactional mode.

Parameters

(optional) Redis::MULTI or Redis::PIPELINE. Defaults to Redis::MULTI. A Redis::MULTI block of commands runs as a single transaction; a Redis::PIPELINE block is simply transmitted faster to the server, but without any guarantee of atomicity. discard cancels a transaction.

`multi` Redis::MULTI Redis::PIPELINE Redis::MULTI

`Redis::MULTI` Redis::PIPELINE

`discard()`

Return value

multi() returns the Redis instance and enters multi-mode. Once in multi-mode, all subsequent method calls return the same object until exec() is called.

`multi()` Redis Redis exec()

Example

```
$ret = $redis->multi()
->set('key1', 'val1')
->get('key1')
->set('key2', 'val2')
->get('key2')
->exec();
```

```
/*
```

```
$ret == array(
```



```
0 => TRUE,  
1 => 'val1',  
2 => TRUE,  
3 => 'val2');  
*/
```

watch, unwatch

Description

Watches a key for modifications by another client. If the key is modified between WATCH and EXEC, the MULTI/EXEC transaction will fail (return FALSE). unwatch cancels all the watching of all keys by this client.

KEYKEYwatch()exec()exec()unwatch()KEYRedis

Parameters

keys: a list of keys

Example

```
$redis->watch('x');
/* long code here during the execution of which other clients could well modify
$ret = $redis->multi()
    ->incr('x')
    ->exec();
/*
$ret = FALSE if x has been modified between the call to WATCH and the call to
*/
```

subscribe

Description

Subscribe to channels. Warning: this function will probably change in the future.

,

Parameters

channels: an array of channels to subscribe to

callback: either a string or an array(\$instance, 'method_name').

The callback function receives 3 parameters: the redis instance, the channel name, and the message.

Example

```
function f($redis, $chan, $msg) {
    switch($chan) {
        case 'chan-1':
            ...
            break;

        case 'chan-2':
            ...
            break;

        case 'chan-2':
            ...
            break;
    }
}
```

```
$redis->subscribe(array('chan-1', 'chan-2', 'chan-3'), 'f'); // subscribe to 3 chans
```

publish

Description

Publish messages to channels. Warning: this function will probably change in the future.

Parameters

channel: a channel to publish to

message: string

Example

```
$redis->publish('chan-1', 'hello, world!'); // send message.
```

exists

Description

Verify if the specified key exists.

KEY

Parameters

key

Return value

BOOL: If the key exists, return TRUE, otherwise return FALSE.

Examples

```
$redis->set('key', 'value');  
$redis->exists('key'); /* TRUE */  
$redis->exists('NonExistingKey'); /* FALSE */
```

incr, incrBy

Description

Increment the number stored at key by one. If the second argument is filled, it will be used as the integer value of the increment.

KEY1KEY

Parameters

key

value: value that will be added to key (only for incrBy)

Return value

INT the new value

INT

Examples

```
$redis->incr('key1'); /* key1 didn't exists, set to 0 before the increment */  
/* and now has the value 1 */
```

```
$redis->incr('key1'); /* 2 */
```

```
$redis->incr('key1'); /* 3 */
```

```
$redis->incr('key1'); /* 4 */
```

```
$redis->incrBy('key1', 10); /* 14 */
```

incrByFloat

Description

Increment the key with floating point precision.

Parameters

key

value: (float) value that will be added to the key

Return value

FLOAT the new value

Examples

```
$redis->incrByFloat('key1', 1.5); /* key1 didn't exist, so it will now be 1.5 */
```

```
$redis->incrByFloat('key1', 1.5); /* 3 */
```

```
$redis->incrByFloat('key1', -1.5); /* 1.5 */
```

```
$redis->incrByFloat('key1', 2.5); /* 3.5 */
```

decr, decrBy

Description

Decrement the number stored at key by one. If the second argument is filled, it will be used as the integer value of the decrement.

KEY1KEY

Parameters

key

value: value that will be subtracted to key (only for decrBy)

Return value

INT the new value

Examples

```
$redis->decr('key1'); /* key1 didn't exists, set to 0 before the increment */  
                /* and now has the value -1 */  
  
$redis->decr('key1'); /* -2 */  
$redis->decr('key1'); /* -3 */  
$redis->decrBy('key1', 10); /* -13 */
```


mGet, getMultiple

Description

Get the values of all the specified keys. If one or more keys don't exist, the array will contain FALSE at the position of the key.

KEYS KEYS ARRAY KEYS FALSE

Parameters

Array: Array containing the list of the keys

:KEYS

Return value

Array: Array containing the values related to keys in argument

KEYS

Examples

```
$redis->set('key1', 'value1');  
$redis->set('key2', 'value2');  
$redis->set('key3', 'value3');  
$redis->mGet(array('key1', 'key2', 'key3')); /* array('value1', 'value2', 'value3');  
$redis->mGet(array('key0', 'key1', 'key5')); /* array(`FALSE`, 'value2', `FALSE`)
```

append

Description

Append specified string to the string stored in specified key.

KEY

Parameters

Key Value

Return value

INTEGER: Size of the value after the append

KEYSIZE

Example

```
$redis->set('key', 'value1');  
$redis->append('key', 'value2'); /* 12 */  
$redis->get('key'); /* 'value1value2' */
```

getRange (substr also supported but deprecated in redis)

Description

Return a substring of a larger string

Parameters

key start end

Return value

STRING: the substring

Example

```
$redis->set('key', 'string value');  
$redis->getRange('key', 0, 5); /* 'string' */  
$redis->getRange('key', -5, -1); /* 'value' */
```

setRange

Description

Changes a substring of a larger string.

Parameters

key

offset

value

Return value

STRING: the length of the string after it was modified.

Example

```
$redis->set('key', 'Hello world');  
$redis->setRange('key', 6, "redis"); /* returns 11 */  
$redis->get('key'); /* "Hello redis" */
```

strlen

Description

Get the length of a string value.

Parameters

key

Return value

INTEGER

Example

```
$redis->set('key', 'value');  
$redis->strlen('key'); /* 5 */
```

getBit

Description

Return a single bit out of a larger string

Parameters

keyoffset

Return value

LONG: the bit value (0 or 1)

Example

```
$redis->set('key', "\x7f"); // this is 0111 1111  
$redis->getBit('key', 0); /* 0 */  
$redis->getBit('key', 1); /* 1 */
```

setBit

Description

Changes a single bit of a string.

Parameters

keyoffset

value: bool or int (1 or 0)

Return value

LONG: 0 or 1, the value of the bit before it was set.

Example

```
$redis->set('key', "*"); // ord("*") = 42 = 0x2f = "0010 1010"  
$redis->setBit('key', 5, 1); /* returns 0 */  
$redis->setBit('key', 7, 1); /* returns 0 */  
$redis->get('key'); /* chr(0x2f) = "/" = b("0010 1111") */
```

bitop

Description

Bitwise operation on multiple keys.

Parameters

operation: either "AND", "OR", "NOT", "XOR"

ret_key: return key

key1

key2...

Return value

LONG: The size of the string stored in the destination key.

bitcount

Description

Count bits in a string.

Parameters

key

Return value

LONG: The number of bits set to 1 in the value behind the input key.

mset, msetnx

Description

Sets multiple key-value pairs in one atomic command. MSETNX only returns TRUE if all the keys were set (see SETNX).

KEY-VALUEKEYSKEY-VALUESET,msetnxTUREKEY

Parameters

Pairs: array(key => value, ...)

Return value

Bool TRUE in case of success, FALSE in case of failure.

Example

```
$redis->mset(array('key0' => 'value0', 'key1' => 'value1'));  
var_dump($redis->get('key0'));  
var_dump($redis->get('key1'));
```

Output:

```
string(6) "value0"  
string(6) "value1"
```

redisliststring

lPush

Description

Adds the string value to the head (left) of the list. Creates the list if the key didn't exist. If the key exists and is not a list, FALSE is returned.

LISTKEYLISTKEYLISTFLASE

Parameters

key

value String, value to push in key

Return value

LONG The new length of the list in case of success, FALSE in case of Failure.

LISTADDFALSE

Examples

```
$redis->delete('key1');  
$redis->lPush('key1', 'C'); // returns 1  
$redis->lPush('key1', 'B'); // returns 2  
$redis->lPush('key1', 'A'); // returns 3  
/* key1 now points to the following list: [ 'A', 'B', 'C' ] */
```

rPush

Description

Adds the string value to the tail (right) of the list. Creates the list if the key didn't exist. If the key exists and is not a list, FALSE is returned.

LISTKEYLISTKEYLISTFLASE

Parameters

key

value String, value to push in key

Return value

LONG The new length of the list in case of success, FALSE in case of Failure.

LISTADDFALSE

Examples

```
$redis->delete('key1');  
$redis->rPush('key1', 'A'); // returns 1  
$redis->rPush('key1', 'B'); // returns 2  
$redis->rPush('key1', 'C'); // returns 3  
/* key1 now points to the following list: [ 'A', 'B', 'C' ] */
```

lPushx

Description

Adds the string value to the head (left) of the list if the list exists.

VALUELISTLIST

Parameters

key

value String, value to push in key

Return value

LONG The new length of the list in case of success, *FALSE* in case of Failure.

ADD **LISTFALSE**

Examples

```
$redis->delete('key1');  
$redis->lPushx('key1', 'A'); // returns 0  
$redis->lPush('key1', 'A'); // returns 1  
$redis->lPushx('key1', 'B'); // returns 2  
$redis->lPushx('key1', 'C'); // returns 3  
/* key1 now points to the following list: [ 'A', 'B', 'C' ] */
```

rPushx

Description

Adds the string value to the tail (right) of the list if the list exists. **FALSE** in case of Failure.

VALUELISTLIST

Parameters

key

value String, value to push in key

Return value

LONG The new length of the list in case of success, **FALSE** in case of Failure.

ADD **LISTFALSE**

Examples

```
$redis->delete('key1');  
$redis->rPushx('key1', 'A'); // returns 0  
$redis->rPush('key1', 'A'); // returns 1  
$redis->rPushx('key1', 'B'); // returns 2  
$redis->rPushx('key1', 'C'); // returns 3  
/* key1 now points to the following list: [ 'A', 'B', 'C' ] */
```

lPop

Description

Return and remove the first element of the list.

LISTVALUELISTVALUE

Parameters

key

Return value

STRING if command executed successfully *BOOL FALSE* in case of failure (empty list)

VALUETURELISTFLASE

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C'); /* key1 => [ 'A', 'B', 'C' ] */  
$redis->lPop('key1'); /* key1 => [ 'B', 'C' ] */
```


rPop

Description

Returns and removes the last element of the list.

LISTVALUELISTVALUE

Parameters

key

Return value

STRING if command executed successfully *BOOL FALSE* in case of failure (empty list)

VALUETURELISTFLASE

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C'); /* key1 => [ 'A', 'B', 'C' ] */  
$redis->rPop('key1'); /* key1 => [ 'A', 'B' ] */
```

blPop, brPop

Description

Is a blocking lPop(rPop) primitive. If at least one of the lists contains at least one element, the element will be popped from the head of the list and returned to the caller. If all the lists identified by the keys passed in arguments are empty, blPop will block during the specified timeout until an element is pushed to one of those lists. This element will be popped.

```
lpopblock LISTVAULE ARRAY 'listName' element'  
TIMEOUT LIST blPop brPop timeout blPop brPop  
TIMEOUT LIST PUSH TIMEOUT POP
```

Parameters

ARRAY Array containing the keys of the lists **INTEGER** Timeout
Or **STRING** Key1 **STRING** Key2 **STRING** Key3
... **STRING** KeyN **INTEGER** Timeout

Return value

ARRAY array('listName', 'element')

Example

```
/* Non blocking feature */  
$redis->lPush('key1', 'A');  
$redis->delete('key2');  
  
$redis->blPop('key1', 'key2', 10); /* array('key1', 'A') */  
/* OR */  
$redis->blPop(array('key1', 'key2'), 10); /* array('key1', 'A') */  
  
$redis->brPop('key1', 'key2', 10); /* array('key1', 'A') */  
/* OR */  
$redis->brPop(array('key1', 'key2'), 10); /* array('key1', 'A') */
```

```
/* Blocking feature */  
  
/* process 1 */  
$redis->delete('key1');  
$redis->blPop('key1', 10);  
/* blocking for 10 seconds */  
  
/* process 2 */  
$redis->lPush('key1', 'A');  
  
/* process 1 */  
/* array('key1', 'A') is returned*/
```

lSize

Description

Returns the size of a list identified by Key. If the list didn't exist or is empty, the command returns 0. If the data type identified by Key is not a list, the command return FALSE.

```
KEYKEYLISTLISTISIZE0KEYLISTFALSE      lSize  
===FLASE0
```

Parameters

Key

Return value

LONG The size of the list identified by Key exists.

```
KEYLISTKEY0
```

BOOL FALSE if the data type identified by Key is not list

```
KEYLISTFALSE
```

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C'); /* key1 => [ 'A', 'B', 'C' ] */  
$redis->lSize('key1');/* 3 */  
$redis->rPop('key1');  
$redis->lSize('key1');/* 2 */
```

lIndex, lGet

Description

Return the specified element of the list stored at the specified key. 0 the first element, 1 the second ... -1 the last element, -2 the penultimate ... Return FALSE in case of a bad index or a key that doesn't point to a list.

KEY LIST01-1-2FLASE

Parameters

key index

Return value

String the element at this index

Bool FALSE if the key identifies a non-string data type, or no value corresponds to this index in the list Key.

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C'); /* key1 => [ 'A', 'B', 'C' ] */  
$redis->lGet('key1', 0); /* 'A' */  
$redis->lGet('key1', -1); /* 'C' */  
$redis->lGet('key1', 10); /* `FALSE` */
```

ISet

Description

Set the list at index with the new value.

VAULE

Parameters

key index value

Return value

BOOL TRUE if the new value is setted. FALSE if the index is out of range, or data type identified by key is not a list.

TUREKEYLISTLISTflase

IsetUPDATEEDITINSERTADDLIST

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C'); /* key1 => [ 'A', 'B', 'C' ] */  
$redis->lGet('key1', 0); /* 'A' */  
$redis->lSet('key1', 0, 'X');  
$redis->lGet('key1', 0); /* 'X' */
```

IRange, IGetRange

Description

Returns the specified elements of the list stored at the specified key in the range [start, end]. start and stop are interpreted as indices: 0 the first element, 1 the second ... -1 the last element, -2 the penultimate ...

Parameters

key start end

Return value

Array containing the values in specified range.

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C');  
$redis->lRange('key1', 0, -1); /* array('A', 'B', 'C') */
```

lTrim, listTrim

Description

Trims an existing list so that it will contain only a specified range of elements.

LISTLISTKEYLIST

JSPHP

Parameters

key start stop

Return value

Array

Bool return FALSE if the key identify a non-list value.

Example

```
$redis->rPush('key1', 'A');  
$redis->rPush('key1', 'B');  
$redis->rPush('key1', 'C');  
$redis->lRange('key1', 0, -1); /* array('A', 'B', 'C') */  
$redis->lTrim('key1', 0, 1);  
$redis->lRange('key1', 0, -1); /* array('A', 'B') */
```


IRem, IRemove

Description

Removes the first count occurrences of the value element from the list. If count is zero, all the matching elements are removed. If count is negative, elements are removed from tail to head.

**IRem,IRemovecountcount0count,count
count**

Note: The argument order is not the same as in the Redis documentation. This difference is kept for compatibility reasons.

Parameters

key

value

count

Return value

LONG the number of elements to remove

BOOL FALSE if the value identified by key is not a list.

Example

```
$redis->lPush('key1', 'A');  
$redis->lPush('key1', 'B');  
$redis->lPush('key1', 'C');  
$redis->lPush('key1', 'A');  
$redis->lPush('key1', 'A');
```

```
$redis->lRange('key1', 0, -1); /* array('A', 'A', 'C', 'B', 'A') */  
$redis->lRem('key1', 'A', 2); /* 2 */  
$redis->lRange('key1', 0, -1); /* array('C', 'B', 'A') */
```

lInsert

Description

Insert value in the list before or after the pivot value. the parameter options specify the position of the insert (before or after). If the list didn't exists, or the pivot didn't exists, the value is not inserted.

**LISTVALUEVALUELISTpivot(key position)
VALUE**

Parameters

key position Redis::BEFORE | Redis::AFTER *pivot value*

Return value

The number of the elements in the list, -1 if the pivot didn't exists.

Example

```
$redis->delete('key1');
$redis->lInsert('key1', Redis::AFTER, 'A', 'X'); /* 0 */

$redis->lPush('key1', 'A');
$redis->lPush('key1', 'B');
$redis->lPush('key1', 'C');

$redis->lInsert('key1', Redis::BEFORE, 'C', 'X'); /* 4 */
$redis->lRange('key1', 0, -1); /* array('A', 'B', 'X', 'C') */

$redis->lInsert('key1', Redis::AFTER, 'C', 'Y'); /* 5 */
$redis->lRange('key1', 0, -1); /* array('A', 'B', 'X', 'C', 'Y') */

$redis->lInsert('key1', Redis::AFTER, 'W', 'value'); /* -1 */
```

rpoplpush (redis >= 1.1)

Description

Pops a value from the tail of a list, and pushes it to the front of another list. Also return this value.

LISTLISTLIST

Parameters

Key: srckey

Key: dstkey

Return value

STRING The element that was moved in case of success, *FALSE* in case of failure.

Example

```
$redis->delete('x', 'y');

$redis->lPush('x', 'abc');
$redis->lPush('x', 'def');
$redis->lPush('y', '123');
$redis->lPush('y', '456');

// move the last of x to the front of y.
var_dump($redis->rpoplpush('x', 'y'));
var_dump($redis->lRange('x', 0, -1));
var_dump($redis->lRange('y', 0, -1));
```

Output:

```
string(3) "abc"
array(1) {
  [0]=>
  string(3) "def"
}
array(3) {
```

```
[0]=>  
string(3) "abc"  
[1]=>  
string(3) "456"  
[2]=>  
string(3) "123"  
}
```

brpoplpush

Description

A blocking version of rpoplpush, with an integral timeout in the third parameter.

rpoplpushLISTtimeout

Parameters

Key: srckey

Key: dstkey

Long: timeout

Return value

STRING The element that was moved in case of success, FALSE in case of timeout.

RedisSetListO(1)Set
4294967295

List **Set**C++setSetListSetSets
unionsintersectionsifferencesIO

sAdd

Description

Adds a value to the set value stored at key. If this value is already in the set, FALSE is returned.

VALUESETVALUESETFLASE

Parameters

key value

Return value

BOOL TRUE if value didn't exist and was added successfully, FALSE if the value is already present.

VALUESETADDEDTRUEFALSE

Example

```
$redis->sAdd('key1' , 'member1'); /* TRUE, 'key1' => {'member1'} */  
$redis->sAdd('key1' , 'member2'); /* TRUE, 'key1' => {'member1', 'member2'}*  
$redis->sAdd('key1' , 'member2'); /* FALSE, 'key1' => {'member1', 'member2'}
```


sRem, sRemove

Description

Removes the specified member from the set value stored at key.

VALUESET

Parameters

key member

Return value

BOOL TRUE if the member was present in the set, FALSE if it didn't.

Example

```
$redis->sAdd('key1' , 'member1');  
$redis->sAdd('key1' , 'member2');  
$redis->sAdd('key1' , 'member3'); /* 'key1' => {'member1', 'member2', 'member3'} */  
$redis->sRem('key1', 'member2'); /* 'key1' => {'member1', 'member3'} */
```

sMove

Description

Moves the specified member from the set at srcKey to the set at dstKey.

MEMBERSETSET

Parameters

srcKey dstKey member

Return value

BOOL If the operation is successful, return TRUE. If the srcKey and/or dstKey didn't exist, and/or the member didn't exist in srcKey, FALSE is returned.

TRUESETSETMEMBERSETFLASE

Example

```
$redis->sAdd('key1' , 'member11');
$redis->sAdd('key1' , 'member12');
$redis->sAdd('key1' , 'member13'); /* 'key1' => {'member11', 'member12', 'member13'} */
$redis->sAdd('key2' , 'member21');
$redis->sAdd('key2' , 'member22'); /* 'key2' => {'member21', 'member22'} */
$redis->sMove('key1', 'key2', 'member13'); /* 'key1' => {'member11', 'member12'}
/* 'key2' => {'member21', 'member22', 'member13'} */
```

sIsMember, sContains

Description

Checks if value is a member of the set stored at the key key.

VALUESET

Parameters

key value

Return value

BOOL TRUE if value is a member of the set at key key, FALSE otherwise.

Example

```
$redis->sAdd('key1' , 'member1');  
$redis->sAdd('key1' , 'member2');  
$redis->sAdd('key1' , 'member3'); /* 'key1' => {'member1', 'member2', 'member3'} */  
  
$redis->sIsMember('key1', 'member1'); /* TRUE */  
$redis->sIsMember('key1', 'memberX'); /* FALSE */
```

sCard, sSize

Description

Returns the cardinality of the set identified by key.

SET

Parameters

key

Return value

LONG the cardinality of the set identified by key, 0 if the set doesn't exist.

Example

```
$redis->sAdd('key1' , 'member1');  
$redis->sAdd('key1' , 'member2');  
$redis->sAdd('key1' , 'member3'); /* 'key1' => {'member1', 'member2', 'member3'} */  
$redis->sCard('key1'); /* 3 */  
$redis->sCard('keyX'); /* 0 */
```

sPop

Description

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

SET

Parameters

key

Return value

String "popped" value

Bool FALSE if set identified by key is empty or doesn't exist.

Example

```
$redis->sAdd('key1' , 'member1');  
$redis->sAdd('key1' , 'member2');  
$redis->sAdd('key1' , 'member3'); /* 'key1' => {'member3', 'member1', 'member2'} */  
$redis->sPop('key1'); /* 'member1', 'key1' => {'member3', 'member2'} */  
$redis->sPop('key1'); /* 'member3', 'key1' => {'member2'} */
```

sRandMember

Description

Returns a random element from the set value at Key, without removing it.

SETSET

Parameters

key

Return value

String value from the set

Bool FALSE if set identified by key is empty or doesn't exist.

Example

```
$redis->sAdd('key1' , 'member1');  
$redis->sAdd('key1' , 'member2');  
$redis->sAdd('key1' , 'member3'); /* 'key1' => {'member3', 'member1', 'member2'}  
$redis->sRandMember('key1'); /* 'member1', 'key1' => {'member3', 'member1', 'member2'}  
$redis->sRandMember('key1'); /* 'member3', 'key1' => {'member3', 'member1', 'member2'}
```

sInter

Description

Returns the members of a set resulting from the intersection of all the sets held at the specified keys. If just a single key is specified, then this command produces the members of this set. If one of the keys is missing, FALSE is returned.

SETSSETSETFLASE

Parameters

key1, key2, keyN: keys identifying the different sets on which we will apply the intersection.

SET

Return value

Array, contain the result of the intersection between those keys. If the intersection between the different sets is empty, the return value will be empty array.

SETSET

Examples

```
$redis->sAdd('key1', 'val1');  
$redis->sAdd('key1', 'val2');  
$redis->sAdd('key1', 'val3');  
$redis->sAdd('key1', 'val4');  
  
$redis->sAdd('key2', 'val3');  
$redis->sAdd('key2', 'val4');  
  
$redis->sAdd('key3', 'val3');
```

```
$redis->sAdd('key3', 'val4');  
var_dump($redis->sInter('key1', 'key2', 'key3'));
```

Output:

```
array(2) {  
  [0]=>  
  string(4) "val4"  
  [1]=>  
  string(4) "val3"  
}
```


sInterStore

Description

Performs a sInter command and stores the result in a new set.

SET

Parameters

Key: dstkey, the key to store the diff into.

key SETKEY

Keys: key1, key2... keyN. key1..keyN are intersected as in sInter.

KEYS

Return value

INTEGER: The cardinality of the resulting set, or *FALSE* in case of a missing key.

Example

```
$redis->sAdd('key1', 'val1');
$redis->sAdd('key1', 'val2');
$redis->sAdd('key1', 'val3');
$redis->sAdd('key1', 'val4');

$redis->sAdd('key2', 'val3');
$redis->sAdd('key2', 'val4');

$redis->sAdd('key3', 'val3');
$redis->sAdd('key3', 'val4');

var_dump($redis->sInterStore('output', 'key1', 'key2', 'key3'));
var_dump($redis->sMembers('output'));
```

Output:

```
int(2)
```

```
array(2) {  
  [0]=>  
  string(4) "val4"  
  [1]=>  
  string(4) "val3"  
}
```

sUnion

Description

Performs the union between N sets and returns it.

NSET

Parameters

Keys: key1, key2, ... , keyN: Any number of keys corresponding to sets in redis.

Return value

Array of strings: The union of all these sets.

Example

```
$redis->delete('s0', 's1', 's2');  
  
$redis->sAdd('s0', '1');  
$redis->sAdd('s0', '2');  
$redis->sAdd('s1', '3');  
$redis->sAdd('s1', '1');  
$redis->sAdd('s2', '3');  
$redis->sAdd('s2', '4');  
  
var_dump($redis->sUnion('s0', 's1', 's2'));
```

Return value: all elements that are either in s0 or in s1 or in s2.

```
array(4) {  
  [0]=>  
  string(1) "3"  
  [1]=>  
  string(1) "4"
```

```
[2]=>  
string(1) "1"  
[3]=>  
string(1) "2"  
}
```

sUnionStore

Description

Performs the same action as sUnion, but stores the result in the first key

sUnion()

Parameters

Key: dstkey, the key to store the diff into.

SETKEY

Keys: key1, key2, ... , keyN: Any number of keys corresponding to sets in redis.

KEYS

Return value

INTEGER: The cardinality of the resulting set, or FALSE in case of a missing key.

Example

```
$redis->delete('s0', 's1', 's2');  
  
$redis->sAdd('s0', '1');  
$redis->sAdd('s0', '2');  
$redis->sAdd('s1', '3');  
$redis->sAdd('s1', '1');  
$redis->sAdd('s2', '3');  
$redis->sAdd('s2', '4');  
  
var_dump($redis->sUnionStore('dst', 's0', 's1', 's2'));
```

```
var_dump($redis->sMembers('dst'));
```

Return value: the number of elements that are either in s0 or in s1 or in s2.

```
int(4)
array(4) {
  [0]=>
  string(1) "3"
  [1]=>
  string(1) "4"
  [2]=>
  string(1) "1"
  [3]=>
  string(1) "2"
}
```

sDiff

Description

Performs the difference between N sets and returns it.

NSSETSETSET (Result = SET0 - (SET1 UNION SET2 UNION ...SET N))

Parameters

Keys: key1, key2, ... , keyN: Any number of keys corresponding to sets in redis.

Return value

Array of strings: The difference of the first set will all the others.

,SET(first set - (N sets))

SET

Example

```
$redis->delete('s0', 's1', 's2');

$redis->sAdd('s0', '1');
$redis->sAdd('s0', '2');
$redis->sAdd('s0', '3');
$redis->sAdd('s0', '4');

$redis->sAdd('s1', '1');
$redis->sAdd('s2', '3');

var_dump($redis->sDiff('s0', 's1', 's2'));
```

Return value: all elements of s0 that are neither in s1 nor in s2.

```
array(2) {
  [0]=>
  string(1) "4"
  [1]=>
```

```
string(1) "2"  
}
```


sDiffStore

Description

Performs the same action as sDiff, but stores the result in the first key

sDiffSETdstkey

Parameters

Key: dstkey, the key to store the diff into.

Key:SETKEY

Keys: key1, key2, ... , keyN: Any number of keys corresponding to sets in redis

SET

Return value

INTEGER: The cardinality of the resulting set, or FALSE in case of a missing key.

Example

```
$redis->delete('s0', 's1', 's2');  
  
$redis->sAdd('s0', '1');  
$redis->sAdd('s0', '2');  
$redis->sAdd('s0', '3');  
$redis->sAdd('s0', '4');  
  
$redis->sAdd('s1', '1');  
$redis->sAdd('s2', '3');
```

```
var_dump($redis->sDiffStore('dst', 's0', 's1', 's2'));  
var_dump($redis->sMembers('dst'));
```

Return value: the number of elements of s0 that are neither in s1 nor in s2.

```
int(2)  
array(2) {  
  [0]=>  
  string(1) "4"  
  [1]=>  
  string(1) "2"  
}
```

sMembers, sGetMembers

Description

Returns the contents of a set.

SET

Parameters

Key: key

Return value

An array of elements, the contents of the set.

Example

```
$redis->delete('s');  
$redis->sAdd('s', 'a');  
$redis->sAdd('s', 'b');  
$redis->sAdd('s', 'a');  
$redis->sAdd('s', 'c');  
var_dump($redis->sMembers('s'));
```

Output:

```
array(3) {  
  [0]=>  
  string(1) "c"  
  [1]=>  
  string(1) "a"  
  [2]=>  
  string(1) "b"  
}
```

The order is random and corresponds to redis' own internal representation of the set structure.

RedisSET

ZSET(stored set)

set double score skip list hash table skip list Score
sorted set

zAdd

Description

Adds the specified member with a given score to the sorted set stored at key.

score

score

Parameters

key

score : double

value: string

Return value

Long 1 if the element is added. 0 otherwise.

Example

```
$redis->zAdd('key', 1, 'val1');  
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 5, 'val5');  
$redis->zRange('key', 0, -1); // array(val0, val1, val5)
```

zRange

Description

Returns a range of elements from the ordered set stored at the specified key, with values in the range [start, end].

start and stop are interpreted as zero-based indices: 0 the first element, 1 the second ... -1 the last element, -2 the penultimate ...

,0,1-1,-2...

Parameters

key

start: long

end: long

withscores: bool = false

Return value

Array containing the values in specified range.

Example

```
$redis->zAdd('key1', 0, 'val0');  
$redis->zAdd('key1', 2, 'val2');  
$redis->zAdd('key1', 10, 'val10');  
$redis->zRange('key1', 0, -1); /* array('val0', 'val2', 'val10') */  
  
// with scores  
$redis->zRange('key1', 0, -1, true); /* array('val0' => 0, 'val2' => 2, 'val10' => 10)
```

zDelete, zRem

Description

Deletes a specified member from the ordered set.

Parameters

key

member

Return value

LONG 1 on success, 0 on failure.

Example

```
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 2, 'val2');  
$redis->zAdd('key', 10, 'val10');  
$redis->zDelete('key', 'val2');  
$redis->zRange('key', 0, -1); /* array('val0', 'val10') */
```

zRevRange

Description

Returns the elements of the sorted set stored at the specified key in the range [start, end] in reverse order. start and stop are interpreted as zero-based indices: 0 the first element, 1 the second ... -1 the last element, -2 the penultimate ...

keyscoreZRRANGE

Parameters

key

start: long

end: long

withscores: bool = false

Return value

Array containing the values in specified range.

Example

```
$redis->zAdd('key', 0, 'val0');
$redis->zAdd('key', 2, 'val2');
$redis->zAdd('key', 10, 'val10');
$redis->zRevRange('key', 0, -1); /* array('val10', 'val2', 'val0') */

// with scores
$redis->zRevRange('key', 0, -1, true); /* array('val10' => 10, 'val2' => 2, 'val0' =
```


zRangeByScore, zRevRangeByScore

Description

Returns the elements of the sorted set stored at the specified key which have scores in the range [start,end]. Adding a parenthesis before start or end excludes it from the range. +inf and -inf are also valid limits. zRevRangeByScore returns the same items in reverse order, when the start and end parameters are swapped.

key score min max score min max score score

LIMITO(N)WITHSCORES score Redis

2.0

Parameters

key

start: string

end: string

options: array

Two options are available: withscores => TRUE, and limit => array(\$offset, \$count)

Return value

Array containing the values in specified range.

Example

```
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 2, 'val2');  
$redis->zAdd('key', 10, 'val10');  
$redis->zRangeByScore('key', 0, 3); /* array('val0', 'val2') */
```

```
$redis->zRangeByScore('key', 0, 3, array('withscores' => TRUE)); /* array('val0'  
$redis->zRangeByScore('key', 0, 3, array('limit' => array(1, 1))); /* array('val2' =  
$redis->zRangeByScore('key', 0, 3, array('limit' => array(1, 1))); /* array('val2') :  
$redis->zRangeByScore('key', 0, 3, array('withscores' => TRUE, 'limit' => array
```

zCount

Description

Returns the *number* of elements of the sorted set stored at the specified key which have scores in the range [start,end]. Adding a parenthesis before start or end excludes it from the range. +inf and -inf are also valid limits.

keyminmax

Parameters

key

start: string

end: string

Return value

LONG the size of a corresponding zRangeByScore.

Example

```
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 2, 'val2');  
$redis->zAdd('key', 10, 'val10');  
$redis->zCount('key', 0, 3); /* 2, corresponding to array('val0', 'val2') */
```

zRemRangeByScore, zDeleteRangeByScore

Description

Deletes the elements of the sorted set stored at the specified key which have scores in the range [start,end].

keys **score** **min** **max** **2.1.6** **min** **max**
ZRANGEBYSCORE

Parameters

key

start: double or "+inf" or "-inf" string

end: double or "+inf" or "-inf" string

Return value

LONG The number of values deleted from the sorted set

Example

```
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 2, 'val2');  
$redis->zAdd('key', 10, 'val10');  
$redis->zRemRangeByScore('key', 0, 3); /* 2 */
```

zRemRangeByRank, zDeleteRangeByRank

Description

Deletes the elements of the sorted set stored at the specified key which have rank in the range [start,end].

**keyrankstartstopstartstop0score-1
score-2score**

Parameters

key

start: LONG

end: LONG

Return value

LONG The number of values deleted from the sorted set

Example

```
$redis->zAdd('key', 1, 'one');  
$redis->zAdd('key', 2, 'two');  
$redis->zAdd('key', 3, 'three');  
$redis->zRemRangeByRank('key', 0, 1); /* 2 */  
$redis->zRange('key', 0, -1, array('withscores' => TRUE)); /* array('three' => 3)
```

zSize, zCard

Description

Returns the cardinality of an ordered set.

key

Parameters

key

Return value

Long, the set's cardinality

Example

```
$redis->zAdd('key', 0, 'val0');  
$redis->zAdd('key', 2, 'val2');  
$redis->zAdd('key', 10, 'val10');  
$redis->zSize('key'); /* 3 */
```

zScore

Description

Returns the score of a given member in the specified sorted set.

keymemberscoremembernil

Parameters

key

member

Return value

Double

Example

```
$redis->zAdd('key', 2.5, 'val2');  
$redis->zScore('key', 'val2'); /* 2.5 */
```

zRank, zRevRank

Description

Returns the rank of a given member in the specified sorted set, starting at 0 for the item with the smallest score. zRevRank starts at 0 for the item with the *largest* score.

keymemberscorerankindex0scorerank0
ZREVRANKrankindex

Parameters

key

member

Return value

Long, the item's score.

Example

```
$redis->delete('z');  
$redis->zAdd('key', 1, 'one');  
$redis->zAdd('key', 2, 'two');  
$redis->zRank('key', 'one'); /* 0 */  
$redis->zRank('key', 'two'); /* 1 */  
$redis->zRevRank('key', 'one'); /* 1 */  
$redis->zRevRank('key', 'two'); /* 0 */
```


zIncrBy

Description

Increments the score of a member from a sorted set by a given amount.

key
members
score
increment
member
key
score
score

Parameters

key

value: (double) value that will be added to the member's score

member

Return value

DOUBLE the new value

Examples

```
$redis->delete('key');  
$redis->zIncrBy('key', 2.5, 'member1'); /* key or member1 didn't exist, so mem  
/* and now has the value 2.5 */  
$redis->zIncrBy('key', 1, 'member1'); /* 3.5 */
```

zUnion

Description

Creates an union of sorted sets given in second argument. The result of the union will be stored in the sorted set defined by the first argument. The third optionnel argument defines weights to apply to the sorted sets in input. In this case, the weights will be multiplied by the score of each element in the sorted set before applying the aggregation. The forth argument defines the AGGREGATE option which specify how the results of the union are aggregated.

**keynumkeysdestinationkeyskeysscore
scoreWEIGHTSscoreWEIGHTS1
AGGREGATESUMscoreMINMAX
scoredestination**

Parameters

keyOutput

arrayZSetKeys

arrayWeights

aggregateFunction Either "SUM", "MIN", or "MAX": defines the behaviour to use on duplicate entries during the zUnion.

Return value

LONG The number of values in the new sorted set.

Example

```
$redis->delete('k1');
$redis->delete('k2');
$redis->delete('k3');
$redis->delete('ko1');
$redis->delete('ko2');
$redis->delete('ko3');

$redis->zAdd('k1', 0, 'val0');
$redis->zAdd('k1', 1, 'val1');

$redis->zAdd('k2', 2, 'val2');
$redis->zAdd('k2', 3, 'val3');

$redis->zUnion('ko1', array('k1', 'k2')); /* 4, 'ko1' => array('val0', 'val1', 'val2', 'v

/* Weighted zUnion */
$redis->zUnion('ko2', array('k1', 'k2'), array(1, 1)); /* 4, 'ko1' => array('val0', 'va
$redis->zUnion('ko3', array('k1', 'k2'), array(5, 1)); /* 4, 'ko1' => array('val0', 'va
```

zInter

Description

Creates an intersection of sorted sets given in second argument. The result of the union will be stored in the sorted set defined by the first argument. The third optionnel argument defines weights to apply to the sorted sets in input. In this case, the weights will be multiplied by the score of each element in the sorted set before applying the aggregation. The forth argument defines the AGGREGATE option which specify how the results of the union are aggregated.

numkeys **keys** **destination** **keys** **keys**
scores **score** **WEIGHTS** **AGGREGATE**
ZUNIONSTORE

Parameters

key *Output*

array *ZSet* *Keys*

array *Weights*

aggregateFunction Either "SUM", "MIN", or "MAX": defines the behaviour to use on duplicate entries during the zInter.

Return value

LONG The number of values in the new sorted set.

Example

```
$redis->delete('k1');  
$redis->delete('k2');
```

```
$redis->delete('k3');

$redis->delete('ko1');
$redis->delete('ko2');
$redis->delete('ko3');
$redis->delete('ko4');

$redis->zAdd('k1', 0, 'val0');
$redis->zAdd('k1', 1, 'val1');
$redis->zAdd('k1', 3, 'val3');

$redis->zAdd('k2', 2, 'val1');
$redis->zAdd('k2', 3, 'val3');

$redis->zInter('ko1', array('k1', 'k2'));          /* 2, 'ko1' => array('val1', 'val3') */
$redis->zInter('ko2', array('k1', 'k2'), array(1, 1)); /* 2, 'ko2' => array('val1', 'val3') */

/* Weighted zInter */
$redis->zInter('ko3', array('k1', 'k2'), array(1, 5), 'min'); /* 2, 'ko3' => array('val1', 'val3') */
$redis->zInter('ko4', array('k1', 'k2'), array(1, 5), 'max'); /* 2, 'ko4' => array('val1', 'val3') */
```

redis hashstringfieldvalue.O(1)
hashhashzipmapsmall hashzipmaphash
tablezipmaphashhashzipmapO(n)field
zipmap,O(1)

hSet

Description

Adds a value to the hash stored at key. If this value is already in the hash, FALSE is returned.

VALUEHASHVALUEHASHFALSE

Parameters

key

hashKey

value

Return value

LONG 1 if value didn't exist and was added successfully, 0 if the value was already present and was replaced, FALSE if there was an error.

Example

```
$redis->delete('h')
$redis->hSet('h', 'key1', 'hello'); /* 1, 'key1' => 'hello' in the hash at "h" */
$redis->hGet('h', 'key1'); /* returns "hello" */

$redis->hSet('h', 'key1', 'plop'); /* 0, value was replaced. */
$redis->hGet('h', 'key1'); /* returns "plop" */
```

hSetNx

Description

Adds a value to the hash stored at key only if this field isn't already in the hash.

VALUEHASH STOREFIELD

Return value

BOOL TRUE if the field was set, FALSE if it was already present.

Example

```
$redis->delete('h')
$redis->hSetNx('h', 'key1', 'hello'); /* TRUE, 'key1' => 'hello' in the hash at "h" :
$redis->hSetNx('h', 'key1', 'world'); /* FALSE, 'key1' => 'hello' in the hash at "h'
```


hGet

Description

Gets a value from the hash stored at key. If the hash table doesn't exist, or the key doesn't exist, FALSE is returned.

HASHVALUEHASHKEYFLASE

Parameters

key

hashKey

Return value

STRING The value, if the command executed successfully *BOOL* FALSE in case of failure

hLen

Description

Returns the length of a hash, in number of items

HASH

Parameters

key

Return value

LONG the number of items in a hash, *FALSE* if the key doesn't exist or isn't a hash.

Example

```
$redis->delete('h')
$redis->hSet('h', 'key1', 'hello');
$redis->hSet('h', 'key2', 'plop');
$redis->hLen('h'); /* returns 2 */
```

hDel

Description

Removes a value from the hash stored at key. If the hash table doesn't exist, or the key doesn't exist, FALSE is returned.

Parameters

key

hashKey

Return value

BOOL TRUE in case of success, FALSE in case of failure

hKeys

Description

Returns the keys in a hash, as an array of strings.

HASHKEYS

Parameters

Key: key

Return value

An array of elements, the keys of the hash. This works like PHP's `array_keys()`.

Example

```
$redis->delete('h');  
$redis->hSet('h', 'a', 'x');  
$redis->hSet('h', 'b', 'y');  
$redis->hSet('h', 'c', 'z');  
$redis->hSet('h', 'd', 't');  
var_dump($redis->hKeys('h'));
```

Output:

```
array(4) {  
  [0]=>  
  string(1) "a"  
  [1]=>  
  string(1) "b"  
  [2]=>  
  string(1) "c"  
  [3]=>  
  string(1) "d"  
}
```

The order is random and corresponds to redis' own

internal representation of the set structure.

hVals

Description

Returns the values in a hash, as an array of strings.

HASHVALUE

Parameters

Key: key

Return value

An array of elements, the values of the hash. This works like PHP's `array_values()`.

Example

```
$redis->delete('h');  
$redis->hSet('h', 'a', 'x');  
$redis->hSet('h', 'b', 'y');  
$redis->hSet('h', 'c', 'z');  
$redis->hSet('h', 'd', 't');  
var_dump($redis->hVals('h'));
```

Output:

```
array(4) {  
  [0]=>  
  string(1) "x"  
  [1]=>  
  string(1) "y"  
  [2]=>  
  string(1) "z"  
  [3]=>  
  string(1) "t"  
}
```

The order is random and corresponds to redis' own

internal representation of the set structure.

hGetAll

Description

Returns the whole hash, as an array of strings indexed by strings.

HASHKEYVALUE

Parameters

Key: key

Return value

An array of elements, the contents of the hash.

Example

```
$redis->delete('h');  
$redis->hSet('h', 'a', 'x');  
$redis->hSet('h', 'b', 'y');  
$redis->hSet('h', 'c', 'z');  
$redis->hSet('h', 'd', 't');  
var_dump($redis->hGetAll('h'));
```

Output:

```
array(4) {  
  ["a"]=>  
  string(1) "x"  
  ["b"]=>  
  string(1) "y"  
  ["c"]=>  
  string(1) "z"  
  ["d"]=>  
  string(1) "t"  
}
```

The order is random and corresponds to redis' own

internal representation of the set structure.

hExists

Description

Verify if the specified member exists in a key.

HASHKEY-VALUE

Parameters

key

memberKey

Return value

BOOL: If the member exists in the hash table, return TRUE, otherwise return FALSE.

Examples

```
$redis->hSet('h', 'a', 'x');  
$redis->hExists('h', 'a'); /* TRUE */  
$redis->hExists('h', 'NonExistingKey'); /* FALSE */
```

hIncrBy

Description

Increments the value of a member from a hash by a given amount.

HASHKEYKEYVALUEVALUE

Parameters

key

member

value: (integer) value that will be added to the member's value

Return value

LONG the new value

Examples

```
$redis->delete('h');  
$redis->hIncrBy('h', 'x', 2); /* returns 2: h[x] = 2 now. */  
$redis->hIncrBy('h', 'x', 1); /* h[x] ← 2 + 1. Returns 3 */
```

hIncrByFloat

Description

Increments the value of a hash member by the provided float value

HASHKEYKEYVALUEVALUE

Parameters

key

member

value: (float) value that will be added to the member's value

Return value

FLOAT the new value

Examples

```
$redis->delete('h');  
$redis->hIncrByFloat('h','x', 1.5); /* returns 1.5: h[x] = 1.5 now */  
$redis->hIncrByFLoat('h', 'x', 1.5); /* returns 3.0: h[x] = 3.0 now */  
$redis->hIncrByFloat('h', 'x', -3.0); /* returns 0.0: h[x] = 0.0 now */
```

hMset

Description

Fills in a whole hash. Non-string values are converted to string, using the standard (string) cast. NULL values are stored as empty strings.

HASHVALUENULL

Parameters

key

members: key → value array

Return value

BOOL

Examples

```
$redis->delete('user:1');  
$redis->hMset('user:1', array('name' => 'Joe', 'salary' => 2000));  
$redis->hIncrBy('user:1', 'salary', 100); // Joe earns 100 more now.
```

hMGet

Description

Retrieve the values associated to the specified fields in the hash.

HASHVALUE

Parameters

key

memberKeys Array

Return value

Array An array of elements, the values of the specified fields in the hash, with the hash keys as array keys.

Examples

```
$redis->delete('h');  
$redis->hSet('h', 'field1', 'value1');  
$redis->hSet('h', 'field2', 'value2');  
$redis->hmGet('h', array('field1', 'field2')); /* returns array('field1' => 'value1', 'f
```