Methods

Redis::__construct

Description

Creates a Redis client

Example

$redis = new Redis();

connect, open

Description

Connects to a Redis instance.

Parameters

host: string
port: int

Return Value

BOOL: `TRUE` on success, `FALSE` on error.

Example

$redis->connect('127.0.0.1', 6379);

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.

Examples

$redis->get('key');

set

Description

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

Parameters

Key Value

Return value

Bool `TRUE` if the command is successful.

Examples

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

setex

Description

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

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.
```

setnx

Description

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

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 */
```

delete

Description

Remove specified keys.

Parameters

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

Return value

Long Number of keys deleted.

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 */
```

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.

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.

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.

Parameters

keys: a list of keys

Example

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

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
messsage: string

Example

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

exists

Description

Verify if the specified key exists.

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

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.

Parameters

key value: value that will be added to key

Return value

INT the new value

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 */
```

decr

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.

Parameters

key value: value that will be substracted to key

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 */
```

getMultiple

Description

Get the values of all the specified keys. If one or more keys dont exist, the array will contain `FALSE` at the position of the key.

Parameters

Array: Array containing the list of the keys

Return value

Array: Array containing the values related to keys in argument

Examples

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

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.

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.

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.

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.

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' ] */
```

lPop

Description

Return and remove the first element of the list.

Parameters

key

Return value

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

Example

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

rPop

Description

Returns and removes the first element of the list.

Parameters

key

Return value

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

Example

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

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`.

Parameters

Key

Return value

LONG The size of the list identified by Key exists.
BOOL `FALSE` if the data type identified by Key is not list

Example

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

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.

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` */
```

lSet

Description

Set the list at index with the new value.

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.

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' */
```

lGetRange

Description

Returns the specified elements of the list stored at the specified key in the range [start, end]. start and stop are interpretated 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->lGetRange('key1', 0, -1); /* array('A', 'B', 'C') */
```

listTrim

Description

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

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

lRemove

Description

Removes the first `count` occurences 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.

Parameters

key count value

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

sAdd

Description

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

Parameters

key value

Return value

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

Example

```
$redis->sAdd('key1' , 'set1'); /* TRUE, 'key1' => {'set1'} */
$redis->sAdd('key1' , 'set2'); /* TRUE, 'key1' => {'set1', 'set2'}*/
$redis->sAdd('key1' , 'set2'); /* FALSE, 'key1' => {'set1', 'set2'}*/
```

sRemove

Description

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

Parameters

key member

Return value

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

Example

```
$redis->sAdd('key1' , 'set1');
$redis->sAdd('key1' , 'set2');
$redis->sAdd('key1' , 'set3'); /* 'key1' => {'set1', 'set2', 'set3'}*/
$redis->sRemove('key1', 'set2'); /* 'key1' => {'set1', 'set3'} */
```

sMove

Description

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

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.

Example

```
$redis->sAdd('key1' , 'set11');
$redis->sAdd('key1' , 'set12');
$redis->sAdd('key1' , 'set13'); /* 'key1' => {'set11', 'set12', 'set13'}*/
$redis->sAdd('key2' , 'set21');
$redis->sAdd('key2' , 'set22'); /* 'key2' => {'set21', 'set22'}*/
$redis->sMove('key1', 'key2', 'set13'); /* 'key1' =>  {'set11', 'set12'} */
                    /* 'key2' =>  {'set21', 'set22', 'set13'} */
```

sContains

Description

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

Parameters

key value

Return value

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

Example

```
$redis->sAdd('key1' , 'set1');
$redis->sAdd('key1' , 'set2');
$redis->sAdd('key1' , 'set3'); /* 'key1' => {'set1', 'set2', 'set3'}*/

$redis->sContains('key1', 'set1'); /* TRUE */
$redis->sContains('key1', 'setX'); /* FALSE */
```

sSize

Description

Returns the cardinality of the set identified by key.

Parameters

key

Return value

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

Example

```
$redis->sAdd('key1' , 'set1');
$redis->sAdd('key1' , 'set2');
$redis->sAdd('key1' , 'set3'); /* 'key1' => {'set1', 'set2', 'set3'}*/
$redis->sSize('key1'); /* 3 */
$redis->sSize('keyX'); /* 0 */
```

sPop

Description

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

Parameters

key

Return value

String "popped" value
Bool `FALSE` if set identified by key is empty or doesn't exist.

Example

```
$redis->sAdd('key1' , 'set1');
$redis->sAdd('key1' , 'set2');
$redis->sAdd('key1' , 'set3'); /* 'key1' => {'set3', 'set1', 'set2'}*/
$redis->sPop('key1'); /* 'set1', 'key1' => {'set3', 'set2'} */
$redis->sPop('key1'); /* 'set3', 'key1' => {'set2'} */
```

sRandMember

Description

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

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' , 'set1');
$redis->sAdd('key1' , 'set2');
$redis->sAdd('key1' , 'set3'); /* 'key1' => {'set3', 'set1', 'set2'}*/
$redis->sRandMember('key1'); /* 'set1', 'key1' => {'set3', 'set1', 'set2'} */
$redis->sRandMember('key1'); /* 'set3', 'key1' => {'set3', 'set1', 'set2'} */
```

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.

Parameters

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

Return value

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

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.

Parameters

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

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

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.

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

Parameters

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

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

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.

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.

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

Parameters

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

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

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.

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.

getSet

Description

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

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'
```

randomKey

Description

Returns a random key.

Parameters

None.

Return value

STRING: an existing key in redis.

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.

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.

Parameters

Key: key, the key to move.

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

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
```

renameKey

Description

Renames a key.

Parameters

STRING: srckey, the key to rename.

STRING: dstkey, the new name for the key.

Return value

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

Example

```
$redis->set('x', '42');
$redis->renameKey('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.

setTimeout, expire

Description

Sets an expiration date (a timeout) on an item.

Parameters

Key: key. The key that will disappear.

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

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

Description

Sets an expiration date (a timestamp) on an item.

Parameters

Key: key. The key that will disappear.

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

Return value

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

Example

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

getKeys

Description

Returns the keys that match a certain pattern.

Description

Parameters

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

Return value

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

Example

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

dbSize

Description

Returns the current database's size.

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.

Parameters

STRING: password

Return value

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

Example

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

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.

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
other: Redis::REDIS_NOT_FOUND

Example

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

flushDB

Description

Removes all entries from the current database.

Parameters

None.

Return value

BOOL: Always `TRUE`.

Example

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

flushAll

Description

Removes all entries from all databases.

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 of strings and integers, with the following keys:

- 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

Parameters

None.

Example

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

ttl

Description

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

Parameters

Key: key

Return value

Long, the time left to live in seconds.

Example

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

mset (redis >= 1.1)

Description

Sets multiple key-value pairs in one atomic command

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"
```

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.

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->lGetRange('x', 0, -1));
var_dump($redis->lGetRange('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"
}
```

zAdd

Description

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

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 ...

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, zRemove

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') */
```

zReverseRange

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 interpretated as zero-based indices: 0 the first element, 1 the second ... -1 the last element, -2 the penultimate ...

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->zReverseRange('key', 0, -1); /* array('val10', 'val2', 'val0') */

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

zRangeByScore

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.

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' => 0, 'val2' => 2) */
$redis->zRangeByScore('key', 0, 3, array('limit' => array(1, 1)); /* array('val2' => 2) */
$redis->zRangeByScore('key', 0, 3, array('limit' => array(1, 1)); /* array('val2') */
$redis->zRangeByScore('key', 0, 3, array('withscores' => TRUE, 'limit' => array(1, 1)); /* array('val2' =
> 2) */
```

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.

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') */
```

zDeleteRangeByScore, zRemoveRangeByScore

Description

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

Parameters

key
start: double
end: double

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->zDeleteRangeByScore('key', 0, 3); /* 2 */
```

zSize, zCard

Description

Returns the cardinality of an ordered set.

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.

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.

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.

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 member1's score is to 0 before
 the increment */
                     /* 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.

Parameters

keyOutput
arrayZSetKeys
arrayWeights
aggregateFunction

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', 'val3') */

/* Weighted zUnion */
$redis->zUnion('ko2', array('k1', 'k2'), array(1, 1)); /* 4, 'ko1' => array('val0', 'val1', 'val2', 'val3
') */
$redis->zUnion('ko3', array('k1', 'k2'), array(5, 1)); /* 4, 'ko1' => array('val0', 'val2', 'val3', 'val1
') */
```

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.

Parameters

keyOutput
arrayZSetKeys
arrayWeights
aggregateFunction

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('val3', 'val1') */
```

hSet

Description

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

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" */
```

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.

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

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.

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.

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.

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.

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.

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 */
```

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.

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.
```