

Keys		
DEL <i>key ...</i>	Delete item(s)	redis.Int
EXISTS <i>key</i>	Check for key	redis.Int
EXPIRE <i>key seconds</i>	Set timeout on item	redis.Int
EXPIREAT <i>key timestamp</i>	Set timeout by timestamp	redis.Int
PEXPIRE <i>key milliseconds</i>	Set timeout (ms)	redis.Int
PEXPIREAT <i>key milliseconds-timestamp</i>	Set timeout (ms timestamp)	redis.Int
PTTL <i>key</i>	Get item time to live (ms)	redis.Int
TTL <i>key</i>	Get item time to live	redis.Int

Strings		
GET <i>key</i>	Get value of key	redis.String
INCR <i>key</i>	Increment integer	redis.Int
INCRBY <i>key increment</i>	Increment integer	redis.Int
SET <i>key value</i> [EX <i>second</i>] [NX XX]	Set key	redis.String
SETEX <i>key seconds value</i>	Set with expiry (seconds)	redis.String
SETNX <i>key value</i>	Set if doesn't exist	redis.Int

Hashes		
HDEL <i>key field ...</i>	Delete item	redis.Int
HEXISTS <i>key field</i>	Check for item	redis.Int
HGET <i>key field</i>	Get item	redis.String
HGETALL <i>key</i>	Return all items	redis.Strings/ redis.StringMap
HINCRBY <i>key field increment</i>	Add to integer value	redis.Int
HINCRBYFLOAT <i>key field increment</i>	Add to float value	redis.Float64
HKEYS <i>key</i>	Return all keys	redis.Strings
HLEN <i>key</i>	Get number of items	redis.Int
HMGET <i>key field ...</i>	Get multiple items	redis.Strings
HMSET <i>key field value ...</i>	Set multiple items	redis.Strings/redis.StringMap
HSET <i>key value</i>	Set item	redis.Int
HSETNX <i>key value</i>	Set item if doesn't exist	redis.Int
HSCAN <i>key cursor</i> [MATCH <i>pattern</i>] [COUNT <i>count</i>]	Iterate items	redis.MultiBulk

Lists		
LINDEX <i>key index</i>	Access by index	redis.String
LINSERT <i>key BEFORE AFTER pivot value</i>	Insert next to	redis.Int
LLEN <i>key</i>	Get length	redis.Int
LPOP <i>key</i>	Pop from start	redis.String
LPUSH <i>key value ...</i>	Push onto start	redis.Int
LPUSHX <i>key value</i>	Push if list exists	redis.Int



Lists (cont)

LRange <i>key start stop</i>	Access range	redis.Strings
LRem <i>key count value</i>	Remove	redis.Int
LSet <i>key index value</i>	Set item by index	redis.String
LTrim <i>key start stop</i>	Trim to specified range	redis.String
RPop <i>key</i>	Pop from end	redis.String
RPush <i>key value ...</i>	Push onto end	redis.Int
RPushX <i>key value</i>	Push onto end if list exists	redis.Int

Sets

SAdd <i>key member ...</i>	Add item	redis.Int
SCard <i>key</i>	Get size	redis.Int
SIsMember <i>key member</i>	Check for item	redis.Int
SPOP <i>key</i>	Pop random item	redis.String
SRandMember <i>key count</i>	Get random item	redis.Strings
SRem <i>key member ...</i>	Remove matching	redis.Int
SScan <i>key cursor [MATCH pattern] [COUNT count]</i>	Iterate items	redis.MultiBulk

Sorted Sets

ZAdd <i>key score member ...</i>	Add item	redis.Int
ZCard <i>key</i>	Get number of items	redis.Int
ZCount <i>key min max</i>	Number of items within score range	redis.Int
ZIncrBy <i>key increment member</i>	Add to score	redis.String
ZRange <i>key start stop [WITHSCORES]</i>	Get items within rank range	redis.Strings/ redis.StringMap
ZRangeByScore <i>key min max [WITHSCORES] [LIMIT offset count]</i>	Get items within score range	redis.Strings/ redis.StringMap
ZRank <i>key member</i>	Get item rank	redis.Int
ZRem <i>key member ...</i>	Remove item(s)	redis.Int
ZRemRangeByRank <i>key start stop</i>	Remove items within rank range	redis.Int
ZRemRangeByScore <i>key min max</i>	Remove items within score range	redis.Int
ZRevRank <i>key member</i>	ZRange in reverse order	redis.Int
ZScore <i>key member</i>	Get item score	redis.String
ZScan <i>key cursor [MATCH pattern] [COUNT count]</i>	Iterate items	redis.MultiBulk

