Apusic 金蝶天燕

生成时间：2026-05-07

Development Manual

Kingdee Apusic In-Memory Data Cache V2.0.1

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 Overview

Apusic In-Memory Data Cache is a high-performance, highly available, and scalable distributed cache software independently developed by Kingdee. The AMDC data cache engine is compatible with the Redis protocol and covers all the functions of Redis. It not only provides a new option for enterprise data caching but also enables a simple, fast, and smooth replacement of Redis. While building its own cache framework, AMDC has spent a lot of effort to be compatible with various functions and interfaces of Redis, which means that no business code needs to be changed during the process of switching from Redis to AMDC in the system service. This implies that AMDC can be connected by Redis clients of all languages and is consistent with Redis in usage without additional costs.

2 Java, Python, Golang

We will use three commonly used languages to demonstrate our compatibility with Redis clients of various languages.

2.1 Java

- Single-node connection and usage

```
Jedis amdc = new Jedis("172.24.4.212", 6359);
System.out.println(amdc.ping());
System.out.println(amdc.set("key11", "value11"));
System.out.println(amdc.get("key11"));
```

- Sentinel mode connection and usage

```
Set<String> sentinels = new HashSet<String>(Arrays.asList(
    "172.24.6.110:27000",
    "172.24.6.110:27001",
    "172.24.6.110:27002"
));
JedisSentinelPool pool = new JedisSentinelPool("mymaster",
sentinels, jedisPoolConfig);
Jedis jedis = pool.getResource();
System.out.println(jedis.set("key12", "value12"));
System.out.println(jedis.get("key12"));
```

- Cluster mode connection and usage

```
Set<HostAndPort> clusterNodes = new HashSet<>();
clusterNodes.add(new HostAndPort("172.24.6.110", 7000));
clusterNodes.add(new HostAndPort("172.24.6.110", 7001));
clusterNodes.add(new HostAndPort("172.24.6.110", 7002));
clusterNodes.add(new HostAndPort("172.24.6.110", 7003));
clusterNodes.add(new HostAndPort("172.24.6.110", 7004));
```

```
clusterNodes.add(new HostAndPort("172.24.6.110", 7005));
JedisCluster ac = new JedisCluster(clusterNodes,
genericObjectPoolConfig);
System.out.println(ac.set("test1", "a"));
System.out.println(ac.set("1test", "b"));
System.out.println(ac.set("aaatest", "c"));
System.out.println("-------------------------");
System.out.println(ac.get("test1"));
System.out.println(ac.get("1test"));
System.out.println(ac.get("aaatest"));
```

2.1.1 Python

- Single-node connection and usage

```
amdc = redis.Redis(host=ip, port=port, db=0, decode_responses=True)
# Test the get set expire ttl commands
print(amdc.set("key1", "value1"))
print(amdc.get("key1"))
amdc.expire(name="key1", time=10)
print(amdc.ttl(name="key1"))
time.sleep(1)
print(amdc.ttl(name="key1"))
```

- Sentinel mode connection and usage

```
SENTINELADDRS = [{"172.21.33.69", "26359"}, {"172.21.33.69",
"26360"}, {"172.21.33.69", "26361"}]
def sentinel():
    amdc = redis.Sentinel(sentinels=SENTINELADDRS)
    # Get the master instance
    master = amdc.master_for("mymaster")
    # Get the slave instance
    slave = amdc.slave_for("mymaster")
    # Write in the master instance
    print(master.set("key2"))
```

```
    time.sleep(2)
    # Get from the slave instance
    print(slave.get("key2"))
```

- Cluster mode connection and usage

```
CLUSTERADDRS = [{"172.21.33.69", "6359"}, {"172.21.33.69", "6360"},
{"172.21.33.69", "6361"}]
def cluster():

amdc_cluster=RedisCluster(startup_nodes=CLUSTERADDRS,decode_responses=T
   print(amdc_cluster.set("key3","value3"))
   print(amdc_cluster.get("key3"))
```

2.1.2 Golang

- Single Node Connections and Usage

```
amdc:=redis.NewClient(&redis.Options{
  Addr: "172.21.33.69:6359",
})
if res,err:=amdc.Ping().Result();err!=nil{
  fmt.Println(err)
  return
}else{
  fmt.Println(res)
}
```

- Sentinel Mode Connection and Use

```
//Get the address of the master node provided by the Sentinel
amdc := redis.NewFailoverClient(&redis.FailoverOptions{
MasterName:    "mymaster",
  SentinelAddrs: []string{"172.21.33.69:26359",
"172.21.33.69:26360", "172.21.33.69:26361"},
```

```
})
defer amdc.Close()
res, err := amdc.Set("hahakey", "hahavalue", 0).Result()
if err != nil {
  fmt.Println(err)
  return
}
fmt.Println(res)
 //Connecting Sentinel Nodes
sentinel := redis.NewSentinelClient(&redis.Options{
  Addr: "172.21.33.69:26380",
})
defer sentinel.Close()
addr, err := sentinel.GetMasterAddrByName("mymaster").Result()
if err != nil {
  fmt.Println(err)
  return
}
fmt.Println(addr)
```

- Cluster Mode Connections and Usage

```
addrs:=[]string{"172.21.33.69:
6359","172.21.33.69:2360","172.21.33.69:6361"}
redis.NewClusterClient(&redis.ClusterOptions{
  Addrs: addrs,
})
if res,err:=amdc.Ping().Result();err!=nil{
  fmt.Println(err)
  return
}else{
  fmt.Println(res)
}
```

2.2 AMDC Commands

Command	Description
ACL LOAD	Reload ACLs from the configured ACL file
ACL SAVE	Save the current ACL rules in the configured ACL file
ACL LIST	List the current ACL rules in ACL configuration file format
ACL USERS	List the usernames of all configured ACL rules
ACL GETUSER username	Get the rules of a specific ACL user
ACL SETUSER username [rule [rule...]]	Modify or create the rules of a specific ACL user
ACL DELUSER username [username...]	Delete the specified ACL user and related rules
ACL CAT [categoryname]	List ACL categories or commands within the category
ACL GENPASS [bits]	Generate a pseudo-randomly secure password for an ACL user
ACL WHOAMI	Return the name of the user associated with the current connection
ACL LOG [count or RESET]	List the latest events that were denied due to ACL
ACL HELP	Display useful text about different subcommands
APPEND key value	Append a value to a key
ASKING	Sent by cluster clients after -ask redirection
AUTH [username] password	Authenticate to the server
BGREWRITEAOF	Rewrite the append-only file asynchronously
BGSAVE [SCHEDULE]	Save the dataset to disk asynchronously
BITCOUNT key [start end [BYTE|BIT]]	Count the set bits in a string
BITFIELD key [GET encoding offset] [SET encoding offset value] [INCRBY encoding offset increment] [OVERFLOW WRAP|SAT|FAIL]	Perform arbitrary bit field integer operations on a string

BITFIELD_RO key GET encoding offset	Perform arbitrary bit field integer operations on a string. Read-only variant of the restricted area
BITOP operation destkey key [key...]	Perform bitwise operations between strings
BITPOS key bit [start [end [BYTE|BIT]]]	Find the first bit set or cleared in a string
BLPOP key [key...] timeout	Remove and get the first element in a list, or block until available
BRPOP key [key...] timeout	Remove and get the last element in a list, or block until available
BRPOPLPUSH source destination timeout	Pop an element from a list, push it to another list and return; or block until one is available
BLMOVE source destination LEFT|RIGHT LEFT|RIGHT timeout	Pop an element from a list, push it to another list and return; or block until one is available
LMPOP numkeys key [key...] LEFT|RIGHT [COUNT count]	Pop elements from a list
BLMPOP timeout numkeys key [key...] LEFT|RIGHT [COUNT count]	Pop elements from a list, or block until available
BZPOPMIN key [key...] timeout	Remove and return the member with the lowest score from one or more sorted sets, or block until available
BZPOPMAX key [key...] timeout	Remove and return the member with the highest score from one or more sorted sets, or block until available
BZMPOP timeout numkeys key [key...] MIN|MAX [COUNT count]	Remove and return the member with a score in the sorted set or block until available
CLIENT CACHING YES|NO	Instruct the server to track or not keys in the next request
CLIENT ID	Return the client ID of the current connection
CLIENT INFO	Return information about the current client connection.
CLIENT KILL [ip:port] [ID client-id] [TYPE normal|master|slave|pubsub] [USER username] [ADDR ip:port] [LADDR ip:port] [SKIPME yes/no]	Kill the client's connection

CLIENT LIST [TYPE normal|master|replica|pubsub] [ID client-id [client-id...]]	Get the list of client connections
CLIENT GETNAME	Get the current connection name
CLIENT GETREDIR	Get the client ID of the tracking notification redirect (if any)
CLIENT UNPAUSE	Resume processing of a paused client
CLIENT PAUSE timeout [WRITE|ALL]	Stop processing commands for a client for a period of time
CLIENT REPLY ON|OFF|SKIP	Instruct the server whether to reply to commands
CLIENT SETNAME connection-name	Set the name of the current connection
CLIENT TRACKING ON|OFF [REDIRECT client-id] [PREFIX prefix [PREFIX prefix...]] [BCAST] [OPTIN] [OPTOUT] [NOLOOP]	Enable or disable server-assisted client caching support
CLIENT TRACKINGINFO	Return information about the server-assisted client caching of the current connection
CLIENT UNBLOCK client-id [TIMEOUT|ERROR]	Unblock a client that was blocked from a blocked command in a different connection
CLIENT NO-EVICT ON|OFF	Set the client eviction mode for the current connection
COMMAND	Get an array of detailed Redis command information
COMMAND COUNT	Get the total number of Redis commands
COMMAND GETKEYS	Extract keys given a full Redis command
COMMAND INFO command-name [command-name...]	Get an array of detailed information for specific Redis commands
CONFIG GET parameter [parameter...]	Get the value of a configuration parameter
CONFIG REWRITE	Rewrite the configuration file with the in-memory configuration
CONFIG SET parameter value [parameter value...]	Set a configuration parameter to the given value
CONFIG RESETSTAT	Reset the statistics returned by INFO

COPY source destination [DB destination-db] [REPLACE]	Copy a key
DBSIZE	Return the number of keys in the selected database
DEBUG OBJECT key	Get debug information about a key
DEBUG SEGFAULT	Crash the server
DECR key	Decrement the integer value of a key by one
DECRBY key decrement	Decrement the integer value of a key by the given number
DEL key [key...]	Delete a key
DISCARD	Abandon all commands issued after a MULTI
DUMP key	Return the serialized version of the value stored in the specified key.
ECHO message	Echo the given string
EVAL script numkeys [key [key...]] [arg [arg...]]	Execute a Lua script server-side
EVAL_RO script numkeys key [key...] arg [arg...]	Execute a read-only Lua script server-side
EVALSHA sha1 numkeys [key [key...]] [arg [arg...]]	Execute a Lua script server-side
EVALSHA_RO sha1 numkeys key [key...] arg [arg...]	Execute a read-only Lua script server-side
EXEC	Execute MULTI
EXISTS key [key...]	Determine if a key exists
EXPIRE key seconds [NX|XX|GT|LT]	Set the time-to-live of a key in seconds
EXPIREAT key timestamp [NX|XX|GT|LT]	Set the expiration time of a key as a UNIX timestamp
EXPIRETIME key	Get the expiration Unix timestamp of a key
FAILOVER [TO host port [FORCE]] [ABORT] [TIMEOUT milliseconds]	Initiate a coordinated failover between this server and one of its replicas.
FLUSHALL [ASYNC|SYNC]	Delete all keys from all databases
FLUSHDB [ASYNC|SYNC]	Delete all keys from the current database

GEOADD key [NX|XX] [CH] longitude latitude member [longitude latitude member...]	Add one or more geospatial items in a geospatial index represented using sorted sets
GEOHASH key member [member...]	Return members of a geospatial index as standard geohash strings
GEOPOS key member [member...]	Return the longitude and latitude of members of a geospatial index
GEODIST key member1 member2 [m|km|ft|mi]	Return the distance between two members of a geospatial index
GEORADIUS key longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count [ANY]] [ASC|DESC] [STORE key] [STOREDIST key]	Query the sorted set representing a geospatial index to get members that match a given maximum distance from a point
GEORADIUSBYMEMBER key member radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count [ANY]] [ASC|DESC] [STORE key] [STOREDIST key]	Query the sorted set representing a geospatial index to get members that match a given maximum distance from a member
GEOSEARCH key [FROMMEMBER member] [FROMLONLAT longitude latitude] [BYRADIUS radius m|km|ft|mi] [BYBOX width height m|km|ft|mi] [ASC|DESC] [COUNT count [ANY]] [WITHCOORD] [WITHDIST] [WITHHASH]	Query the sorted set representing a geospatial index to get members within a box or circular area.
GEOSEARCHSTORE destination source [FROMMEMBER member] [FROMLONLAT longitude latitude] [BYRADIUS radius m|km|ft|mi] [BYBOX width height m|km|ft|mi] [ASC|DESC] [COUNT count [ANY]] [STOREDIST]	Query the sorted set representing a geospatial index to get members within a box or circular area and store the result in another key.
GET key	Get the value of a key
GETBIT key offset	Return the bit value at the offset in the string value stored in key
GETDEL key	Get the value of a key and delete the key
GETRANGE key start end	Get the substring of the string stored in a key
GETSET key value	Set the string value of a key and return its old value
HDEL key field [field...]	Delete one or more hash fields
HELLO [protover [AUTH username password] [SETNAME clientname]]	Handshake with Redis

HEXISTS key field	Determine if a hash field exists
HGET key field	Get the value of a hash field
HGETALL key	Get all fields and values in a hash
HINCRBY key field increment	Increment the integer value of a hash field by the given number
HINCRBYFLOAT key field increment	Increment the floating-point value of a hash field by the given amount
HKEYS key	Get all fields in a hash
HLEN key	Get the number of fields in a hash
HMGET key field [field...]	Get the values of all given hash fields
HMSET key field value [field value...]	Set multiple hash fields to multiple values
HSET key field value [field value...]	Set the string value of a hash field
HSETNX key field value	Set the value of a hash field only if the field does not exist
HRANDFIELD key [count [WITHVALUES]]	Get one or more random fields from a hash
HSTRLEN key field	Get the length of the value of a hash field
HVALS key	Get all values in a hash
INCR key	Increment the integer value of a key by one
INCRBY key increment	Increment the integer value of a key by the given amount
INCRBYFLOAT key increment	Increment the floating-point value of a key by the given amount
INFO [section]	Get information and statistics about the server
LOLWUT [VERSION version]	Display some computer art and the Redis version
KEYS pattern	Find all keys that match the given pattern
LASTSAVE	Get the Unix timestamp of the last successful save to disk
LINDEX key index	Get an element from a list by index

LINSERT key BEFORE|AFTER pivot element	Insert an element before or after another element in a list
LLEN key	Get the length of a list
LPOP key [count]	Remove and get the first element in a list
LPOS key element [RANK rank] [COUNT num-matches] [MAXLEN len]	Return the index of the matching element in a list
LPUSH key element [element...]	Add one or more elements to a list
LPUSHX key element [element...]	Add an element to a list only if the list exists
LRANGE key start stop	Get a range of elements from a list
LREM key count element	Remove elements from a list
LSET key index element	Set the value of an element in a list by index
LTRIM key start stop	Trim a list to the specified range
MEMORY DOCTOR	Output a memory issue report
MEMORY HELP	Display useful text about different subcommands
MEMORY MALLOC-STATS	Display internal statistics of the allocator
MEMORY PURGE	Request the allocator to release memory
MEMORY STATS	Display memory usage details
MEMORY USAGE key [SAMPLES count]	Estimate the memory usage of a key
MGET key [key...]	Get the values of all given keys
MIGRATE host port key| destination-db timeout [COPY] [REPLACE] [AUTH password] [AUTH2 username password] [KEYS key [key...]]	Atomically transfer a key from one Redis instance to another.
MODULE LIST	List all modules loaded by the server
MODULE LOAD path [arg [arg...]]	Load a module
MODULE UNLOAD name	Unload a module
MONITOR	Listen to all requests received by the server in real-time
MOVE key db	Move a key to another database

MSET key value [key value...]	Set multiple keys to multiple values
MSETNX key value [key value...]	Set multiple keys to multiple values only if none of the keys exist
MULTI	Mark the beginning of a transaction block
OBJECT ENCODING key	Check the internal encoding of a Redis object
OBJECT FREQ key	Get the logarithmic access frequency counter of a Redis object
OBJECT IDLETIME key	Get the time since the last access of a Redis object
OBJECT REFCOUNT key	Get the reference count of a key value
OBJECT HELP	Display useful text about different subcommands
PERSIST key	Remove the expiration time from a key
PEXPIRE key milliseconds [NX|XX|GT|LT]	Set the time-to-live of a key in milliseconds
PEXPIREAT key milliseconds-timestamp [NX|XX|GT|LT]	Set the expiration time of a key as a Unix timestamp specified in milliseconds
PEXPIRETIME key	Get the expiration Unix timestamp of a key in milliseconds
PFADD key [element [element...]]	Add the specified elements to the specified HyperLogLog.
PFCOUNT key [key...]	Return the approximate cardinality of the set observed at the key(s) in HyperLogLog.
PFMERGE destkey sourcekey [sourcekey...]	Merge N different HyperLogLogs into one.
PING [message]	Ping the server
PSETEX key milliseconds value	Set the key value and expiration time in milliseconds
PSUBSCRIBE pattern [pattern...]	Listen for messages published to channels that match the given pattern
PUBSUB CHANNELS [pattern]	List active channels
PUBSUB NUMPAT	Get the count of unique pattern subscriptions
PUBSUB NUMSUB [channel [channel...]]	Get the number of subscribers for channels

PUBSUB HELP	Display useful text about different subcommands
PTTL key	Get the time-to-live of a key in milliseconds
PUBLISH channel message	Publish a message to a channel
PUNSUBSCRIBE [pattern [pattern...]]	Stop listening for messages published to channels that match the given pattern
QUIT	Close the connection
RANDOMKEY	Return a random key from the keyspace
READONLY	Enable read queries on cluster replica node connections
READWRITE	Disable read queries on cluster replica node connections
RENAME key newkey	Rename a key
RENAMENX key newkey	Rename a key only if the new key does not exist
RESET	Reset the connection
RESTORE key ttl serialized-value [REPLACE] [ABSTTL] [IDLETIME seconds] [FREQ frequency]	Create a key using the provided serialized value, previously obtained using DUMP.
ROLE	Return the role of the instance in the replication context
RPOP key [count]	Remove and get the last element in a list
RPOPLPUSH source destination	Remove the last element from a list, add it to another list and return it
LMOVE source destination LEFT|RIGHT LEFT|RIGHT	Pop an element from a list, push it to another list and return it
RPUSH key element [element...]	Append one or more elements to a list
RPUSHX key element [element...]	Append an element to a list only if the list exists
SADD key member [member...]	Add one or more members to a set
SAVE	Synchronously save the dataset to disk
SCARD key	Get the number of members in a set

SCRIPT DEBUG YES|SYNC|NO	Set the debug mode for executed scripts.
SCRIPT EXISTS sha1 [sha1...]	Check if a script exists in the script cache.
SCRIPT FLUSH [ASYNC|SYNC]	Remove all scripts from the script cache.
SCRIPT KILL	Terminate the currently executing script.
SCRIPT LOAD script	Load the specified Lua script into the script cache.
SDIFF key [key...]	Subtract multiple sets
SDIFFSTORE destination key [key...]	Subtract multiple sets and store the result set in a key
SELECT index	Change the database selected by the current connection
SET key value [EX seconds|PX milliseconds|EXAT timestamp|PXAT milliseconds-timestamp|KEEPTTL] [NX|XX] [GET]	Set the string value of a key
SETBIT key offset value	Set or clear the bit at the offset in the key
SETEX key seconds value	Set the value and expiration time of a key
SETNX key value	Set the value of a key only if the key does not exist
SETRANGE key offset value	Overwrite part of the string at the key starting from the specified offset
SHUTDOWN [NOSAVE|SAVE]	Synchronously save the dataset to disk and then shut down the server
SINTER key [key...]	Intersect multiple sets
SINTERCARD numkeys key [key...] [LIMIT limit]	Intersect multiple sets and return the cardinality of the result
SINTERSTORE destination key [key...]	Intersect multiple sets and store the result set in a key
SISMEMBER key member	Determine if the given value is a member of the set
SMISMEMBER key member [member...]	Return the membership associated with the given elements of the set

REPLICAOF host port	Make the server a replica of another instance or promote it to a master server.
SLOWLOG GET [count]	Get entries of the slow log
SLOWLOG LEN	Get the length of the slow log
SLOWLOG RESET	Clear all entries in the slow log
SLOWLOG HELP	Display useful text about different subcommands
SMEMBERS key	Get all members of a set
SMOVE source destination member	Move a member from one set to another
SORT key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern...]] [ASC|DESC] [ALPHA] [STORE destination]	Sort elements in a list, set or sorted set
SORT_RO key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern...]] [ASC|DESC] [ALPHA]	Sort elements in a list, set or sorted set. Read-only variant of SORT.
SPOP key [count]	Remove and return one or more random members from a set
SRANDMEMBER key [count]	Get one or more random members from a set
SREM key member [member...]	Remove one or more members from a set
LCS key1 key2 [LEN] [IDX] [MINMATCHLEN len] [WITHMATCHLEN]	Find the longest common substring
STRLEN key	Get the length of the value stored in the key
SUBSCRIBE channel [channel...]	Listen for messages published to the given channel
SUNION key [key...]	Add multiple sets
SUNIONSTORE destination key [key...]	Add multiple sets and store the result set in a key
SWAPDB index1 index2	Swap two Redis databases
SYNC	Internal command for replication
PSYNC replicationid offset	Internal command for replication
TIME	Return the current server time

TOUCH key [key...]	Change the last access time of the key. Return the number of specified existing keys.
TTL key	Get the time-to-live of a key in seconds
TYPE key	Determine the type stored at the key
UNSUBSCRIBE [channel [channel...]]	Stop listening for messages published to the given channel
UNLINK key [key...]	Asynchronously delete a key in another thread. Otherwise, it's like DEL but non-blocking.
UNWATCH	Forget all watched keys
WAIT numreplicas timeout	Wait for synchronous replication of all write commands sent in the current connection context
WATCH key [key...]	Observe the given keys to determine the execution of the MULTI/EXEC block
ZADD key [NX|XX] [GT|LT] [CH] [INCR] score member [score member...]	Add one or more members to the sorted set, or update its score if it already exists
ZCARD key	Get the number of members in the sorted set
ZCOUNT key min max	Count the members in the sorted set within the given score range
ZDIFF numkeys key [key...] [WITHSCORES]	Subtract multiple sorted sets
ZDIFFSTORE destination numkeys key [key...]	Subtract multiple sorted sets and store the result sorted set in a new key
ZINCRBY key increment member	Increase the score of a member in the sorted set
ZINTER numkeys key [key...] [WEIGHTS weight [weight...]] [AGGREGATE SUM|MIN|MAX] [WITHSCORES]	Intersect multiple sorted sets
ZINTERCARD numkeys key [key...] [LIMIT limit]	Intersect multiple sorted sets and return the cardinality of the result
ZINTERSTORE destination numkeys key [key...] [WEIGHTS weight [weight...]] [AGGREGATE SUM|MIN|MAX]	Intersect multiple sorted sets and store the result sorted set in a new key

ZLEXCOUNT key min max	Calculate the number of members in the sorted set within the given lexicographical range
ZPOPMAX key [count]	Delete and return the members with the highest scores in the sorted set
ZPOPMIN key [count]	Delete and return the members with the lowest scores in the sorted set
ZMPOP numkeys key [key...] MIN|MAX [COUNT count]	Delete and return the members with scores in the sorted set
ZRANDMEMBER key [count [WITHSCORES]]	Get one or more random elements from the sorted set
ZRANGESTORE dst src min max [BYSCORE|BYLEX] [REV] [LIMIT offset count]	Store a series of members from the sorted set to another key
ZRANGE key min max [BYSCORE|BYLEX] [REV] [LIMIT offset count] [WITHSCORES]	Return a series of members from the sorted set
ZRANGEBYLEX key min max [LIMIT offset count]	Return the member range in an ordered set, sorted lexicographically
ZREVRANGEBYLEX key max min [LIMIT offset count]	Return the member range in the sorted set, sorted lexicographically from high to low string sorting.
ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]	Return a series of members in the sorted set by score
ZRANK key member	Determine the index of a member in the sorted set
ZREM key member [member...]	Delete one or more members from the sorted set
ZREMRANGEBYLEX key min max	Delete all members in the sorted set between the given lexicographical ranges
ZREMRANGEBYRANK key start stop	Delete all members in the sorted set within the given index range
ZREMRANGEBYSCORE key min max	Delete all members in the sorted set within the given score range
ZREVRANGE key start stop [WITHSCORES]	Return a series of members in the sorted set by index, sorted by score from high to low

ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count]	Return a series of members in the sorted set by score, sorted by score from high to low
ZREVRANK key member	Determine the index of a certain member in an ordered set, sorted by score from high to low
ZSCORE key member	Get the score associated with the given member in the sorted set
ZUNION numkeys key [key...] [WEIGHTS weight [weight...]] [AGGREGATE SUM|MIN|MAX] [WITHSCORES]	Add multiple sorted sets
ZMSCORE key member [member...]	Get the scores associated with the given members in the sorted set
ZUNIONSTORE destination numkeys key [key...] [WEIGHTS weight [weight...]] [AGGREGATE SUM|MIN|MAX]	Add multiple sorted sets and store the result sorted set in a new key
SCAN cursor [MATCH pattern] [COUNT count] [TYPE type]	Incrementally iterate the key space
SSCAN key cursor [MATCH pattern] [COUNT count]	Incrementally iterate the Set elements
HSCAN key cursor [MATCH pattern] [COUNT count]	Incrementally iterate the hash fields and associated values
ZSCAN key cursor [MATCH pattern] [COUNT count]	Incrementally iterate the sorted set elements and associated scores
XINFO CONSUMERS key groupname	List the consumers in the consumer group
XINFO GROUPS key	List the consumer groups of the stream
XINFO STREAM key [FULL [COUNT count]]	Get information about the stream
XINFO HELP	Display useful text about different subcommands
XADD key [NOMKSTREAM] [MAXLEN|MINID [=|~] threshold [LIMIT count]] *|ID field value [field value...]	Append new entries to the stream
XTRIM key MAXLEN|MINID [=|~] threshold [LIMIT count]	Trim the stream to (approximately if '~' is passed) a specific size
XDEL key ID [ID...]	Delete the specified entries from the stream. Return the number of actually deleted items,

	which may be different from the number of IDs passed if some IDs do not exist.
XRANGE key start end [COUNT count]	Return a series of elements in the stream whose IDs match the specified ID range
XREVRANGE key end start [COUNT count]	In contrast to XRANGE, return a series of elements in the stream in the opposite order (from larger to smaller IDs) where the IDs match the specified ID range
XLEN key	Return the number of entries in the stream
XREAD [COUNT count] [BLOCK milliseconds] STREAMS key [key...] ID [ID...]	Return elements never seen in multiple streams whose IDs are greater than the IDs reported by the caller for each stream. Can be masked.
XGROUP CREATE key groupname id|$ [MKSTREAM]	Create a consumer group.
XGROUP CREATECONSUMER key groupname consumername	Create a consumer in the consumer group.
XGROUP DELCONSUMER key groupname consumername	Delete a consumer from the consumer group.
XGROUP DESTROY key groupname	Destroy a consumer group.
XGROUP SETID key groupname id|$	Set the consumer group to any last delivered ID value.
XGROUP HELP	Display useful text about different subcommands
XREADGROUP GROUP group consumer [COUNT count] [BLOCK milliseconds] [NOACK] STREAMS key [key...] ID [ID...]	Return new entries from the stream using the consumer group, or access the history of pending entries for a given consumer. Can be masked.
XACK key group ID [ID...]	Mark the pending messages as correctly processed, effectively removing them from the pending entry list of the consumer group. The return value of this command is the number of successfully acknowledged messages, that is, the IDs we can actually parse in the PEL.
XCLAIM key group consumer min-idle-time ID [ID...] [IDLE ms] [TIME ms-unix-time] [RETRYCOUNT count] [FORCE] [JUSTID]	Change (or obtain) the ownership of messages in the consumer group as if the message has been delivered to the specified consumer.

XAUTOCLAIM key group consumer min-idle-time start [COUNT count] [JUSTID]	Change (or obtain) the ownership of messages in the consumer group as if the message has been delivered to the specified consumer.
XPENDING key group [[IDLE min-idle-time] start end count [consumer]]	Return information and entries from the pending entry list of the stream consumer group, that is, messages that have been obtained but never acknowledged.
LATENCY DOCTOR	Return a human-readable latency analysis report.
LATENCY GRAPH event	Return the latency graph of the event.
LATENCY HISTORY event	Return the timestamp latency samples of the event.
LATENCY LATEST	Return the latest latency samples of all events.
LATENCY RESET [event [event...]]	Reset the latency data of one or more events.
LATENCY HELP	Displays useful text about the different subcommands.

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com