Tair用户使用手册

# 前提

Tair集群搭建完毕。

Diamond服务搭建完毕。

# 配置diamond信息

这里假设Tair集群配置为：

Master cs：192.168.0.1:5198

slave cs：192.168.0.1:5198

Groupname: group\_1

暂定该集群名称为alitair，

用户服务器ip段为10.\*和20.\*

我们以此为例说明diamond的配置方法。

**步骤一.** 配置

Dataid：alitair

Group：alitair-A

配置项值：

{"clusters":[

{

"address":

{

"master":"192.168.0.1:5198",

"slave":"192.168.0.2:5198",

"group":"group\_1"

},

"readWeight":"1",

"writeWeight":"1"

}

]}

注：该配置是在diamond里面记录集群的配置信息，Group这里暂时命名为alitair-A，A可以用机房名或其他来替代标识。

**步骤二.** 配置

Dataid：alitair

Group：alitair-GROUP

配置项值：

1=alitair-A

2=alitair-A

注：该配置是在diamond里面确定路由规则，上述配置表示应用服务器ip段为1打头或者2打头的会访问到diamond中Dataid为alitair，Group为alitair-A的配置项中配置的集群上。这里的路由规则为最长匹配规则。

以上就是diamond的配置方法，下面就可以用配置的alitair这个Dataid使用tair客户端了。

# 使用java客户端

## 初始化方式

### Spring配置

**<bean** id="tairManager" class="com.taobao.tair.impl.mc.MultiClusterTairManager" init-method="init"**>**

**<property** name="configID"**>**

**<value>alitair</value>//**diamond中配置的dataid

**</property>**

**<property** name="dynamicConfig"**>**

**<value** type="java.lang.Boolean"**>**true**</value>**

**</property>**

**</bean>**

需要指定超时时间的应用，请加属性：

**<property** name="timeout"**>**

**<value>**500**</value>** // 单位为ms，默认2s

**</property>**

### 代码配置

**public** **void** testInitMultiCluster() {

MultiClusterTairManager mcTairManager = **new** MultiClusterTairManager();

mcTairManager.setConfigID("alitair"); *//diamond上的dataid，*

mcTairManager.setDynamicConfig(**true**); *//非常重要，不要忘记*

*//mcTairManager.setTimeout(500); //如需配置超时时间，单位为ms，默认2s*

mcTairManager.init();

......

}

## 接口说明

### put

ResultCode put(int namespace, Object key, Serializable value)

**描述**

写入数据，key若存在则强制更新，不过期

**参数：**

namespace - 申请时分配的namespace

key - key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode put(int namespace, Object key, Serializable value, int version)，ResultCode put(int namespace, Object key, Serializable value, int version, int expireTime)

### put

ResultCode put(int namespace, Object key, Serializable value, int version)

**描述**

写入数据，可用version实现并发处理，不过期

**参数：**

namespace - 申请时分配的namespace

key - key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

version - 参见 3.3节 特殊参数说明。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode put(int namespace, Object key, Serializable value)

ResultCode put(int namespace, Object key, Serializable value, int version, int expireTime)

### put

ResultCode put(int namespace, Object key, Serializable value, int version, int expireTime)

**描述**

写入数据，可设置版本号和过期时间

**参数：**

namespace - 申请时分配的namespace。

key - key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

参见 3.3节 特殊参数说明。

expireTime – 参见 3.3节 特殊参数说明。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode put(int namespace, Object key, Serializable value)，ResultCode put(int namespace, Object key, Serializable value, int version)

### get

Result<DataEntry> get(int namespace, Object key)

**描述**

读取key相对应的value。

**参数**

namespace - 申请时分配的namespace。

key - key，不超过1k。

**返回值**

Result对象，可用isSuccess()方法判断请求是否成功，再用getRc()方法获取到ResultCode。isSuccess有两种情况，ResultCode.SUCCESS表示读成功，ResultCode.DATANOTEXSITS表示数据不存在。其余情况皆表示读取失败。其他返回值参见3.4 节常见返回码说明。

### delete

ResultCode delete(int namespace, Object key)

**描述**

删除key。

**参数：**

namespace - 申请时分配的namespace

key - key，不超过1k

**返回值**

ResultCode对象，ResultCode.SUCCESS表示删除成功。其他返回值参见3.4 节常见返回码说明。

### mget

Result<List<DataEntry>> mget(int namespace, List<Object> keys)

**描述**

批量读取key。

**参数：**

namespace - 申请时分配的namespace

List<Object> - key列表，key的个数没有硬性规定，但过多的key可能会导致超时。

**返回值**

Result对象，Result List中仅返回数据存在且查询成功的key。其中ResultCode.SUCCESS表示所有key信息获取成功。ResultCode.PARTSUCC表示部分成功，如果有key不存在，也会返回该错误码，用户需对未在Result List中未返回的key再次查询，确认失败原因。Result.CONNERROR表示和所有server链接失败。其他返回值参见3.4 节常见返回码说明。

### mdelete

ResultCode mdelete(int namespace, List<Object> keys)

**描述**

批量删除key。

**参数：**

namespace - 申请时分配的namespace

key - key列表，key的个数没有硬性规定，过多的key可能会导致超时。

**返回值**

ResultCode对象， ResultCode.SUCCESS表示删除成功，其他返回值参见3.4 节常见返回码说明。

### setCount

ResultCode setCount(int namespace, Serializable key, int count)

**描述**

设置计数。设置某一个key的当前计数，可配合incr/decr使用。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - 设置的key，不超过1k

count - 要设置的计数值。

**返回值**

ResultCode 对象，ResultCode.SUCCESS表示读成功，其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode setCount(int namespace, Serializable key, int count, int version, int expireTime)

### setCount

ResultCode setCount(int namespace, Serializable key, int count, int version, int expireTime)

**描述**

设置计数。设置某一个key的当前计数，可配合incr/decr使用。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - 设置的key，不超过1k

count -要设置的计数值。

version - 参见 3.3节 特殊参数说明。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

ResultCode 对象，ResultCode.SUCCESS表示读成功，其余表示失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode setCount(int namespace, Serializable key, int count)

### incr

Result<Integer> incr(int namespace, Serializable key, int value, int defaultValue, int expireTime)

**描述**

增加计数。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量。

defaultValue - 第一次调用incr时的key的count初始值，第一次返回的值为defaultValue + value。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

Result对象，返回值可为负值。当key不存在时，第一次返回defaultValue+ value。 后续的incr基于该值增加value。

**另见**

[ResultCode incr (int namespace, Serializable key, int value, int defaultValue, int expireTime, int lowBound, int upperBound)](http://baike.corp.taobao.com/index.php/CS_RD/tair/java_detail#incr_2)

### incr

Result<Integer> incr(int namespace, Serializable key, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

**描述**

增加计数。可设置最大值和最小值。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量。

defaultValue - 第一次调用incr时的key的count初始值，第一次返回的值为defaultValue + value。

expireTime - 参见 3.3节 特殊参数说明。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，返回值可为负值。返回COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。当key不存在时，第一次返回defaultValue+ value。 后续的incr基于该值增加value。

**另见**

Result<Integer> incr(int namespace, Serializable key, int value, int defaultValue, int expireTime)

### decr

Result<Integer> decr(int namespace, Serializable key, int value, int defaultValue, int expireTime)

**描述**

减少计数。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量。

defaultValue - 第一次调用decr时的key的count初始值，第一次返回的值为defaultValue - value。

expireTime -参见 3.3节 特殊参数说明。

**返回值**

Result对象，返回值可为负值。当key不存在时，第一次返回defaultValue - value。 后续的incr基于该值减少value。

**另见**

ResultCode decr(int namespace, Serializable key, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

### decr

Result<Integer> decr(int namespace, Serializable key, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

**描述**

减少计数。可设置最大值和最小值。注意：不可和put接口混用！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量。

defaultValue - 第一次调用decr时的key的count初始值，第一次返回的值为defaultValue - value。

expireTime - 参见 3.3节 特殊参数说明。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，返回值可为负值。返回COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。当key不存在时，第一次返回defaultValue - value。 后续的incr基于该值增加value。

**另见**

Result<Integer> decr(int namespace, Serializable key, int value, int defaultValue, int expireTime)

### prefixPut

ResultCode prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value)

**描述**

写入带子key数据，pkey和skey若存在则强制更新，不过期。

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version)，prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version, int expireTime)

### prefixPut

ResultCode prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version)

**描述**

写入带子key数据，可用version实现并发处理，不过期。

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

version - 参见 3.3节 特殊参数说明。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value)，prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version, int expireTime)

### prefixPut

ResultCode prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version, int expireTime)

**描述**

写入带子key数据，可用version实现并发处理，可设置过期时间

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

value - 可序列化对象value，不超过1M。基于性能考虑，建议在10k之内。

Version - 参见 3.3节 特殊参数说明。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

ResultCode对象。ResultCode.SUCCESS表示写成功，其他表示写失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value)，prefixPut(int namespace, Serializable pkey, Serializable skey, Serializable value, int version)

### prefixGet

Result<DataEntry> prefixGet(int namespace, Serializable pkey, Serializable skey)

**描述**

读取key相对应的value

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

**返回值**

Result对象，可用isSuccess()方法判断请求是否成功，再用getRc()方法获取到ResultCode。isSuccess有两种情况，ResultCode.SUCCESS表示读成功，ResultCode.DATANOTEXSITS表示数据不存在。其余情况皆表示读取失败。其他返回值参见3.4 节常见返回码说明。

### prefixDelete

ResultCode prefixDelete(int namespace, Serializable pkey, Serializable skey)

**描述**

删除本集群数据

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

**返回值**

ResultCode对象， ResultCode.SUCCESS表示删除成功。其他返回值参见3.4 节常见返回码说明。

### prefixPuts

Result<Map<Object, ResultCode>> prefixPuts(int namespace, Serializable pkey, List<KeyValuePack> keyValuePacks)

**描述**

写入带多子key数据，可用version实现并发处理，可设置过期时间

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

keyValuePacks - 子key和value的列表，KeyValuePack有多个构造函数重载，可以对该kv设置version和expire，具体参见com.taobao.tair.etc.KeyValuePack。

**返回值**

Result对象，可用getRc()方法获取到ResultCode。ResultCode.SUCCESS表示写成功，ResultCode.PARTSUCC表示部分成功，其余情况皆表示写入失败。map中存放是当返回值**不为**ResultCode.SUCCESS时各子key的写入返回值。其他返回值参见3.4 节常见返回码说明。。

**另见**

prefixPuts(int namespace, Serializable pkey, List<KeyValuePack> keyValuePacks, List<KeyCountPack> keyCountPacks)

### prefixPuts

Result<Map<Object, ResultCode>> prefixPuts(int namespace, Serializable pkey, List<KeyValuePack> keyValuePacks, List<KeyCountPack> keyCountPacks)

**描述**

融合了prefixPuts和prefixSetCounts两个接口，写入带子key数据，即子key既可以是kv数据，也可以是计数器数据。比如pkey是用户id，skey既有用户的计数信息，又有对象信息，用这个接口可以一次性完成，减少网络交互。

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

keyValuePacks - 子key和value的列表，KeyValuePack有多个构造函数重载，可以对该kv设置version和expire，具体参见com.taobao.tair.etc.KeyValuePack。

keyCountPacks - 子key和计数count的列表，KeyCountPack有多个构造函数重载，可以对该kv设置version和expire，具体参见com.taobao.tair.etc.KeyCountPack。

**返回值**

Result对象，可用getRc()方法获取到ResultCode。ResultCode.SUCCESS表示写成功，ResultCode.PARTSUCC表示部分成功，其余情况皆表示写入失败。map中存放是当返回值**不为**ResultCode.SUCCESS时各子key的写入返回值。 其他返回值参见3.4 节常见返回码说明。。

**另见**

Result<Map<Object, ResultCode>> prefixPuts(int namespace, Serializable pkey, List<KeyValuePack> keyValuePacks)

### prefixGets

Result<Map<Object, Result<DataEntry>>> prefixGets(int namespace, Serializable pkey, List<? extends Serializable> skeys)

**描述**

读取带多个子key数据相对应的value数据

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skeys - 子key列表

**返回值**

Result对象，可用getRc()方法获取到ResultCode。ResultCode.SUCCESS表示写成功，ResultCode.PARTSUCC表示部分成功。map中存放是读取成功各子key的value值。其余情况皆表示读取失败。其他返回值参见3.4 节常见返回码说明。

### prefixDeletes

Result<Map<Object, ResultCode>> prefixDeletes(int namespace, Serializable pkey, List<? extends Serializable> skeys)

**描述**

批量删除key，删除一个机房的数据，适用于双机房单集群配置。

**参数：**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skeys - 子key列表

**返回值**

Result对象，可用getRc()方法获取到ResultCode。ResultCode.SUCCESS表示删除成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。map中存放是各子key的返回值。其余情况皆表示删除失败。其他返回值参见3.4 节常见返回码说明。

### mprefixGets

Result<Map<Object, Map<Object, Result<DataEntry>>>> mprefixGets(int namespace, Map<? extends Serializable, ? extends List<? extends Serializable>> pkeySkeyListMap )

**描述**

读取带多个pkey多个子key数据相对应的value数据

**参数：**

namespace - 申请时分配的namespace

pkeySkeyListMap - 主key子key列表，一个pkey可以带上多个子key作为map的一条数据。

**返回值**

Result对象，可用getRc()方法获取到ResultCode。ResultCode.SUCCESS表示写成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。map中存放返回成功的主key子key对应的value返回值。其余情况皆表示读取失败。其他返回值参见3.4 节常见返回码说明。

### prefixSetCount

ResultCode prefixSetCount(int namespace, Serializable pkey, Serializable skey, int count)

**描述**

设置子key计数。可以用于prefixincr/prefixdecr之前或者之后。

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

count - 要设置的值

**返回值**

ResultCode 对象，ResultCode.SUCCESS表示设置成功，其余表示失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode prefixSetCount(int namespace, Serializable pkey, Serializable skey, int count, int version, int expireTime)

### prefixSetCount

ResultCode prefixSetCount(int namespace, Serializable pkey, Serializable skey, int count, int version, int expireTime)

**描述**

设置子key计数。可以用于prefixincr/prefixdecr之前或者之后。可用version控制并发，可配置过期时间。

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

skey - 子key，不超过1k

count - 要设置的值

version - 参见 3.3节 特殊参数说明。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

ResultCode 对象，ResultCode.SUCCESS表示设置成功，其余表示失败。其他返回值参见3.4 节常见返回码说明。

**另见**

ResultCode prefixSetCount(int namespace, Serializable pkey, Serializable skey, int count)

### prefixIncr

Result<Integer> prefixIncr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime)

**描述**

增加子key计数。当子key不存在时，第一次返回defaultValue+ value，以后每次增加value。注意：prefixIncr前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量。

defaultValue - 第一次调用incr时的key的count初始值，第一次返回的值为defaultValue + value。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

Result对象，返回值可为负值。当key不存在时，第一次返回defaultValue+ value。 后续的incr基于该值增加value。

**另见**

Result<Integer> prefixIncr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

### prefixIncr

Result<Integer> prefixIncr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

**描述**

增加计数。当子key不存在时，第一次返回defaultValue+ value，以后每次增加value。可设置最大值和最小值。注意：prefixIncr前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 增加量

defaultValue - 第一次调用incr时的key的count初始值，第一次返回的值为defaultValue + value。

expireTime - 参见 3.3节 特殊参数说明。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，返回值可为负值。返回COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。当key不存在时，第一次返回defaultValue+ value。 后续的incr基于该值增加value。

**另见**

Result<Integer> prefixIncr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime)

### prefixDecr

Result<Integer> prefixDecr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime)

**描述**

减少子key计数。当子key不存在时，第一次返回defaultValue - value，以后每次减少value。注意：prefixDecr前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 减少量

defaultValue - 第一次调用prefixDecr时的子key的count初始值，第一次返回的值为defaultValue - value。

expireTime - 参见 3.3节 特殊参数说明。

**返回值**

Result对象，返回值可为负值。当key不存在时，第一次返回defaultValue - value。 后续的incr基于该值减少value。

**另见**

Result<Integer> prefixDecr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

### prefixDecr

Result<Integer> prefixDecr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

**描述**

减少子key计数。当子key不存在时，第一次返回defaultValue - value，以后每次减少value。可设置最大值和最小值。注意：prefixDecr前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

key - key列表，不超过1k

value - 减少量

defaultValue - 第一次调用prefixDecr时的key的count初始值，第一次返回的值为defaultValue - value。

expireTime - 参见 3.3节 特殊参数说明。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，返回值可为负值。返回COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。当key不存在时，第一次返回defaultValue - value。 后续的prefixDecr基于该值减少value。

**另见**

Result<Integer> prefixDecr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime)

### prefixSetCounts

Result<Map<Object, ResultCode>> prefixSetCounts(int namespace, Serializable pkey, List<KeyCountPack> keyCountPacks)

**描述**

设置多个子key的当前计数，可配置子key的version和expireTime。可以用于prefixincr(s)/prefixdecr(s)之前或者之后。

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

keyValuePacks - 子key和value的列表，KeyValuePack有多个构造函数重载，可以对该kv设置version和expire，具体参见com.taobao.tair.etc.KeyValuePack。

**返回值**

Result对象，可用getRc()函数获取ResultCode。ResultCode.SUCCESS表示设置成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。map中返回了失败的子key对应的返回值。成功时，map为null。其他返回值参见3.4 节常见返回码说明。

### prefixIncrs

Result<Map<Object, Result<Integer>>> prefixIncrs(int namespace, Serializable pkey, List<CounterPack> packList)

**描述**

增加多个子key计数。每个子key可以增加不同的count。注意：prefixIncrs前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

packList - count列表，CounterPack可配置子key的value和defaultvalue，另外也可以设置过期时间。

**返回值**

Result对象，可用getRc()函数获取ResultCode。ResultCode.SUCCESS表示设置成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。map中返回了所有失败子key对应的返回值。成功时，map为null。其他返回值参见3.4 节常见返回码说明。

**另见**

Result<Integer> prefixIncr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

### prefixIncrs

Result<Map<Object, Result<Integer>>> prefixIncrs(int namespace, Serializable pkey, List<CounterPack> packList, int lowBound, int upperBound)

**描述**

增加多个子key计数。每个子key可以增加不同的count。注意：prefixIncrs前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

packList - count列表，CounterPack可配置子key的value和defaultvalue，另外也可以设置过期时间。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，可用getRc()函数获取ResultCode。ResultCode.SUCCESS表示设置成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。map中返回了失败的子key对应的返回值。成功时，map为null。其他返回值参见3.4 节常见返回码说明。

**另见**

Result<Map<Object, Result<Integer>>> prefixIncrs(int namespace, Serializable pkey, List<CounterPack> packList)

### prefixDecrs

Result<Map<Object, Result<Integer>>> prefixDecrs(int namespace, Serializable pkey, List<CounterPack> packList)

**描述**

减少多个子key计数。每个子key可以减少不同的count。注意：prefixDecrs前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

packList - count列表，CounterPack可配置子key的value和defaultvalue，另外也可以设置过期时间，具体参见com.taobao.tair.etc.CounterPack。

**返回值**

Result对象，可用getRc()函数获取ResultCode。ResultCode.SUCCESS表示设置成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。map中返回了所有子key对应的返回值。

**另见**

Result<Integer> prefixDecr(int namespace, Serializable pkey, Serializable skey, int value, int defaultValue, int expireTime, int lowBound, int upperBound)

### prefixDecrs

Result<Map<Object, Result<Integer>>> prefixDecrs(int namespace, Serializable pkey, List<CounterPack> packList, int lowBound, int upperBound)

**描述**

减少多个子key计数。每个子key可以减少不同的count。注意：prefixDecrs前不要prefixPut！！

**参数**

namespace - 申请时分配的namespace

pkey - 主key，不超过1k

packList - count列表，CounterPack可配置子key的value和defaultvalue，另外也可以设置过期时间。

lowBound - 最小值

upperBound - 最大值

**返回值**

Result对象，可用getRc()函数获取ResultCode。ResultCode.SUCCESS表示设置成功，ResultCode.PARTSUCC表示部分成功，可对失败部分进行重试。COUNTER\_OUT\_OF\_RANGE表示超出了最小或最大值。map中返回了所有子key对应的返回值。

**另见**

Result<Map<Object, Result<Integer>>> prefixDecrs(int namespace, Serializable pkey, List<CounterPack> packList)

## 特殊参数说明

### expiredTime

expiredTime 是指数据的过期时间。一旦超过过期时间，数据将对应用不可见，后端清理策略和应用无关。对应用来说，过期时间始终是精确的。

调用接口时，expiredTime单位是秒。 expireTime <= 0，表示数据永不过期。expireTime > 0，表示设置过期时间。若expireTime>当前时间的时间戳（即Unix时间戳，需转换成距1970-1-1 00:00:00的秒数，比如：2014-01-01 00:00:00，则设置expireTime为1388505600），则表示使用绝对时间，否则使用相对时间。

### version

#### version的作用

在Tair的put接口中，有一个**version**参数，这个参数是为了**解决并发更新**同一个数据而设置的。

很多情况下，更新数据是先get，修改get回来的数据，然后put回系统。如果有多个客户端get到同一份数据，都对其修改并保存，那么先保存的修改就会被后到达的修改覆盖，从而导致数据丢失问题。在某些情况下，这个是我们不希望发生的。比如客户端A和B取到数据{key, value}，A将value修改为value\_a，B将value修改为value\_b，A先保存，B然后保存，则最终的value为value\_b，这时A的修改就不可见了。 version是为了防止这中情况而设置的。

服务器端version初始化为0，当数据第一次put时，version增加为1。后续每次更新数据，服务器端都会将version加1。每次get数据，服务器都会返回当前数据的版本，如果Tair中的数据版本与更新接口中传入的version相同，此次更新才会成功。否则如果在get与更新之间，数据被更新，导致Tair中的version与client传来的version不一致，该次更新会失败，返回ResultCode.VERERROR。version不会无限制增大，当version达到int16的最大值后，会重置为1。

如果应用不关心并发更新的一致性，调用客户端接口时，version必须不传或者传入0。  
**注意：**

1. version的增加是服务器端的行为，tair并不支持多版本数据。另外，更新也是使用put接口。

2. version的值是由服务器端决定的，和client指定的version值无关。

#### version使用方法

get接口返回的是DataEntry对象，该对象中包含get到的数据的版本号，可以通过getVersion()接口获得该版本号。在put时，将该版本号作为put的参数即可。 如果不考虑版本问题，则可设置version参数为0，系统将强行覆盖数据，即使版本不一致。

#### 如果返回version不一致，怎么办？

如果更新所基于的version和系统中当前的版本不一致，则服务器会返回ResultCode.VERERROR。 这时你可以选择重新get数据，然后在新版本的数据上修改；或者设置version为0重新请求，以达到强制更新的效果。

#### version具体使用案例

如果应用有10个client会对key进行并发put，那么操作过程如下：   
1. get key。如果get key成功，则进入步骤2；如果数据不存在，则进入步骤3.   
2. 在调用put的时候将get key返回的verison重新传入put接口。服务端根据version是否匹配来返回client是否put成功。   
3. get key数据不存在，则新put数据。此时传入的version必须不是0和1，其他的值都可以（例如10，要保证所有client是一套逻辑）。  
不传入0是因为，tair会认为强制覆盖；  
不传入1是因为，会出现两个client都写入成功的情况，不能防止并发。假设这样一个场景，client A第一次put key，写入成功，这时服务器端version为1。如果client B设置version为1，服务器端接收到后比较version正确，那么client B也会写成功。这和我们的预期就不一致了。而把version设为其他值时，服务器端接收到后比较version不同，则可以返回version error，client B就写失败了，需要重新get后获取到version值再进行更新。

#### 案例分析：分布式锁

tair中存在该key，则认为该key所代表的锁已被lock；不存在该key，在未加锁。操作过程和上面相似。业务方可以在put的时候增加expire，已避免该锁被长期锁住。

#### 我该不该使用Version？

这取决于用户对数据一致性的要求，如果对数据一致性有较高的要求，并且访问并发高，有很多类似append的操作，那么通过version可以避免数据的意外结果。   
当不需要关心并发的时候，不传入或者传入version为0。

#### 什么情况下version会重置为1？

第一次put到tair，或version已经达到int16的最大值。   
***NOTE：原则上，用户在通过 Get 接口拿到真实 version 之前，『不应该』对 version 的当前值做任何假定。***

## 常见返回码说明

|  |  |  |  |
| --- | --- | --- | --- |
| code | msg | 含义 | 解决方法 |
| 0 | success | 操作成功 |  |
| 1 | data not exist | 该key在tair中不存在 |  |
| -1 | connection error or timeout | 与tair服务器交互失败或超时 |  |
| -2 | unknown error | 服务器端接收到未知包 | 确认报错接口是否为合法接口。， |
| -5 | key length error | key的长度超过限制 | key长度超过1k，需减小key长度，过大key将极大影响集群性能。 |
| -6 | value length error | value的长度超过限制 | value长度超过1M，建议value最大最大不要超过10k，过大value将极大影响集群性能。 |
| -10 | partly success | 批量操作接口（如mget）部分成功 | 有部分key请求失败，遍历返回值可知道哪些key操作失败。 |
| -3967 | key should proxy | 该key将由另一台 proxy ds服务 | 集群发生迁移，请稍等片刻后重试。 |
| -3968 | server-side task-queue overflowed | invalid报错，server端队列满 |  |
| -3979 | count reach zero | count减到了0 |  |
| -3980 | count reach bound | count超过了限制 |  |
| -3981 | can not override | key已用普通接口写入，无法用计数器接口覆盖 | 计数器接口和普通接口不能混用，incr/decr前不要put，prefix系列类似。 |
| -3982 | invalid argument | 参数非法 | 接口参数使用错误，具体参见百科 开发文档 |
| -3984 | migrate busy | server端数据迁移中 | server端数据迁移中，请稍等片刻重试 |
| -3986 | write not on master | key对应的数据服务器发生变化 | server端数据迁移中，请稍等片刻重试 |
| -3987 | server can not work | 服务器无法提供服务 | 服务器故障。 |
| -3990 | send packet failed | 发包失败 | 客户端发包失败，请检查应用服务器网络状态 |
| -3991 | item size error | key或value大小不对 | 检查key或value大小是否为0，key是否大于1k，value是否大于1M。如果是incr/decr/setcount等计数类接口，还要看value是否小于0。 |
| -3994 | serialize error | 序列化错误 | 1. 确认tair初始化是否成功，如果没有初始化成功，那么写入和读取接口都会返回-3994错误。 2. 如果是写入接口报该错误，一般是key或者value传入的对象中存在有对象没有序列化函数，请详细排查自定义对象中的各成员变量。class要仔细检查每一个成员！！！ 写失败还有一种情况是value > 1M。 3. 如果是读取接口报该错误，a.请查看是否传入的key或者value的序列化方式发生变化了，比如字段序列化顺序发生变化，或者增减了字段，class路径发生变化等等。b. 请确认你的代码 能够找到 要反序列化的 自定义类. |
| -3997 | version error | 指定的version和server不一致 | 一般发生在高并发写的情况，防止并发。具体参见关于version的介绍 |
| -3999 | server error | server异常 |  |
| -5114 | counter out of range | 计数器超出设置的[lowbound, upbound)返回，没有设置范围的则是超出了int（有符号数）的范围。 |  |
| -6000 | rpc overflow | 服务器端限流。 | 用户需检查是否有热点访问，或者是流量访问突增。 |