跑酷项目数据传输协议文档

|  |  |  |  |
| --- | --- | --- | --- |
| 修改记录 | 修改人 | 修改时间 | 版本 |
| 创建 | 林志贤 | 2014.10.23 | V1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 游戏域名

[http://parkour.funugame.com](http://parkour.funugame.com/)

# MD5 key

某些提交修改数据的接口要加上MD5签名，详细查看接口定义

MD5 Key: 是在用户注册时返回给用户，以后不再改变

在提交数据时必须按指定格式用该key签名，以便检查提交的有效性

协议统一信息格式说明:

头部需要带上基本的字段有： deviceId (string) ,imis(string) ,model(string )

发送方式统一为Http Post

内容格式统一为 json格式字符串

# 配置接口定义

**约定一**：请求地址： /parkour/controller 注意：所有功能接口的url都是一样的

**约定二**：请求的数据都是json格式，返回也是json格式

**约定三**：Request 的json格式以:{“功能名”：{数据内容}}，可以同时请求多个数据内容，例如：

{

“login” :{.............},

“updateRoleName”:{.....}

}

**约定四**：Response 为相应的 {“功能名+Response”：{数据内容}},例如：

{

“loginResponse” :{.............},

“updateRoleNameResponse”:{.....}

}

说明：采用如上约定可以一次http请求多个功能和返回

**约定四**：下面功能名放在标题里，json数据只列出数据内容。

## 检查是否更新配置文件(checkDataVersion)

RequestBody:

{}

Response Body：

{

version:(int如果和请求的version值不同就去下面地址下载配置文件)

}

## 更新配置文件(updateData)

Request Body:

{}

Response Body：

{

"version": 11,

"fileList": {

"file1": {

"columns": [

"id",

"name"

],

"data": [

[

"1",

"name"

],

[

"2",

"name"

]

]

},

"file2": {

"columns": [

"id",

"name"

],

"data": [

[

"1",

"name"

],

[

"2",

"name"

]

]

}

}

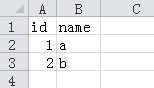
}

a)可能有多个config\_fileName，每个config\_fileName都是客户端已经知道的名字，客户端根据名字取得配置内容！

b)客户端要保存version，和配置文件内容到本地

c)配置文件在服务端是一个二维表结构：

例如：config\_file1.txt



## 注册或登录(login)

Request Body:

{

uid:(String),

password(string)

}

}

该正式提交的内容应该是：

{“**login**”:{“uid”:123,”password”:”pass”}}

Response 正式内容应该是：

{“loginResponse”:{...数据格式....}}

为了简化文档，下面功能说明都只说明数据格式的内容。

Response Body：

{

code:0/1(int),

new:true/false(boolean 是否新用户),

md5key:(string 新用户注册有该项，用来更新数据签名)，

user:{uid:(int),

account(string账号名),

password(string isNew=1时该处有值，否则为null),

rolename:(string角色名),

rank:(int 荣耀排行，如果该值大于0表示有排名，去配置表里找相应排名奖励),

exp:(int经验值),  
 age:(int年龄),

gold:(int金币),

diamond:(int钻石),

strength:(int体力),

piece:(int 碎片),

bone:(int 骨头

daynum(每日奖励每周连续登陆天数（1-7）)

proclamation( 公告内容)

proclamationtime（发布公告时间）

version（发布公告版本号）

},

pets:{//宠物，id对应宠物表：pet.txt

[

{“id”:(int),”level”:(int)},

....

]

},

tech:{//科技，id对应科技表：tech.txt

[

{“id”:(int科技id),”level”:(int等级)}，

...

]

},

achievement:{//成绩列表(这里只有可领或已领)

[

{“id”:(int),”status”:(int status 1可领，2 已领)}

.....

]

}

}

## 更新角色名(updateRoleName)

Request Body:

{

uid:(int),

roleName:(string)

}

Response Body：

{

code:0/1(int 1 成功，0失败),

roleName:(string,更新后的角色名)

}

## 每日奖励领取

获得奖励有金币和道具在LonginReward.txt中有对应哪一天（1-7）所获得金币数量和道具的数量，前4天获得的道具类型参照props.txt随机选取id为11-15的道具（暂定，参考跑酷游戏细节策划，详询设计），判断第几天登陆的天数，为注册和登录（login）中的user中的daynum,isaward(true,false)🡪true:表示已领取，false：未领取；当客户端点击领取时需在sync中同步数据

{

"sync": {

"loginaward": {"uid": int,"award": true},

"prop": [{"mark": "insert/update/delete","uid": int,"id": int,"num": int}]——（uid:用户id，id：道具id）

……..

}

}

|  |  |
| --- | --- |
|  | 所获奖励 |
| 第一天 | 金币+5000 道具\*1（随机） |
| 第二天 | 金币+8000 道具\*3（随机） |
| 第三天 | 金币+15000 道具\*5（随机） |
| 第四天 | 荣耀值+10 道具\*5（随机） |
| 第五天 | 荣耀值+50 金币+30000 |
| 第六天 | 荣耀值+80 金币+10000 |
| 第七天 | 6个月减衰药剂 |

## 同步数据（sync）

**原则：为了尽量减少客户端和服务的交互甚至依赖，甚至在没网络的时候也可以玩。**

服务端只做数据保存功能，数据的变化由客户维护，然后同步到服务端

用户在登陆时会把用户的所有数据都传给用户

无需每次变更都同步，可以累积到一定程度再更新，或者在一些关键步骤再更新， 例如：一局比赛结算，游戏退出等。

Response Body：

（关于传入mark简要说明：根据传入的mark判断insert或者update或者delete数据

其中prop，tech，pet，piece的mark需传入insert/update/delete即可

Achievement需传入insert，update数据；

Vary无mark）

{

“uid”:int,

"sync": {

"loginaward": {"award": true},--（领取每日奖励）

"prop": [{"mark": "insert/update/delete","id": int,"num": int}],——（uid:用户id，id：道具id）

"vary": {"exp": int,"age": int,"gold": int,"diamond": int,"strength": int,"bone": int},（upadte数据）

“friend”:[{“mark”:”insert/delete”,”fuid”:int}],

"pet":[{"mark": " insert/update/delete ","id": int,"level": int}],——(uid:用户id，id：宠物id)

"tech":[{"mark": " insert/update/delete ","id": int,"level": int}],——（uid:用户id，id：科技道具

"piece":[{"mark": " insert/update/delete ","id": int,"piece": int}],——（uid:用户id，id：）

"achievement":[{"mark": "insert/update","id": ont,"status": int}]——（uid:用户id，id：成就id）

}

}

如下测试传入json格式：

{

“uid”:45,

"sync": {

"prop": [{"mark": "insert","id": 0,"num": 1}],

"vary": {"exp": 60,"age": 60,"gold": 60,"diamond": 60,"strength": 60,"bone": 60},

"friend": [{"mark": "insert","fuid": 56}],

"pet":[{"mark": "delete","id": 2,"level": 5}],

"tech":[{"mark": "delete","id": 3,"level": 6}], id）

"piece":[{"mark": "delete","id": 3,"piece": 6}],

"achievement":[{"mark": "insert","id": 2,"status": 6}]

}

}

返回Response Body：

{"syncResponse":{"code":1,"code\_prop":false,"code\_vary":true,"code\_friend":false,"code\_pet":false,"code\_tech":true,"code\_piece":true,"code\_achievement":false}}

（"code":1：表示传入”sync”中所有数据传入并处理完成，"code\_prop":false表示对道具传入数据未完成修改，如mark为“delete“时"code\_prop":false表示未删除。。。）

## 提交比赛结果(submitGameResult)

Request Body:

{

type:(int;0普通比赛 1挑战进阶 2 好友挑战),

uid:(int),

muid:(int 挑战用户id，type为1、2时才有该值),

chapterid:(int场景id),

distance:(int距离),

score:(int得分）,

exp:(int得到经验值),

gold:(int),

piece:(int 得到碎片),

bone:(int得到骨头),

sign:(string,签名

md5(按参数的首字母顺序连接起来的字符串+key)

}

Response Body：

{

type:(int;0普通比赛 1挑战进阶 2 好友挑战),

win:(boolean true 胜利，false失败，type为1,2才会有该值，客户端根据该值增加胜利奖励：奖杯金币等)

code：0/1(1有效，0无效，如果是挑战已经赢过的挑战用户这里返回0,挑战已经挑战过的好友也返回0)（成功后，自行增加客户金币等值），

}

## 获得荣耀排行榜(scoreRank)

Request Body:

{

“uid”:(int)

chapterid:(int值得哪个场景的排行，如果是总排行，传0),

}

Response Body：

{

“ranklist”:[

{"uid":45,"rolename":null,"score":820,"age":0,"cup":0,"cuplevel":0},

...

]，

“score”:(int自己的当前最佳得分)

}

## 获得：好友排行/好友系统/好友挑战(friendScoreRank)

Request Body:

{

“uid”:(int)

chapterid:(int值得哪个场景的排行，如果是总排行，传0),

}

Response Body：

{

“ranklist”:[

{"uid":45,"rolename":null,"score":820,"age":0,"cup":0,"cuplevel":0， result:(int挑战结果：0 无；1赢，2输)},

{ ... }，

.....

]，

“score”:(int自己的当前最佳得分)

}

## 获得消息列表（getUserMessageList）

（在加好友发送请求时自动产生好友消息）

Request Body:

{

"getUserMessageList":{

"type":int, （0：系统消息 / 1:好友消息）

"uid":int

}

}

Request Body:

{"getUserMessageListResponse":{"code":1,"error":null,"userMessageList":[{"id":5,"uid":548,"type":1,"content":"2015-01-07 11:33:05 游客931 添加您为好友","from\_uid":547,"isread":0}]}}

Code：1 / 5 1：成功获取用户消息 / 5：失败

Uid：用户 from\_uid：加你为好友的用户

type=1 好友消息 type=0 系统消息

content：消息内容 isread：是否已查看消息

## 获得游戏公告（getNoticeList）

Request Body:

{

"getNoticeList":{

"type":int

}

}

Type：

(0:所有公告，包括游戏公告和滚屏所有公告，即设计文档中5.1和5.2所有公告；

1：游戏公告，即设计文档中5.1的系统公告；

2：滚屏公告，设计文档中的5.2中的第1个滚屏公告；

3：滚屏公告，设计文档中的5.2中的第2个滚屏公告；

4：滚屏公告，设计文档中的5.2中的第3—7的滚屏公告)

Response Body： (当传入type为1—4时)

触发 重复次数 滚屏时间 间隔时间 是否循环

Tigger repeatnum scrolltime intervaltime iscycle

Tigger ：0,重新进入游戏，1游戏中

Iscycle：0，不循环；1，循环

{"getNoticeListResponse":{"code":1,"error":null,"userNoticeList":[{…}],"userAllNoticeList":null}}

例，如下

{"getNoticeListResponse":{"code":1,"error":null,"userNoticeList":[{"id":1,"type":1,"tigger":0,"repeatnum":0,"scrolltime":0,"intervaltime":0,"iscycle":0,"content":"系统V1.0更新。。。","time":1420767190000,"version":1}],"userAllNoticeList":null}}

“code”:(int:1/5 1成功 / 5失败)

Response Body： (当传入type为0时)

{

"getNoticeListResponse":{"code":1,"error":null,"userNoticeList":null,"userAllNoticeList":[{…}]}

}

例，如下

{"getNoticeListResponse":{"code":1,"error":null,"userNoticeList":null,"userAllNoticeList":[{"gameNoticeList":[{"id":1,"type":1,"tigger":0,"repeatnum":0,"scrolltime":0,"intervaltime":0,"iscycle":0,"content":"系统V1.0更新","time":1420767190000,"version":1}]},{"scrollSystemNoticeListOne":[{"id":2,"type":2,"tigger":0,"repeatnum":2,"scrolltime":10,"intervaltime":1,"iscycle":0,"content":"《跑到大》1.1版本更新内容，增加了新的宠物和新的障碍，修复了XX问题。","time":1420767274000,"version":1}]},{"scrollSystemNoticeListTwo":[{"id":3,"type":3,"tigger":1,"repeatnum":0,"scrolltime":10,"intervaltime":300,"iscycle":0,"content":"活动1：充值大酬宾，充值任意金额即可获得双头龙宠物。\r\n活动2：个人阶位达到3阶，即可获得400个瓢虫宠物兑换碎片。\r\n活动3：连续登陆７天，即可获得300个牦牛宠物兑换碎片。\r\n活动4：累计奔跑100000米，即可获得独角兽宠物。\r\n","time":1420767438000,"version":1}]},{"scrollUserNoticeList":[]}]}}

## 领取荣耀排行奖励(getRankAward)

Request Body:

{

“uid”:(int)

}

Response Body：

{

“code”:(int:0/1 1领取成功，客户端根据配置获得奖励内容，自行增加奖励内容，并通过sync同步到服务端)

}

## 获取挑战进阶用户列表(matchUserList)

Request Body:

{

“uid”:(int),

“chapterid”:(int场景id)

}

Response Body：

{

[

{"uid":45,"rolename":null,"score":820,"age":0,"cuplevel":0,”result”:0},

...

]

}

result 为竞赛结果：0没有结果，1赢，2输

附加说明：如果没有用户列表生成相近得分的用户列表，如果有且全部赢了，也重新生成，如果全都挑战过了，但是有输的，24小时后才会重新生成。

## 强制重新生成挑战用户列表(reGenerateMatchUserList)

在用户付费的情况下调用该方法，付费后的用sync同步

Request Body:Response Body： 同 《获取挑战进阶用户列表》

## 用户反馈(userFeedBack)

Request Body:

{

“userFeedBack”:{

“uid”:(int),

“content”:(string), 反馈内容

“contacts”:(string) 联系方式

}（insert）

}

Response Body：

{

code:0/1(int 1 成功，0失败)

}

## 推荐好友列表

Request Body:

{

“getRecommendFriendList”:{“uid”:(int)}

}

Response Body：

{

code:0/1(int 1 成功，5失败)，“friendList”：{[{…}，{…}，{…}]}

}

如：

{"getRecommendFriendListResponse":{"code":1,"friendList":[{"uid":272,"age":18,"rolename":"游客839","cuplevel":0,"figureurl\_qq":null,"totalscore":9401},{"uid":272,"age":18,"rolename":"游客839","cuplevel":0,"figureurl\_qq":null,"totalscore":9401},{"uid":272,"age":18,"rolename":"游客839","cuplevel":0,"figureurl\_qq":null,"totalscore":9401},{"uid":272,"age":18,"rolename":"游客839","cuplevel":0,"figureurl\_qq":null,"totalscore":9401},{"uid":272,"age":18,"rolename":"游客839","cuplevel":0,"figureurl\_qq":null,"totalscore":9401}]}}

每请求一次，返回5个不同的好友(根据好友的经验值进行搜索，当游戏最开始人数少于5时，可能返回的好友数在1-5个，后续如果人数增多，基本每次返回5个不同的好友)

## 首测活动

（完成活动1任务，前端需传递用户uid，money(充值金额)大于0，方可完成任务，其他3个活动服务端进行判断）

活动1充值接口如下：

Request Body:

{

"rechargeMoney": {

"uid":int,

"money":int

}

}

Response Body：

{"rechargeMoneyResponse":false/true}

false:表示活动未开启或者money值小于0

true：表示完成活动1任务

获取用户活动列表（getUserTestActivity）

Request Body:

{

"getUserTestActivity":int 用户uid

}

Response Body：

{"getUserTestActivityResponse":{"code":1,"error":null,"userTestActivities":[{"uid":int,"aid":int,"result":int,"currentprogress":int,"targetprogress":int}…]}}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| code | uid | aid | result | currentprogress | targetprogress |
| 1 成功 | Int | aid：活动id，即testactivity.txt中的id | 0：不可领 | 完成活动的当前进度（活动1和活动4中，0表示未完成活动，1表示完成活动） | 完成活动的目标进度 |
| 5 失败 |  |  | 1：可领 |  |  |
|  |  |  | 2：已领 |  |  |

获得活动奖励petid，piecenum， activitycontent前端在testactivity.txt中取

petid：获得活动相应的宠物id（活动1——4）

piecenum：获得活动相应的宠物碎片（活动2，3）

activitycontent：活动内容

例，如下

{"getUserTestActivityResponse":{"code":1,"error":null,"userTestActivities":[{"uid":543,"aid":1,"result":1,"currentprogress":1,"targetprogress":1},{"uid":543,"aid":3,"result":2,"currentprogress":7,"targetprogress":7},{"uid":543,"aid":4,"result":1,"currentprogress":1,"targetprogress":1},{"uid":543,"aid":2,"result":0,"currentprogress":0,"targetprogress":3}]}}

领取活动按钮状态（updateUserTestActivityResponse）

Request Body:

{

"updateUserTestActivity": {

"uid": int,

"aid": int

}

}

Response Body：

{"updateUserTestActivityResponse":{"code":int,"error":null,"code\_userTestActivity":boolean}}

Code：1 / 5 成功 / 失败

Boolean：（false / true）领取失败 / 领取成功

奖励发放前端根据同步接口（Sync）进行处理，后台未处理

## 用户消息列表

获取用户消息列表（getUserMessageList）

Request Body:

{

"getUserMessageList":{

"type":null,

"uid":548

}

}

|  |
| --- |
| Type |
| null:表示拉去所有消息（包含系统消息，好友消息，用户反馈消息，荣耀小榜消息） |
| 0：表示指定拉去系统消息 |
| 1：拉取好友消息 |
| 2：用户反馈系统回复消息 |
| 3：荣耀小榜消息 |

Response Body：

{"getUserMessageListResponse":{"code":1,"error":null,"userMessageList":[{"uid":int,"type": int,"content":"您2015-01-16在古城遗迹跑出了6分，特此10000金币和10奖杯奖励。","from\_uid": int,"isread": int,"scenerank": int …}]}}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Type | from\_uid | Isread | scenerank |
| 1:成功 | 0：系统消息 | 系统消息from\_uid为0 | 0：用户为查看消息 | 为荣耀小榜各场景（from\_uid）玩家的排名 |
| 5：失败 | 1：好友消息 | 好友消息：发起消息者from\_uid为玩家的uid | 1：用户已查看消息 |  |
|  | 2：用户反馈系统回复消息 | 用户反馈系统回复消息from\_uid为0 |  |  |
|  | 3：荣耀小榜消息 | 荣耀小榜消息from\_uid表示场景的id，即场景1，目前有3个场景（基础训练场，南山洞穴，古城遗迹） |  |  |

荣耀小榜的奖励表为scenerankreward.txt id=1表示1-3名的奖励，id=2表示4-10名的奖励，id=3表示10名外的奖励

读取用户消息（readUserMessage）

Request Body:

{

"readUserMessage": {

"uid":int,

"from\_uid":int,

"type":int

}

}

需指定消息类型（type）和from\_uid进行更新

Response Body：

{"readUserMessageResponse":false/true}

false:表示未读取

true:表示已读取

# 签名

为提升同步数据时的安全性，防止他人随意篡改数据，为同步数据sync和提交比赛结果submitGameResult添加签名，其json格式稍作改变，但也兼容以上的所有json格式进行数据交互，签名的json格式如下

json格式：

{

"uid":45,

"sign":"MD5（MD5key+value）算法得到的结果值"

"value":{"userFeedBack":{

"content":"123",

"contacts":"123456"

},

"sync":{"loginaward":{"award":true},"prop":[{"mark":"insert","id":1,"num":2}],"vary":{"exp":50,"age":50,"gold":50,"diamond":60,"strength":60,"bone":60},"friend":[{"mark":"insert","fuid":49}],"pet":[{"mark":"insert","id":4,"level":4}],"tech":[{"mark":"insert","id":4,"level":4}],"piece":[{"mark":"insert","id":4,"piece":4}],"achievement":[{"mark":"insert","id":5,"status":5}]}}

}

}

Uid：签名方式下必须带用户id参数传递

Sign：签名后的值为：MD5key的值拼接value的json字符串组成一个新的字符串，再将新的字符串进行MD5加密生成一个加密后的字符串，与客户端生成的“sign”的值进行比较，所以需保证客户端的“value”的json格式与服务端一致

MD5key为用户注册时生成的MD5key，客户端有保存，“value”为json字符串“value”的值，其格式必须无空格且有序，否则会影响MD5计算的值，

Value：的值相当于同步数据（sync）中的未签名方式的json格式

以下为“value”json字符串值示例（不包含任何空格）

{"sync":{"loginaward":{"award":true},"prop":[{"mark":"insert","id":1,"num":2}],"vary":{"exp":50,"age":50,"gold":50,"diamond":60,"strength":60,"bone":60},"friend":[{"mark":"insert","fuid":49}],"pet":[{"mark":"insert","id":4,"level":4}],"tech":[{"mark":"insert","id":4,"level":4}],"piece":[{"mark":"insert","id":4,"piece":4}],"achievement":[{"mark":"insert","id":5,"status":5}]}}