平台文档

开放接口文档

API platform

开放接口文档

1. 开发指南

1.1 概览

离线文档下载

请在这里下载离线文档

该文档是总社对接使用文档，文档包含总社调用API实例和技术要求

1.1.1 接口概览

游戏相关

接口	说明
获取游戏列表	获取游戏列表
启动游戏	启动游戏
采集游戏记录	获取游戏记录
采集数据汇总	获取游戏记录汇总
获取游戏标签	获取游戏标签
获取游戏品牌组	获取游戏品牌组
游戏详情内嵌	获取游戏详情内嵌
获取原生包	获取原生包版本
获取押分	获取押分
创建赠送免费	创建赠送免费
获取赠送游戏列表	获取赠送游戏列表
获取赠送免费记录	获取赠送免费记录
获取对押监控汇总记录	获取对押监控汇总记录

玩家相关

接口	说明
创建玩家	创建玩家
踢出玩家	踢出玩家

转账钱包相关

接口	说明
获取玩家金币	获取玩家金币
存取玩家金币	存取玩家金币

免转钱包相关

接口	说明
请求钱包余额	请求钱包余额
下注/加注	下注/加注
结算金币	结算金币
下注回滚	下注回滚
更新金币	更新金币
通知进入游戏	通知进入游戏

代理人相关

接口	说明
创建代理人	创建代理人

系统相关

接口	说明
获取货币列表	获取货币列表

1.2 API参考

1.2.1 如何调用API

构造请求

请求URI

```
protocol://endpoint[:port]/path
```

示例请求(每个接口都有一个示例请求)

```
https://api.example.com/v1/game_list
```

参数说明

参数	必选	描述
protocol	是	请求使用协议, 如HTTP, HTTPS。HTTPS表示通过的安全的HTTPS访问该资源,支持的协议请参考环境列表中的 `支持协议`
endpoint	是	访问地址,请参考环境列表中的 `访问地址`
port	否	请求使用的端口,缺省时使用默认端口,HTTP默认端口为80，HTTPS默认端口为443
path	是	请求资源路径, 请参考各接口的 `请求路径` ,请求路径后加 `/` 和不加 `/` 表达不同的地址,需要注意区分， 不然会出现 `404`

HEADER

`Sn-Account` , `time` , `sign` 为每个接口都必须在header中携带的参数

参数	类型	是否必须	说明
Sn-Account	string	是	总社账号, 对接时发放
time	int64	是	请求时间, unix时间戳(10位)
sign	string	是	签名

请求BODY格式

```
{
    "自定义参数1":100,
    "自定义参数2":"test",
    "自定义参数N":"...."
}
```

参数说明

参数	类型	是否必须	说明
自定义参数1	string	否	自定义参数1
自定义参数2	type	否	自定义参数2
自定义参数N...	type	否	自定义参数N....

请求签名

签名步骤

1. `header` 中的 `Sn-Account` 和 `time` (unix时间戳(10位)),组成一个待签名Body体, 例如:

   启动游戏原始请求体为:

```
{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "fffffffffff",
    "agent_id": 1,
    "dealer_id": 1,
    "open_id": 90751924103860220
}
```

   添加Sn-Account和time参数后的请求体为:

```
{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "fffffffffff",
    "agent_id": 1,
    "dealer_id": 1,
    "open_id": 90751924103860220,
    "Sn-Account": "test123",
    "time": 1678092020
}
```

2. 按照字符代码对请求体中的 `Key` 进行升序排序,然后按照 `Key=Value&Key1=Value1` 格式进行拼接生成待签名字符串 `SignStr`, 其中value为 `空` 的字段不参与排序 例如:

```
agent_id=1&dealer_id=1&game_id=3003111&is_free=0&language=en&open_id=90751924103860220&Sn-Account=test123&third_name=fffffffffff&time=1678092020
```

3. 在待签名字符串 `SignStr` 最后追加 `&key=xxx` (其中key为API访问密钥, 会在对接时同Sn-Account一起发放)组成一个新的待签名字符串 `NewSignStr`, 例如:

```
agent_id=1&dealer_id=1&game_id=3003111&is_free=0&language=en&open_id=90751924103860220&Sn-Account=test123&third_name=fffffffffff&time=1678092020&key=xxx
```

4. 对待签名字符串 `NewSignStr` 进行Md5加密然后转成大写字符串, 例如:

```
4793F8CB4E113F084946FF46D61C8436
```

5. 发起请求

签名算法实现

Golang Postman

```
// GeneratedSign 签名算法: MD5,签名结果为大写
func GeneratedSign(params map[string]interface{}, key string) string {
    keys := make([]string, 0, len(params))
    for k := range params {
        if k != "" && k != "sign" && params[k] != nil && params[k] != "" {
            keys = append(keys, k)
        }
    }
    sort.Strings(keys)
    var sign string
    for _, v := range keys {
        sign += v + "=" + convertor.ToString(params[v]) + "&"
    }
    sign += "key=" + key
    fmt.Printf("sign:%v \n", sign)
    return strings.ToUpper(createMd5(sign))
}

// createMd5 生成md5字符串
func createMd5(str string) string {
    h := md5.New()
    h.Write([]byte(str))
    return hex.EncodeToString(h.Sum(nil))
}

//-----------------------------v2-----------------------
const dateTime = Date.now();
const timestamp = Math.floor(dateTime / 1000);
var Property = require('postman-collection').Property;
var resolvedBody = Property.replaceSubstitutions(pm.request.body.raw||'{}', pm.environment.toObject());
const body = JSON.parse(resolvedBody);
console.log("----request body: ", body)
body["time"] = timestamp
// 总社账号方式签名
body["sn_account"] = pm.environment.get('sn_account')
// 总社ID方式签名, 和总社账号签名二选一
// body["sn"] = pm.environment.get('sn')
const keys = Object.keys(body)
console.log("---request body keys:", keys.sort())
var sign = ""
for (const k of keys.sort()) {
    if (typeof body[k] == "object") {
        sign += k + "=" + JSON.stringify(body[k]) + "&"
    } else {
        sign += k + "=" + body[k] + "&"
    }

}
sign += "key=" +  pm.environment.get('secret_key')
console.log("------sign str: ", sign)
const sign_md5 = CryptoJS.MD5(sign).toString().toUpperCase()
console.log("----sign md5: ", sign_md5)
body["sign"] = sign_md5
console.log("-----request body: ", body)
pm.request.body.raw=JSON.stringify(body)
```

请求示例

启动游戏

```
curl --location 'https://api.gmlave.com/v1/launch_game' \
--header 'Content-Type: application/json' \
--header 'Sn-Account: test123' \
--header 'sign: 7E7881E609DF4CFCAAE7F9794A6DD564' \
--header 'time: 1678091419' \
--data '{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "fffffffffff",
    "agent_id": 1,
    "dealer_id": 1,
    "open_id": 90751924103860220
}'
```

公共请求参数

参数	必带	位置	类型	描述
Content-Type	是	header	string	请求数据类型, 当前只支持 `application/json`
X-Request-Id	否	header	string	请求的唯一标识符, 此ID仅用于故障排除，请勿使用此ID于重复请求的验证中。
Sn-Account	是	body	string	总社账户
time	是	body	int64	unix秒级时间戳(10位)
sign	是	body	string	客户端签名后的签名串携带在此字段中
need_http_code	否	query	string	非空则表示返回http状态码, 否则一律返回200
only_data	否	query	string	非空表示只返回数据，不返回code
zero_code	否	query	string	非空表示成功错误为 `0` ,默认为 `1001`

公共返回参数

携带need_http_code且不为空 **未携带need_http_code或为空** **携带only_date且不为空** **携带zero_code且不为空**

```
{
    "code": 100216,
    "success": false,
    "status_code": 409, <- 返回对应的HTTP状态码
    "system": 100,
    "msg": "player 'xxx' already exist in agency xxx",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Player.Create",
    "doc": "",
    "data": null
}

{
    "code": 100216,
    "success": false,
    "status_code": 200, <- 一律返回200
    "system": 100,
    "msg": "player 'xxxx' already exist in agency xxx",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Player.Create",
    "doc": "",
    "data": null
}
```

如果请求成功，只返回数据, 不返回 `code`，`message` 等表示错误的信息

```
{
    "open_id": 45749139335483393,
    "open_id_str": "45749139335483393"
}
```

如果失败, 只返回 `code`，`message` 等错误信息

```
{
    "code": 100214,
    "success": false,
    "status_code": 400,
    "system": 100,
    "msg": "invalid game id '20014411'",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Launch",
    "doc": "",
    "data": null
}
```

```
{
    "code": 0, <- 成功情况下错误码返回0
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Player.Create",
    "provider": "srv_admin",
    "doc": "",
    "data": {
        "open_id": 45749139335483393,
        "open_id_str": "45749139335483393"
    }
}
```

参数	参数位置	类型	描述
code	body	uint32	错误码
success	body	bool	请求是否成功
status_code	body	uint32	http状态码， 如果没有携带need_http_code参数且不为空则一律为200
system	body	uint32	系统码
message	body	string	如果失败，失败原因
prompt	body	string	国际化错误提示信息
request_id	body	string(int64)	链路追踪ID
request_method	body	string	请求方法
provider	body	string	服务提供者
doc	body	string	如果失败，处理错误参考文档地址
data	body	object	返回结果

公共错误码

错误名称	错误码	错误描述
INTERNAL_SERVER_ERR	1	系统内部错误
INVALID_REQUEST_ERR	2	无效的请求参数
OPERATE_DB_ERR	3	操作数据库失败
OPERATE_CACHE_ERR	4	缓存错误
MAINTAIN_ERR	5	服务维护中
TOO_MANY_REQUEST	6	请求过于频繁
ILEGAL_REQUEST	7	拆分成不同类型的非法请求
ILEGAL_TOKEN	8	非法token
ILEGAL_IP	9	非法IP请求
ILEGAL_SIGN	10	非法签名
INVALID_TOKEN_CONTENT	11	非法的token内容
REGION_REJECT	105	区域受限

环境列表

访问地址	支持协议	说明
请联系管理员	HTTPS	开放接口访问地址

多语言对照表

代码	语言
zh	简体中文
tw	繁体中文
en	英语
ko	韩语
ja	日语
th	泰语
vi	越南语
pt	葡萄牙语
id	印尼语
es	西班牙语
bn	孟加拉语
uk	乌克兰语
ru	俄语
ar	阿拉伯语
fil	菲律宾语
fr	法语
ur	乌尔都语
hi	印地语
ha	豪萨语
ta	泰米尔语

Postman

- 下载Collection并导入
- 下载Environment并导入

1.2.2 游戏相关

获取游戏列表

请求路径:

```
/v1/game_list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "language": "en",
    "agent_id": 12345,
    "page": 0,
    "size": 1
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
language	是	body	string	语言
agent_id	是	body	int64	代理人ID
page	否	body	int32	当前页数
size	否	body	int32	每页数量, 取值范围为[10,200], 当size=0时默认为10，当size>200时则默认为200，size<0代表所有
is_hot	否	body	int32	是否热门游戏, 0-否, 1-是, 不携带为所有, 和 `is_hot` 为或关系
is_new	否	body	int32	是否新游戏, 0-否, 1-是, 不携带为所有, 和 `is_new` 为或关系

返回体:

```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "game_id": 123456,
            "name": "xxx",
            "icon_link": "https://cdn-api-pc.xxx.app/icon/en/123456.png?v=1689760755",
            "loading_link": "",
            "game_type_id": 23,
            "game_sub_type": "y-xxx",
            "sort": 123456,
```

```
            "label": 1,
            "brand": 0,
            "is_hot": 0,
            "is_new": 0,
            "created": 1689760793532,
            "created_at": "1970-01-20T09:22:40-04:00",
            "updated": 1689760857423,
            "updated_at": "1970-01-20T09:22:40-04:00"
        }
    ]
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
game_id	是	body	int64	游戏ID
name	是	body	string	游戏名称
icon_link	是	body	string	游戏ICON
game_type_id	是	body	int64	游戏类型ID
game_sub_type	是	body	string	游戏类型名称
sort	是	body	int32	排序值
is_hot	是	body	int32	是否热门， 0-否， 1-是
is_new	是	body	int32	是否最新， 0-否， 1-是
introduction	是	body	string	游戏介绍
label	是	body	int64	系列，1 欧美系列，2 东方系列， 水果系列
brand	是	body	int64	品牌ID

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/game_list' \
--header 'Content-Type: application/json' \
--data '{
    "language": "en",
    "agent_id": 1,
    "page": 0,
    "size": 1
}'
```

启动游戏

如果是首次启动游戏则需要创建玩家

请求路径:

```
/v2/launch_game
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "xxx",
    "agent_id": 123,
    "return_url": "",
    "recharge_url": "",
    "terminal": "pc",
    "extra": "",
    "launch_params": {
        "xxx": "yyy"
    },
    "client_ip": "xxx.xxx.xxx.xxx"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
game_id	是	body	int64	游戏
language	否	body	string	语言, 默认英语
is_free	否	body	int32	是否试玩, 0-否，1-是
third_name	是	body	string	第三方用户名称, 如果为非试玩则需要携带此字段且 该玩家已通过接口创建玩家创建
agent_id	是	body	int64	代理ID
return_url	否	body	string	游戏返回包网指定路径, 如果是APP内则参考APP内返回
recharge_url	否	body	string	充值跳转地址
terminal	否	body	string	游戏终端pc 或 h5
extra	否	body	string	json格式字符串, 参考下面 `extra` 字段描述
launch_params	否	body	map	自定义启动参数, 将会携带在返回的URL上, 其中 `game_id` , `language` , `return_url` , `recharge_url` , `api_game` , `api_game_token` 为保留字段不可携带在此字段中, 否则有可能会出现游戏无法进入问题
client_ip	是	body	string	玩家客户端IP

extra可选参数:

参数	必填	类型	描述
nick_name	否	string	昵称
avatar	否	string	头像
use_coin_type	否	string	货币列表中的字符串标识符,默认使用代理人的货币
sum_recharge	否	map<string,int64>	充值总计, key为货币Code
sum_withdraw	否	map<string,int64>	提现总计, key为货币Code
recharge_times	否	int64	充值次数
share_url	否	string	分享链接(可带参数), 分享会在此url拼接上参数(share_type: 1 大奖分享 2 创房, room_id: 房间id),玩家可将该链接分享至各大社交渠道，实现用户裂变

launch_params可选参数

参数	必填	类型	描述

返回体:


```
{
    "msg": "success",
    "data": {
        "game_url": "https://example.com/start/index.html?game_id=MTIzNAo=&api_game=aHR0cHM6Ly9leGFtcGxlLWdhbWUuY29tCg==&api_game_token=dG9rZW4K&xxx=yyyy"
    },
    "code": 1001
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
game_url	是	body	string	游戏地址

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100214	400	无效的游戏ID
100202	404	游戏不存在
100213	404	玩家不存在

示例:

Curl

```
curl --location 'http://example.com/v1/launch_game' \
--header 'Content-Type: application/json' \
--data '{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "fffffffffff",
    "agent_id": 1
}'
```

获取游戏标签

请求路径:

```
/v1/game_labels
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述

返回体:

```
{
    "data": [
        {
            "value": 1,
            "index": "欧美系列"
        },
        {
            "value": 2,
            "index": "东方系列"
        },
        {
            "index": "水果系列",
            "value": 3
        }
    ],
    "code": 1001,
    "msg": "success"
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述

返回错误:

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/game_labels' \
--header 'Content-Type: application/json' \
--data '{}'
```

获取游戏品牌组

通过该功能可以获取游戏品牌组列表

请求路径:

```
/v1/gamebrandgroup/list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述

返回体:


```
{
    "code": 1001,
    "msg": "success",
    "data": [
        {
            "gbg_id": 6,
            "name": "zy",
            "num": 0
        },
        {
            "gbg_id": 5,
            "name": "test5",
            "num": 0
        },
        {
            "gbg_id": 4,
            "name": "test4",
            "num": 0
        },
        {
            "gbg_id": 3,
            "name": "test3",
            "num": 0
        },
        {
            "gbg_id": 2,
            "name": "test2",
            "num": 0
        },
    ]
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
gbg_id	是	body	int64	品牌组ID
name	是	body	string	品牌组名称
num	是	body	string	游戏数量

返回错误:

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/gamebrandgroup/list' \
--header 'Content-Type: application/json' \
--data '{}'
```

采集游戏记录

> **Warning**
>
> 此接口有 1分钟 延迟，每秒2次 请求限制

请求路径:

```
/v1/query_order
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "seq": 7388852,
    "page": 0,
    "limit": 2,
    "begin_time": 0,
    "end_time": 0,
    "type_id": [],
    "lottery_game_order": 0,
    "reverse": true,
    "need_count": true,
    "agent_id": 0
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
seq	是	body	int64	请求偏移量，首次可携带-1, 后续请求使用返回列表中第一条或最后一条数据中的 `seq` 参数
begin_time	否	body	int64	开始时间， unix时间戳(10位)
end_time	否	body	int64	结束时间，unix时间戳(10位)
page	否	body	int32	当前页数
limit	否	body	int32	每页数量,默认3000,最多返回5000条数据, >5000最多返回5000条数据(不足5000返回所有数据)， <=0返回最多返回3000条数据(不足3000返回所有数据)
type_id	否	body	[]int64	游戏类型列表,默认所有游戏类型
lottery_game_order	否	body	int32	注单类型, 默认为 `0` 电子游戏
agent_id	否	body	int64	代理人ID, 如果非 `0` 则返回所在代理人的注单列表
reverse	否	body	bool	返回注单列表是否反转， 默认从大到小返回
need_count	否	body	bool	是否需要统计总数量，此操作会比较耗时

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Betlog.GetBetlogs",
    "doc": "",
    "data": {
        "err_msg": "",
        "total": 0,
        "list": [
            {
                "seq": 7388854,
                "order_id": "72765775810650114",
                "round_id": "72765775810650113",
                "user_id": "45584981873872897",
                "user_name": "cary",
                "sn": 1,
                "agent_id": 1,
                "all_bets": 0,
                "eff_dama": 0,
                "jp_contri": 0,
                "jp_bonus": 0,
                "all_bonus": 0,
                "win_lose": 0,
                "bullet_count": 0,
                "type1_id": 4,
                "type2_id": 10,
                "type3_id": 2,
                "game_id": 1004501,
                "currency_id": 1,
                "currency": {
                    "id": 1,
                    "name": "BRL",
                    "code": "BRL",
                    "icon": "https://cdn-api-pc.ydev.app/currency/7eeb95b4-b66c-4d95-9bc0-84325342a266.png?v=1680513115",
```

```
                    "point": 3,
                    "symbol": "",
                    "coin_rate": 0
                },
                "end_time": 1684982340,
                "payout_time": 1684982340,
                "order_state": 2,
                "payout_time_at": "2023-05-24T22:39:00-04:00",
                "end_time_at": "2023-05-24T22:39:00-04:00",
                "created_at": "2023-05-24T22:39:00-04:00",
                "updated_at": "2023-05-24T22:39:00-04:00",
                "order_id_str": "72765775810650114",
                "round_id_str": "72765775810650113",
                "user_id_str": "45584981873872897",
                "currency_name": "",
                "freegame": 1
            }
        ]
    }
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
seq	是	body	int64	采集序号
order_id	是	body	int64	注单ID
round_id	是	body	int64	局ID
user_id	是	body	int64	玩家ID
user_name	是	body	string	玩家名称
sn	是	body	int64	总社ID
agent_id	是	body	int64	代理ID
all_bets	是	body	int64	押注金额, 单位: 货币最小单位
eff_dama	是	body	int64	有效投注金额, 单位: 货币最小单位
jp_contri	是	body	int64	彩池贡献金额, 精度: +4
jp_bonus	是	body	int64	彩池中奖金额, 单位: 货币最小单位
all_bonus	是	body	int64	派彩金额, 单位: 货币最小单位
win_lose	是	body	int64	输赢金额, 单位: 货币最小单位
type1_id	是	body	int64	游戏分类
type2_id	是	body	int64	游戏类型
type3_id	是	body	int64	游戏玩法类型
game_id	是	body	int64	游戏ID
currency_id	是	body	int64	货币ID
currency	是	body	object	货币详情
end_time	是	body	int64	派彩时间, unix时间戳(10位)
payout_time	是	body	int64	投注时间, unix时间戳(10位)
order_state	是	body	int32	请参考注单状态中的整形标识符

返回错误:

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/query_order' \
--header 'Content-Type: application/json' \
--data '{
    "seq": -1,
    "limit": 10,
    "begin_time": 0,
    "end_time": 0,
    "sn": 1,
    "type_id": 0,
    "lottery_game_order": 0
}'
```

FQA:

- **Q:** 如何同步游戏记录?
  A: 使用 `seq` 方案同步游戏记录。首次请求时，`seq=-1`，后续请求传递上次请求返回的最大 `seq` 值，`page` 和 `limit` 不传，确保数据完整同步。
- **Q:** 为什么不使用 `page` + `size` + `time` 方案同步游戏记录?
  A: 如果游戏记录入库失败，系统会重新入库。通过时间过滤可能会导致漏拉数据，因为重新入库的数据的时间可能已经被拉取过，导致这些记录无法被重新同步。

采集数据汇总

> **Warning**
>
> 1、数据汇总统计时区为美东时间(-04:00)
>
> 2、接入方需延时2个小时拉取数据，即美东时间05:00:00之后才能拉取到02:00:00~02:59:59的汇总统计数据，如此类推
>
> 3、按请求的开始时间和结束时间，返回按小时统计的对应数据 如[开始时间:2025-01-01 00:02:45~~结束时间：2025-01-01 03:45:45]，即统计2025-01-01 00、01、02、03点对应数据

请求路径:

```
/v1/query_game_order_sum
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "begin_time": 1704067200,
    "end_time": 1704070800,
    "page_size": 20,
    "page_index": 1
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
begin_time	是	body	int64	开始时间， unix时间戳(10位)
end_time	是	body	int64	结束时间，unix时间戳(10位)
page_size	否	body	int32	每页数量,默认20,最多返回100条数据
page_index	否	body	int32	页码,默认1

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "YKeDkBsoDyVQIPkVblKJrrhjNSsvLPan",
    "request_method": "api.analysis.Analysis.OrderSummaryTotal",
    "provider": "srv_analysis",
    "doc": "",
    "data": {
    "err": "",
    "list": [
        {
            "time": 2025101923,
            "all_bets": 650200,
            "eff_dama": 650200,
            "all_bonus": 595800,
            "win_lose": -54400,
            "jp_contri": 6502000,
            "jp_bonus": 0,
            "order_sum": 9,
            "sn": 1,
            "currency": "PHP"
        }
  ]
}
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
time	是	body	int64	时间格式：2025101923
sn	是	body	int64	总社ID
all_bets	是	body	int64	押注金额, 单位: 货币最小单位
eff_dama	是	body	int64	有效投注金额, 单位: 货币最小单位
jp_contri	是	body	int64	彩池贡献金额, 精度: +4
jp_bonus	是	body	int64	彩池中奖金额, 单位: 货币最小单位
all_bonus	是	body	int64	派彩金额, 单位: 货币最小单位
win_lose	是	body	int64	输赢金额, 单位: 货币最小单位
currency	是	body	object	货币详情
order_sum	是	body	int64	笔数

返回错误:

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/query_game_order_sum' \
--header 'Content-Type: application/json' \
--data '{
    "begin_time": 0,
    "end_time": 0
}'
```

获取游戏详情内嵌

请求路径:

```
/v1/game_order_items
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "order_number": "123456",
    "language": "en",
    "lottery_game_order": 0
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
order_number	是	body	string	注单ID, 对应采集游戏记录中返回的 `order_id`
language	否	body	string	语言,默认英语
lottery_game_order	否	body	int32	注单类型, 默认为 `0` 电子游戏

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": {
        "item_url": "https://example.com/#/gameDetails?token=dG9rZW4K"
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
item_url	是	body	string	页面地址

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
103201	404	注单不存在

示例:

Curl

/game_version_md5_zip_list

```
curl --location 'http://example.com/v1/game_order_items' \
--header 'Content-Type: application/json' \
--data '{
    "order_number": "1649656234593"
}'
```

获取原生包版本

请求路径:

```
/v1/game_version_md5_zip_list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "game_id": [100100,100344],
    "plat_form": "api"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
game_id	是	body	array	游戏id列表
plat_form	否	body	string	默认是获取app的版本原生包，默认值:`app`， 可选值：`app`、`api`

返回体:

```
{
  "code": 1001,
  "data": {
    "game_1008201": [
      {
        "key": "api_gc-zy-game_1008201-1.8.0",
        "name": "game_1008201",
        "full_name": "api_gc-zy-game_1008201",
        "version": "1.8.0",
        "md5": "9d2ba45fe3c45d35ea7177283ee74020",
        "zip": "api_gc-zy-game_1008201.zip",
        "package": "api_gc-zy-game_1008201-1.12.0-4-1682241359",
        "game_id": 1008201,
        "base": "1.12.0",
        "game_core": "1.12.0",
        "slot": "1.12.0",
        "hunter": "",
        "arcade": "",
        "lottery": "",
        "single": "",
        "bingo": ""
      },
    ]
  },
  "msg": "success"
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
key	是	body	string	版本key
name	是	body	string	版本的标识
full_name	是	body	string	版本的全名标识
version	是	body	string	版本号
md5	是	body	string	版本的md5
zip	是	body	string	版本的资源包
game_id	否	body	int64	游戏id
base	否	body	string	游戏的base库版本号
game_core	否	body	string	游戏的game_core库版本号
slot	否	body	string	游戏的slot库版本号
arcade	否	body	string	游戏的arcade库版本号
lottery	否	body	string	游戏的lottery库版本号
single	否	body	string	游戏的single库版本号
bingo	否	body	string	游戏的bingo库版本号

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/game_version_md5_zip_list' \
--header 'Content-Type: application/json' \
--data '{
    "game_id": [1008201,1000142],
}'
```

启动游戏大厅

请求路径:

```
/v1/hall/launch
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "language": "en",
    "lobby_code": "hall_2334433",
    "third_name": "xxx",
    "agent_id": 123
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
language	否	body	string	语言,默认英语
lobby_code	是	body	string	游戏大厅code
third_name	是	body	string	第三方用户名称, 如果为非试玩则需要携带此字段且该玩家已通过接口创建玩家创建
agent_id	是	body	int64	代理ID

返回体:

```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "eAkxgB1BNjoXPOMn8UyJN45ekhEajk9A",
    "request_method": "openapi.game.Hall.Launch",
    "provider": "srv_admin",
    "doc": "",
    "data": {
        "launch_url": "https://cdn-api-game-lobby.ydev.app/test/dating/1.0.0/index.html?
token=V0VCQVBQIGV5SmhiR2NpT2lKSVV6STFOaUlzSW5SNWNDSTZJa3BYVkNKOS5leUpTWlhFaU9uc2laR1ZoYkdWeVgybGtJam94TENKaFoyVnVZM2xmYVdRaU9qRXNJbUZuWlc1MFgybGtJam93TENKd2J
HRjVaWEpmYVdRaU9qTTJOemcyTlRFM09UVXhNRGs0T0RneExDSnNZWFZ1WTJoZmNtVnhJanA3SW1selgyWnlaV1VpT21aaGJITmxMQ0owYUdseVpGOXVZVzFsSWpvaWVYVjBkWFJsYzNRaUxDSm5ZVzFsWDJs
a0lqb3dMQ0pzWVc1bmRXRm5aU0k2SWlJc0ltRm5aVzUwWDJsa0lqb3dMQ0p5WlhSMWNtNWZkWEpzSWpvaUlpd2lkR1Z5YldsdVlXd2lPaUlpTENKbGVIUnlZU0k2Ym5Wc2JDd2ljMjRpT2pFc0lteGhkVzVqY
UY5d1lYSmhiWE1pT201MWJHd3NJbkpsWTJoaGNtZGxYM1Z5YkNJNklpSjlMQ0ptYVhKemRGOXdZWEpsYm5SZmFXUWlPakI5TENKaGRXUWlPaUl6TmpjNE5qVXhOemsxTVRBNU9EZzRNU0lzSW1WNGNDSTZNVF
k1TnpnM05ESTNOU3dpYW5ScElqb2lPVE00TUdObFptRXROMkZsWVMwME5ERTFMVGsyWkdNdFpUVTBaVGc1TWpsa05qTTVJaXdpYVdGMElqb3hOamszTnpnM09EYzFMQ0pwYzNNaU9pSndiR0YwWm05eWJTNXp
jblpmWVdSdGFXNHVaR1YySWl3aWJtSm1Jam94TmprM056ZzNPRGMxTENKemRXSWlPaUpYUlVKQlVGQWlmUS5oaktMakg1LWhYczh5eEFJMEtHdEZTU2NWbTJOSmI1aXFDcERuNjZqTDdZ&language=&lobby
_code="
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
launch_url	是	body	string	游戏大厅地址

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
1005	404	系统维护中
1001017	404	游戏大厅code无效
100213	404	玩家不存在

示例:

Curl

```
curl --location 'http://example.com/v1//v1/hall/launch' \
--header 'Content-Type: application/json' \
--data '{
    "lobby_code": "test2",
    "language": "en",
    "player_name": "fffffffffff",
    "agent_id": 1
}'
```

游戏大厅列表

请求路径:

```
/v1/hall/list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "language": "en"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
language	是	body	string	语言,默认英语

返回体:

```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "EZHof6ucQpyVubYmZcPc4KZiQ1hCDVc5",
    "request_method": "openapi.game.Hall.GetHallList",
```

```
    "provider": "srv_admin",
    "doc": "",
    "data": {
        "page": 0,
        "size": 0,
        "total": 0,
        "list": [
            {
                "id": 56,
                "hall_type": 3,
                "name": "测试大厅模板2",
                "game_num": 0,
                "code": "大厅模板1",
                "url": "/test/dating",
                "loading": {
                    "language": "en",
                    "image_h": "https://cdn-api-pc.ydev.app/game_hall/20231011/47486a44-be94-4e52-bc7b-a28977086392_h.png?v=1697005282",
                    "image_s": "https://cdn-api-pc.ydev.app/game_hall/20231011/89d29969-0eab-4d92-964d-624fc60481e7_s.png?v=1697005285",
                    "name": "测试大厅模板2",
                    "icon": "https://cdn-api-pc.ydev.app/game_hall/20231011/a989c9a2-b58a-4a5d-907f-0e8bda976f57_icon.png?v=1697005279"
                },
                "version": "1.0.0",
                "hall_id": 56
            },
            {
                "id": 61,
                "hall_type": 3,
                "name": "22eee",
                "game_num": 0,
                "code": "大厅模板2",
                "url": "/test",
                "loading": {
                    "language": "en",
                    "image_h": "https://cdn-api-pc.ydev.app/game_hall/20231011/6fcdadc5-9b8c-4057-a410-689586557edd_h.png?v=1697005446",
                    "image_s": "https://cdn-api-pc.ydev.app/game_hall/20231011/d47796d5-f920-465a-a6fb-53b25dfbba95_s.png?v=1697005449",
                    "name": "22eee",
                    "icon": "https://cdn-api-pc.ydev.app/game_hall/20231011/a02d556d-091f-4bd9-a9d1-132c7f690a6d_icon.png?v=1697005442"
                },
                "version": "1.2.0",
                "hall_id": 61
            }

        ]
    }
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
list	是	body	array	游戏大厅列表
id	是	body	int	游戏大厅id
hall_type	是	body	int	游戏大厅类型 1 棋牌大厅 2 捕猎大厅 3 slot大厅 4 街机大厅 5 综合大厅
name	是	body	string	游戏大厅名称
game_num	是	body	int	游戏数量
code	是	body	string	游戏大厅code
url	是	body	string	游戏大厅url
hall_id	是	body	int	游戏大厅id
version	是	body	string	游戏大厅版本号
loading	是	body	object	游戏大厅loading图和icon设置
loading.language	是	body	string	游戏大厅loading图多语言，默认是en
loading.image_h	是	body	string	游戏大厅loading横图
loading.image_s	是	body	string	游戏大厅loading竖图
loading.icon	是	body	string	游戏大厅icon

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
1005	404	系统维护中

示例:

Curl

```
curl --location 'http://example.com/v1//v1/hall/launch' \
--header 'Content-Type: application/json' \
--data '{
    "language": "en"
}'
```

获取押分

请求路径:

```
/v1/free_give/get_bet
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "game_id": 12345,
    "currency_symbol": "BTC"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
game_id	是	body	int64	游戏id
currency_symbol	是	body	string	货币符号

返回体:

```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "betting_list": []
        }
    ]
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
betting_list	是	body	array	押分列表

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/free_give/get_bet' \
--header 'Content-Type: application/json' \
--data '{
    "betting_list": [1,2,3]
}'
```

创建赠送免费

请求路径:

```
/v1/free_give/create_free_give
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "game_id": 12345,
    "currency_symbol": "BTC",
    "free_give_type": 20,
    "validity_time": 10,
    "remark": "备注",
    "account_data": "用户账号1,用户账号2",
    "point_list": [
      {
        "single_betting": 1,
        "free_rounds": 10,
        "name": "小游戏奖励key"
      }
    ]
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
game_id	是	body	int64	游戏id
currency_symbol	是	body	string	货币符号
free_give_type	是	body	int64	赠送免费类型:20 固定押分普通旋转 21 灵活组合普通旋转 22 小游戏免费转
validity_time	是	body	int64	有效期 单位小时
remark	否	body	string	备注
account_data	是	body	array	用户账号列表用英文逗号分割
point_list	是	body	array	押分数据: single_betting 单局押分等于-1 表示触发时押分, 等于-2表示游戏最小押分, free_rounds 免费局数, name 只有小游戏免费转才会传这个值

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "account_list": []
        }
    ]
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
account_list	是	body	array	成功创建的用户账号列表

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/free_give/create_free_give' \
--header 'Content-Type: application/json' \
--data '{
    "account_list": [1,2,3]
}'
```

获取赠送游戏列表

请求路径:

```
/v1/free_give/get_game_list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "language": "en"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
language	是	body	string	语言

返回体:

```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "list": [
              {
                "game_id": "1",
                "game_name": "接财神",
                "mini_game_list": [
                  {
                    "name": "小游戏的奖励KEY",
                    "multiple": "小游戏倍数"
                  }
                ]
              }
            ]
        }
    ]
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
list	是	body	array	游戏列表

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/free_give/get_game_list' \
--header 'Content-Type: application/json' \
--data '{
    "list": [
        {
            "game_id": 1,
            "game_name": "gameName",
            "mini_game_list": [
                {
                    "name": "奖励key",
                    "multiple": "奖励倍数"
                }
            ]
        }
    ]
}'
```

获取赠送免费记录

请求路径:

```
/v1/free_give/query_order
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "language": "en",
    "begin_time": "1747624874",
    "end_time": "1747624874",
    "brand_group_id": "1",
    "game_id": "111",
    "user_search_data": "用户账号或uid",
    "free_give_type": 20,
    "page_size": 1,
    "page_index": 10,
    "currency_symbol": "BTC",
    "free_give_status": 1
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
language	否	body	string	语言
begin_time	否	body	int64	开始时间， uninx时间戳(10位)
end_time	否	body	int64	结束时间，unix时间戳(10位)
brand_group_id	否	body	int64	品牌组id
game_id	否	body	int64	游戏id
user_search_data	否	body	string	用户账号或uid
free_give_type	否	body	int64	赠送免费类型:20 固定押分普通旋转 21 灵活组合普通旋转 22 小游戏免费转
page_size	否	body	int64	当前页数
page_index	否	body	int64	每页数量,默认3000,最多返回5000条数据, >5000最多返回5000条数据(不足5000返回所有数据)， <=0返回最多返回3000条数据(不足3000返回所有数据)
currency_symbol	是	body	string	货币符号
free_give_status	否	body	int64	赠送免费状态:1 未开始 2 进行中 3 已结束 4 被移除 5 已失效

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "total_count": 1,
            "list": [
              {
                "order_id_str": "72765775810650114",
                "round_id_str": "72765775810650113",
                "game_name": "game",
                "user_name": "test",
                "currency_symbol": "BTC",
                "free_give_type": 20,
                "all_bets": 1,
                "all_bonus": 1,
                "win_lose": 1,
                "rate_of_return_str": 1.1,
                "create_time": 111223123211,
                "free_give_status": 1,
                "validity_time": 11,
                "round_id_list": [
                  "12312321",
                  "2134214124"
                ]
              }
            ]
        }
    ]
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
order_id_str	是	body	string	采集序号
round_id_str	是	body	string	记录号
game_name	是	body	string	游戏名称
user_name	是	body	string	用户名
currency_symbol	是	body	string	货币符号
free_give_type	是	body	int64	赠送免费类型:20 固定押分普通旋转 21 灵活组合普通旋转 22 小游戏免费转
all_bets	是	body	int64	押注金额, 单位: 货币最小单位
all_bonus	是	body	int64	派彩金额, 单位: 货币最小单位
win_lose	是	body	int64	输赢金额, 单位: 货币最小单位
rate_of_return_str	是	body	string	回报率
create_time	是	body	int64	创建时间
free_give_status	是	body	int64	赠送状态 1未开始 2进行中 3 已结束 4 被移除 5 已失效
validity_time	是	body	int64	有效时间
round_id_list	是	body	array	游戏局数列表

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/free_give/query_order' \
--header 'Content-Type: application/json' \
--data '{
    "total":  0,
    "list": [
          {
            "order_id_str": "72765775810650114",
            "round_id_str": "72765775810650113",
            "game_name": "game",
            "user_name": "test",
            "currency_symbol": "BTC",
            "free_give_type": 20,
            "all_bets": 1,
            "all_bonus": 1,
            "win_lose": 1,
            "rate_of_return_str": 1.1,
            "create_time": 111223123211,
            "free_give_status": 1,
            "validity_time": 11,
            "round_id_list": [
              "12312321",
              "2134214124"
            ]
          }
    ]
}'
```

获取对押监控汇总记录

请求路径:

```
/v1/pressure/summary
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "user_search_data": "用户账号或uid",
    "page": 1,
    "page_size": 10,
    "coin_type": "BTC"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
user_search_data	否	body	string	用户账号或uid
page	否	body	int64	当前页数
page_size	否	body	int64	每页数量,默认3000,最多返回5000条数据, >5000最多返回5000条数据(不足5000返回所有数据)， <=0返回最多返回3000条数据(不足3000返回所有数据)
coin_type	是	body	string	货币符号

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Game.Search",
    "doc": "",
    "data": [
        {
            "total": 1,
            "list": [
              {
                "agent_id": 1234,
                "user_id": 72765775810650113,
                "one_vs_one_count": 1,
                "one_vs_n_count": 1,
                "coin_type_name": "BTC",
                "total_bet": 20,
                "total_winlose": 1,
                "agent_name": "agent_name",
                "user_name": "test"
              }
            ]
        }
    ]
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
agent_id	是	body	int64	代理id
user_id	是	body	int64	用户id
one_vs_one_count	是	body	int64	1对1游戏次数
one_vs_n_count	是	body	int64	1对n游戏次数
coin_type_name	是	body	string	货币名称
total_bet	是	body	int64	总押注金额, 单位: 货币最小单位
total_winlose	是	body	int64	总输赢金额, 单位: 货币最小单位
agent_name	是	body	string	代理名称
user_name	是	body	string	用户名

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'http://example.com/v1/pressure/summary' \
--header 'Content-Type: application/json' \
--data '{
    "total":  0,
    "list": [
          {
            "agent_id": 1234,
            "user_id": 72765775810650113,
            "one_vs_one_count": 1,
            "one_vs_n_count": 1,
            "coin_type_name": "BTC",
            "total_bet": 20,
            "total_winlose": 1,
            "agent_name": "agent_name",
            "user_name": "test"
          }
    ]
}'
```

Models

游戏分类

ID	分类
4	电子游戏

游戏类型

ID	类型
10	旋转游戏
11	捕猎
12	桌游
13	彩票
15	BINGO
16	棋牌
17	柏青哥
20	旋转游戏哈希
21	捕猎哈希
22	桌游哈希
23	彩票哈希
26	棋牌哈希
30	旋转游戏时间戳
31	捕猎时间戳
32	桌游时间戳
40	旋转游戏币安
41	捕猎币安
42	桌游币安
50	旋转游戏创世哈希
51	捕猎创世哈希
52	桌游创世哈希
56	棋牌创世哈希
62	桌游波场
63	彩票波场

注单类型

整形标识符	字符串标识符	类型
0	Electronic	电子游戏
1	Lottery	彩票游戏

注单状态

整形标识符	字符串标识符	描述
0	UNKNOWN	未知
1	BETED	押分(未开奖)
2	SETTLED	结算(已开奖)
3	ROLLBACK	退还,目前只针对彩票游戏和一元购游戏

游戏玩法类型

游戏类型ID	玩法类型ID	玩法类型
10	0	未知
10	1	普通旋转
10	2	免费旋转
10	3	重转
10	4	小游戏
10	5	JP大奖
10	6	世界BOSS
10	7	购买免费游戏
10	8	触发奖励事件
10	9	收集奖励金币
10	10	翻倍游戏
10	11	事件奖励金币
10	12	特色免费
11	1	普通
11	2	积分抽奖
11	3	刮刮乐
11	4	猜大小
11	5	jackpot
12	1	押分

货币详情

参数	类型	描述
id	int	货币ID
name	string	货币名称
code	string	货币码
icon	string	货币图标
point	string	货币精度

1.2.3 玩家相关

> **Note**
>
> 此文档中玩家加减钱为转账模式，如需免转钱包模式请参考免转钱包

> **关于货币**
>
> - 支持所有法币及虚拟货币,只支持两位精度展示,如btc需要乘以1000000 例：1 BTC = 1000000.00 uBTC
> - 对于弱势货币，可按具体需求配置对应比值

获取玩家金币

请求路径:

```
/v1/balance_get
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "third_name": "xxx",
    "coin_type": -1,
    "agent_id": 123
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
third_name	是	body	string	玩家账号
coin_type	否	body	int64	货币列表中的整形标识符， 小于0： 代表获取所有货币余额, 等于0: 代表玩家最近一次使用的货币(启动游戏时所携带的货币)
agent_id	是	body	int64	代理人ID

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": {
        "USD": 100,         // 精度为2，代表用户余额是1.00美元
        "balance": 10000    // 精度为2，代表玩家最近一次使用的货币余额是100.00
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
balance	是	body	int64	玩家余额，最近一次使用的货币的余额, 单位：该货币的最小单位
USD	否	body	int64	USD所在货币的余额, 单位：该货币的最小单位

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100213	404	玩家不存在

示例:

Curl

```
curl --location 'http://example.com/v1/balance_get' \
--header 'Content-Type: application/json' \
--data '{
    "third_name": "fffffffffff"
}'
```

存取玩家金币

同一个 `biz_id` 交易幂等

请求路径:

```
/v1/balance_transfer
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "third_name": "xxx",
    "amount": 100,      // 精度为2, 代表1.00
    "biz_id": "bid-123",
    "coin_type": 0,
    "agent_id": 123
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
third_name	是	body	string	玩家账号
amount	是	body	int64	加减金额，单位为当前货币的最小单位, >0加金额, <0减金额
biz_id	是	body	string(50)	订单ID, 同一个订单ID交易幂等
coin_type	否	body	int64	货币列表中的整形标识符
agent_id	是	body	int64	代理人ID

返回体:


```
{
    "code": 1001,
    "success": true,
    "status_code": 200,
    "system": 0,
    "msg": "Success",
    "prompt": "",
    "request_id": "",
    "request_method": "openapi.game.Player.WalletTransfer",
    "doc": "",
    "data": {
        "amount": 10000,            // 精度为2，代表100.00
        "balance_before": 6027785,  // 精度为2，代表60277.85
        "balance_after": 6037785    // 精度为2，代表60377.85
    }
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
amount	是	body	int64	加减金额，单位为当前货币的最小单位
balance_before	是	body	int64	加减 前 的余额，单位为当前货币的最小单位
balance_after	是	body	int64	加减 后 的余额，单位为当前货币的最小单位

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100905	412	桌游游戏中无法取款
100221	409	交易ID(biz_id)重复

示例:

Curl

```
curl --location 'http://example.com/v1/balance_transfer' \
--header 'Content-Type: application/json' \
--data '{
    "third_name": "fffffffffff",
    "amount": 100,
    "biz_id": "da68d00b-a251-45f7-a0b7-172dba678c61"
}'
```

获取存取金币结果

请求路径:

```
/v1/transfer
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "biz_id": "xxx123dsfsdfsdfsa"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
biz_id	是	body	string	交易代碼需要是唯一值(允許字元:a-z A-Z 0-9 _-), 10-50个字符

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": {
      "biz_id": "1f83a95d-fd1a-4712-8d07-f51304392a4b",
      "third_name": "player1",
      "amount": 9,
      "coin_type": 7,
      "balance_before": 1,
      "balance_after": 10
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
biz_id	是	body	string	交易代碼需要是唯一值(允許字元:a-z A-Z 0-9 _-), 10-50个字符
third_name	是	body	string	玩家账号
amount	是	body	int64	加减金额，单位为当前货币的最小单位
coin_type	是	body	int64	货币列表中的整形标识符
balance_before	是	body	int64	加减 前 的余额，单位为当前货币的最小单位
balance_after	是	body	int64	加减 后 的余额，单位为当前货币的最小单位

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
103403	200	biz_id不存在

示例:

Curl

```
curl --location 'http://example.com/v1/transfer' \
--header 'Content-Type: application/json' \
--data '{
    "biz_id": "fffffffffffff"
}'
```

创建玩家

可以重复创建，只会创建一次，如果玩家已存在则返回已存在玩家信息

请求路径:

```
/v1/create_user
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "third_name": "xxx",
    "agent_id": 123,
```

```
    "user_type": 1
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
third_name	是	body	string(50)	玩家账号, **全局可能重复**, 每个总社不能重复
agent_id	是	body	int64	代理人id（厅主id）
user_type	否	body	int32	玩家类型, 1-真实用户, 2-系统用户, 默认为 `1`

由调用方提供账号，不同总社也会出现相同，所以不以此为唯一依据。

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": {
        "open_id": 90751924103860225
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
open_id	是	body	int	玩家账号ID, openid是玩家唯一标识，玩家注册后，请保留openid，供后续接口使用。

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100216	409	玩家已存在
102201	404	代理不存在

示例:

踢出玩家

请求路径:

```
/v1/kick_user
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "third_name": "xxx",
    "agent_id": 123
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
third_name	是	body	string	玩家账号
agent_id	是	body	int64	代理人ID

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": ""
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100213	404	玩家不存在

示例:

==="Curl"

```
```bash
curl --location 'http://example.com/v1/kick_user' \
--header 'Content-Type: application/json' \
--data '{
    "third_name": "fffffffffff"
```
```

```
}'
```

是否锁定在游戏中

请求路径:

```
/v1/user/is_locked_game
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "third_name": "xxx",
    "agent_id": 1
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
third_name	是	body	string	玩家账号
agent_id	是	body	int64	代理人ID

返回体:

```
{
    "code": 1001,
    "success": true,
    "data": {
        "flag": 0,
        "game_id": 0
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
game_id	是	body	int62	游戏ID
flag	是	body	int32	状态: 1 锁定 0 未锁定

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
100213	404	玩家不存在

示例:

===="Curl"

```bash
curl --location 'http://example.com/v1/user/is_locked_game' \
--header 'Content-Type: application/json' \
--data '{
    "third_name": "xxx",
    "agent_id": 1
}'
```

1.2.4 代理人相关

创建代理人

请求路径:

```
/v1/agent_new
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{
    "agent_account": "xx",
    "agent_name": "xx",
    "currency_symbol":"BRL"
}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述
agent_account	是	body	string(50)	代理人账号，统一总社下需保证唯一
agent_name	是	body	string(30)	代理人名称
currency_symbol	否	body	string	货币代码

返回体:

```
{
    "code": 1001,
    "msg": "success",
    "data": {
        "agent_id": 111
    }
}
```

返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
agent_id	是	body	int64	代理人ID

返回错误:

公共错误码请参考公共错误码

错误码	HTTP状态码	描述
1001	200	成功
102204	409	代理人已存在

示例:

Curl

```
curl --location 'https://example.com/v1/agent_new' \
--header 'Content-Type: application/json' \
--data '{
    "agent_account": "testttJZdsn1OsaaWQaM11Ss11yjFO1",
    "currency_codes": ["BRL"]
}'
```

1.2.5 系统相关

获取货币列表

请求路径:

```
/v1/currency_list
```

请求方式:

```
POST
```

请求头:

请求头	必填	类型	描述

请求体:

```
{}
```

请求描述:

公共请求参数请参考公共请求

参数	必填	位置	类型	描述

返回体:


```
{
    "code": 1001,
    "msg": "success",
    "data": {
        "list": [
            {
                "Id": 2,
                "name": "BTC",
                "code": "BTC",
                "symbol": "BTC",
                "country": "c",
                "point": 8,
                "exchange_rate": 0,
                "display_rate": 0,
                "icon": "/currency/0f322156-4da4-4b47-886c-0ccddea437c1.png?v=1681197294",
                "creator_id": 1,
                "updator_id": 12,
                "created_at": "1969-12-31T20:00:00-04:00",
                "updated_at": "1969-12-31T20:00:00-04:00",
                "ratio": 0
            }
        ]
    }
}
```


返回头:

返回头	必带	类型	描述

返回描述:

公共返回参数请参考公共请求

参数	必带	位置	类型	描述
Id	是	body	int64	货币整形标识符
code	是	body	string	货币字符串标识符
point	是	body	int32	货币精度

返回错误:

错误码	HTTP状态码	描述
1001	200	成功

示例:

Curl

```
curl --location 'https://example.com/v1/currency_list' \
--header 'Content-Type: application/json' \
--data '{}'
```

1.2.6 免转钱包

免转钱包基本配置

- 配置免转钱包域名
- 选择免转钱包接口版本：`v2`
- 配置签名key

> **关于余额**
>
> - 接口中请求或返回的金额均为所在货币的最小单位的值, 游戏里面显示的金额单位为 `元` ,货币精度可参考获取货币列表中返回的 `货币精度` 字段
> - 支持所有法币及虚拟货币,只支持两位精度展示,如btc需要乘以1000000 例：1 BTC = 1000000.00 uBTC
> - 对于弱势货币，可按具体需求配置对应比值

玩家身份验证

我方在调用钱包接口时会把用户账号统一放在http请求头Header中的Account字段，如：

```
[
    {Content-Type: <<"application/json">>},
    {Account: <<"123456">>}
]
```

签名方式

sign签名规则：请求参数除了sign参数除外，其他参数按参数名升序排序，key=value 以 & 拼接，最后用sign=SignKey加密串，再以md5方式进行加密，最后将所有字母转为大写

> **Tip**
>
> 签名的时候需要对接口Body中所有的参数进行签名，否则签名校验时可能会失败，除了必要的字段之外，我方可能会在接口中添加可选字段

```
%% 请求钱包余额
sign_key: <<"testkey_1111111111">>
CoinCode: <<"BRL">>
GameId: 1000101
RoomType: <<"1">>
得到加密字符串: <<"coin_code=BRL&game_id=1000101&room_type=1&key=testkey_1111111111">>
md5加密得到: <<"ee4e60c1cd517f6af351c5a2ffc80ac7">>
UpperStr得到: <<"EE4E60C1CD517F6AF351C5A2FFC80AC7">>
最终请求body:
Body = #{
    coin_code => <<"BRL">>,
    game_id => 1000101,
    room_type => <<"1">>,
    sign => <<"EE4E60C1CD517F6AF351C5A2FFC80AC7">>
}
```

接口实现

> **Tip**
>
> 以下接口请求方式一律为 `POST` , 请求及响应头 `Content-Type` 为 `application/json`

通知进入游戏

请求路径:

```
/enter_game
```

请求示例:

```
{
    coin_code: "BRL",
    game_id: 1000101,
    timestamp: 1757559709,
    sign: "C61BD35BA2D99F08A43F4DCF2C613A4C"
}
```

请求描述:

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
timestamp	时间戳(秒)	int(64)	必须

返回描述

不关心返回内容

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	可选
data	数据	map	可选
msg	备注说明	string	可选

返回示例:

```
{
    code: 0,
    data: {},
    msg: "success"
}
```

请求钱包余额

请求路径:

```
/wallet_request
```

请求示例:

```
{
    coin_code: "BRL",
    game_id: 1000101,
    room_type: "1",
    sign: "C61BD35BA2D99F08A43F4DCF2C613A4C"
}
```

请求描述:

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
room_type	游戏倍场(玩法)	string(10)	必须

返回描述

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	必须
data	数据(成功时必须返回数据，详见 data说明)	map	可选
msg	备注说明	string	可选

data说明:

名称	说明	类型	状态
chips	玩家余额	int(64)	必须

返回示例:

```
{
    code: 0,
    data: {
        chips: 10000    // 精度为2，游戏内用户余额100.00
    },
    msg: "success"
}
```

下注/加注

请求路径:

```
/wallet_bet
```

请求示例:

```
{
    coin_code: "BRL",
    game_id: 1000101,
    room_type: "1",
    order_id: "2342342334123",
    min_amount: 1000,   // 精度为2，代表最小下注额10.00
    amount: 10000,      // 精度为2，代表最大下注额100.00
    bet_id "23455534666666",
    sign: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

请求描述:

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
room_type	游戏倍场(玩法)	string(10)	必须
order_id	订单id(与结算接口的 `order_id` 对应)	string(50)	必须
min_amount	最小下注(单次下注最小额度（可以为0），玩家余额必须大于等于最小下注额才可完成本次请求)	int(64)	必须
amount	最大下注(在捕鱼类游戏中，每次请求可能包含多个下注的额度，若玩家余额小于最大下注额，则扣除玩家所有余额并完成本次请求)	int(64)	必须
bet_id	每次下注时生成的唯一标识	string(50)	必须

返回描述

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	必须
data	数据(成功时必须返回数据，详见 `data说明`)	map	可选
msg	备注说明	string	可选

data说明:

名称	说明	类型	状态
chips	本次下注后玩家剩余余额	int	必须
deduction_amount	本次扣除下注额	int	必须

```
扣除逻辑:
如果`amount`字段的值等于`min_amount`字段的值:
  玩家余额大于等于`amount`的值，则扣除`amount`等值的金额，否则下注失败
如果`amount`字段的值大于`min_amount`字段的值:
  玩家余额大于等于`amount`的值，则扣除`amount`等值的金额;
  玩家余额小于`amount`的值且大于等于`min_amount`字段的值，则扣除玩家所有余额;
  否则下注失败
需要注意的是下注的`order_id`字段与结算的`order_id`字段对应，slot旋转游戏的免费旋转和重转因为不会扣除用户余额，所以不会回调下注接口，只会回调结算接口，并且每次结算的`order_id`不同
```

结算金币

请求路径:

```
/wallet_settlement
```

请求示例:

```
{
    coin_code: "BRL",
    game_id: 1000101,
    room_type: "1",
    order_id: "11123213213",
    round_id: "22223243445",
    jp_contri: 10000,         // 精度为6, 代表jp贡献为0.01
    jp_bonus: 100,            // 精度为2, 代表jp奖金为1.00
    back_amount: 1000,        // 精度为2, 代表回退金额为10.00
    details: [
        {
            index: 1,
            bet_id: "112312323233",
            amount: 10000,      // 精度为2, 代表本次结算的下注额为100.00
```

```
            win_amount: 20000,  // 精度为2, 代表获得奖金200.00
            coded_quantity: 10000,
            all_bets: 10000,
            is_finish: true
        }
    ],
    is_order_finish: true,
    settlement_id: "123334565",
    sign: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

请求描述:

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
room_type	游戏倍场(玩法)	string(10)	必须
order_id	订单id(与下注接口的 `order_id` 对应)	string(50)	必须
round_id	回合id	string(50)	必须
is_free	是否免费游戏	boolean	必须
jp_contri	本次的jp贡献值	int(64)	必须
jp_bonus	本次的jp奖金	int(64)	必须
back_amount	退回结算退回的本金	int(64)	必须
details	结算详情(详见 `details说明`)	list	必须
is_order_finish	订单是否结束	boolean	必须
settlement_id	每次结算生成的唯一标识	string(50)	必须

details说明:

- 在捕猎游戏中，会有多个map结构数据，每一个结构数据为每一发子弹的结算信息，
- 在非捕猎游戏中，一般只有一个map结构数据，即本次下注的结算数据
- map结构如下:

名称	说明	类型	状态
index	数据索引(从1开始自增)	int(32)	必须
bet_id	每次下注返回的唯一id	string(50)	必须
amount	本次结算的下注额	int(64)	必须
win_amount	本次结算的奖金	int(64)	必须
coded_quantity	有效打码(游戏记录用)	int(64)	必须
all_bets	总下注(游戏记录用)	int(64)	必须
is_finish	该下注是否结束完成	boolean	必须

返回描述:

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	必须
data	数据(成功时必须返回数据，详见 data说明)	map	可选
msg	备注说明	string	可选

data说明:

名称	说明	类型	状态
chips	玩家余额	int	必须

```
结算逻辑: `jp_bonus` + `back_amount` + `sum(win_amount)`
三方平台可以通过结算回调接口的`order_id`字段找到对应的下注信息，
需要注意的是因为slot旋转游戏的免费旋转和重转不会回调下注接口，所以回调结算接口时的`order_id`找不到对应的下注信息，
可以先通过结算回调接口的`is_free`字段进行判断，然后获取用户最后一次下注的信息做对应的结算
```

下注回滚

请求路径:

```
/wallet_bet_rollback
```

请求示例

```
{
    coin_code: "BRL",
    game_id: 1000101,
    room_type: "1",
    order_id: "123123123123123",
    sign: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

请求描述:

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
room_type	游戏倍场(玩法)	string(10)	必须
order_id	订单id(游戏内生成)	string(50)	必须

返回描述

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	必须
data	数据(成功时必须返回数据，详见 data说明)	map	可选
msg	备注说明	string	可选

返回描述:

名称	说明	类型	状态
chips	玩家余额	int	必须

更新金币

请求路径:

```
/wallet_update
```

请求示例

```
{
    coin_code: "BRL",
    game_id: 1000101,
    room_type: "1",
    order_id: "11123213213",
    sub_type_id: 80005,
    amount: 100000,
    extend: {...},
    sign: "16645220BD299887A4115171CE55CA2B"
}
```

请求描述

名称	说明	类型	状态
coin_code	金币类型	string(10)	必须
game_id	游戏id	int(64)	必须
room_type	游戏倍场(玩法)	string(10)	必须
order_id	订单id(游戏内生成)	string(50)	必须
sub_type_id	更新金币类型(详见 类型说明)	int(32)	必须
amount	更新数值	int(64)	必须
extend	根据 sub_type_id 类型不同会有不同的参数，详见 extend说明	map	可选

类型说明:

类型	描述
80005	小费(扣除)
80006	活动(增加)
80007	JP(增加)
80016	游戏内功能奖励(需要关心正负值)
80030	创房消耗/续房消耗(扣除)
80031	创房收益(增加)
101010	创房消耗回退(增加)

```
结算逻辑:
sub_type_id:80005 => 用户余额 - amount
sub_type_id:80006 => 用户余额 + amount
sub_type_id:80007 => 用户余额 + amount
sub_type_id:80016 => 用户余额 + amount
sub_type_id:80030 => 用户余额 - amount
sub_type_id:80031 => 用户余额 + amount
sub_type_id:101010 => 用户余额 + amount
```

extend说明:

```
sub_type_id 为 80007:
{
    jp_contri: 10000,   // jp贡献
    jp_bonus: 0         // jp奖金
```

```
}

sub_type_id 为 80030:
{
    friend_room_id: "123123122344"  // 好友房id，续房时关联多个order_id
}

sub_type_id 为 80031:
{
    friend_room_id: "123123122344"  // 好友房id，关联多个order_id
}

sub_type_id 为 101010:
{
    refund_order_id: "123123122344"  // 创房/续房的order_id
}
```

返回描述

名称	说明	类型	状态
code	返回码	int(0:成功 1:失败)	必须
data	数据(成功时必须返回数据，详见data说明)	map	可选
msg	备注说明	string	可选

data说明:

名称	说明	类型	状态
chips	玩家余额	int	必须

1.2.7 错误列表

错误名称	错误码	错误描述
INTERNAL_SERVER_ERR	1	系统内部错误
INVALID_REQUEST_ERR	2	无效的请求参数
OPERATE_DB_ERR	3	操作数据库失败
OPERATE_CACHE_ERR	4	缓存错误
MAINTAIN_ERR	5	服务维护中
TOO_MANY_REQUEST	6	请求过于频繁
ILEGAL_REQUEST	7	拆分成不同类型的非法请求
ILEGAL_TOKEN	8	非法token
ILEGAL_IP	9	非法IP请求
ILEGAL_SIGN	10	非法签名
INVALID_TOKEN_CONTENT	11	非法的token内容
REGION_REJECT	105	区域受限
ERR_GAME_TYPE_NOT_FOUND	100201	游戏类型找不到
ERR_GAME_NOT_FOUND	100202	游戏找不到
ERR_GAME_BRAND_NOT_FOUND	100203	游戏品牌找不到
ERR_GAME_ALIAS_NOT_FOUND	100204	游戏别名不存在
ERR_NEW_GAME_TOKEN_FAIL	100205	生成游戏token失败
ERR_INVALID_GAME_TOKEN	100206	无效的游戏token
ERR_DOMAIN_NOT_FOUND	100207	域名不存在
ERR_DEL_USED_GAME_TYPE	100208	删除游戏类型失败，游戏类型已被使用
ERR_DEL_USED_GAME	100209	删除已被使用的游戏
ERR_AGENT_NOT_FOUND	100210	代理不存在
ERR_CURRENCY_NOT_FOUND	100211	货币不存在
ERR_AGENCY_NOT_FOUND	100212	总社不存在
ERR_PLAYER_NOT_FOUND	100213	游戏玩家不存在
ERR_INVALID_GAME_ID	100214	无效的游戏ID
ERR_AGENT_DISABLED	100215	代理已被禁用
ERR_PLAYER_EXIST	100216	玩家已存在
ERR_DOMAIN_EXIST	100217	域名已存在
ERR_GAME_DISABLED	100218	游戏已被禁用
ERR_GAME_NOT_CONFIGED	100219	未配置游戏
ERR_INVALID_DOMAIN	100220	无效的域名
ERR_BIZ_ID_REPEAT	100221	业务ID重复

错误名称	错误码	错误描述
ERR_GM_NOT_FOUND	100222	GM不存在
ERR_START_VERSION_NOT_FOUND	100223	游戏的start版本不存在
ERR_GAME_RECORD_NOT_FOUND	100224	游戏记录不存在
ERR_LOADING_NOT_FOUND	100225	loading图不存在
ERR_GAME_TYPE_ALIAS_NOT_FOUND	100226	游戏类型别名不存在
ERR_IP_ACCESSABLE_NOT_FOUND	100227	IP白名单不存在
ERR_IP_NOT_ALLOWED	100228	IP受限
ERR_BRAND_LOADING_RECORD_NOT_FOUND	100300	brand_loading查询出错
ERR_CREATE_BRAND_LOADING	100301	创建品牌loading出错
ERR_DELETE_BRAND_LOADING	100302	删除品牌loading出错
ERR_UPDATE_BRAND_LOADING	100303	更新品牌loading出错
ERR_BRAND_LOADING_RECORD_ONE_NOT_FOUND	100304	查询一条品牌loading出错
ERR_AGENCY_LOADING_RECORD_NOT_FOUND	100305	agency_loading查询出错
ERR_CREATE_AGENCY_LOADING	100306	创建agency_loading出错
ERR_DELETE_AGENCY_LOADING	100307	删除agency_loading出错
ERR_UPDATE_AGENCY_LOADING	100308	更新agency_loading出错
ERR_AGENCY_LOADING_RECORD_ONE_NOT_FOUND	100309	查询一条agency_loading出错
ERR_AGENT_LOADING_RECORD_NOT_FOUND	100310	agent_loading查询出错
ERR_CREATE_AGENT_LOADING	100311	创建agent_loading出错
ERR_DELETE_AGENT_LOADING	100312	删除agent_loading出错
ERR_UPDATE_AGENT_LOADING	100313	更新agent_loading出错
ERR_AGENT_LOADING_RECORD_ONE_NOT_FOUND	100314	查询一条agent_loading出错
ERR_DEALER_LOADING_RECORD_NOT_FOUND	100315	dealer_loading查询出错
ERR_CREATE_DEALER_LOADING	100316	创建dealer_loading出错
ERR_DELETE_DEALER_LOADING	100317	删除dealer_loading出错
ERR_UPDATE_DEALER_LOADING	100318	更新dealer_loading出错
ERR_DEALER_LOADING_RECORD_ONE_NOT_FOUND	100319	查询一条dealer_loading出错
ERR_PLATFORM_LOADING_RECORD_NOT_FOUND	100320	platform_loading查询出错
ERR_CREATE_PLATFORM_LOADING	100321	创建platform_loading出错
ERR_DELETE_PLATFORM_LOADING	100322	删除platform_loading出错
ERR_UPDATE_PLATFORM_LOADING	100323	更新platform_loading出错
ERR_PLATFORM_LOADING_RECORD_ONE_NOT_FOUND	100324	查询一条platform_loading出错
ERR_UPDATE_DEFAULT_LOADING	100325	更新platform_loading出错

错误名称	错误码	错误描述
ERR_DELETE_LOADING_GATHER	100326	删除缓存聚合表失败
ERR_GAME_VERSION_RECORD_ONE_NOT_FOUND	100327	查询游戏版本号记录不存在
ERR_GAME_VERSION_RECORD_NOT_FOUND	100328	查询游戏列表的版本号不存在
ERR_CREATE_GAME_VERSION	100329	添加游戏版本入库失败
ERR_UPDATE_GAME_VERSION	100330	游戏版本入库更新失败
ERR_DELETE_GAME_VERSION	100331	游戏版本删除失败
ERR_UPDATE_VERSION_MD5_ZIP	100332	更新游戏的zip失败
ERR_GET_GAME_VERSION	100333	获取游戏版本号出错
ERR_GET_ETCD_GAME_VERSION	100334	获取etcd游戏版本号出错
ERR_IMPORT_GAME_VERSION	100335	导入游戏版本号出错
ERR_LANGUAGE_NOT_FOUND	100338	语言不存在
ERR_LANGUAGE_EXIST	100339	语言已存在
ERR_CREATE_DEFAULT_LOADING	100340	创建默认loading出错
ERR_INIT_LOADING	100341	初始化loading错误
ERR_INVALID_TOKEN	100400	无效的TOKEN
ERR_WEB_BRAND_NOT_FOUND	100500	WebBrand不存在
ERR_CURRENCY_CONVERSION_NOT_FOUND	100601	货币转换不存在
ERR_MAINTENANCE_EXIST	100700	维护已存在
ERR_MAINTENANCE_NOT_FOUND	100701	维护不存在
ERR_MAINTENANCE_NOT_EFFECT	100702	维护暂未生效
ERR_MAINTENANCE_NOT_DISABLED	100703	维护没有被禁用
ERR_NOT_IN_ALL_MAINTENANCED	100704	没有处于全局维护状态
ERR_GAME_BRAND_EXIST	100800	游戏品牌已存在
ERR_GAME_BRAND_IN_USED	100801	游戏品牌正在被使用
ERR_CREATE_AGENT_GAME	1001000	创建代理人游戏失败
ERR_FIND_AGENCY_GAME	1001001	查询总数游戏失败
ERR_CREATE_AGENCY_CURRENCY	1001002	创建总社货币失败
ERR_UPDATE_AGENCY_CURRENCY	1001003	更新总社货币失败
ERR_FIND_AGENCY_CURRENCY	1001004	查找总社货币失败
ERR_CREATE_AGENCY_CURRENCY_ETCD	1001005	总社货币写入Etcd失败
ERR_DELETE_AGENCY_CURRENCY_ETCD	1001006	删除总社货币失败
ERR_CREATE_IMPORT_GAME_VERSION_LOG	1001007	创建游戏版本号导入记录失败
ERR_SEARCH_IMPORT_GAME_VERSION_LOG	1001008	查询游戏版本号导入记录失败

错误名称	错误码	错误描述
ERR_UPDATE_IMPORT_GAME_VERSION_LOG	1001009	更新游戏版本号导入记录失败
ERR_PAGE_PARAMS	1001010	分页参数错误
ERR_CREATE_GAME_SWITCH	1001011	创建游戏开关错误
ERR_UPDATE_GAME_SWITCH	1001012	更新游戏开关错误
ERR_FIND_GAME_SWITCH	1001013	查询游戏开关列表错误
ERR_CREATE_CURRENCY	1001014	创建金币失败
ERR_DELETE_CURRENCY	1001015	删除金币失败
ERR_UPDATE_CURRENCY	1001016	更新金币失败
ERR_FIND_PLAYER	1001016	查找玩家失败
ERR_FIND_GAMEHALL	1001017	查找游戏大厅失败
ERR_CREATE_GAMEHALL	1001018	创建游戏大厅失败
ERR_UPDATE_GAMEHALL	1001019	更新游戏大厅失败
ERR_DELETE_GAMEHALL	1001020	删除游戏大厅失败
ERR_PRIZE_LIMIT_RULE_NOT_FOUND	1001021	奖赏限制规则不存在
ERR_GET_VERSION_CONF_ERR	1001022	获取游戏版本号配置
ERR_SAVE_VERSION_CONF_ERR	1001023	报错游戏版本号配置
ERR_FIND_GAMEHALL_GAMES	1001024	查找游戏大厅游戏失败
ERR_CREATE_GAMEHALL_GAMES	1001025	创建游戏大厅游戏失败
ERR_DELETE_GAMEHALL_GAMES	1001026	删除游戏大厅游戏失败
ERR_CREATE_LIBRARY_GAME	1001027	添加游戏大厅游戏库游戏失败
ERR_UPDATE_LIBRARY_GAME	1001028	修改游戏大厅游戏库游戏失败
ERR_DELETE_LIBRARY_GAME	1001029	删除游戏大厅游戏库游戏失败
ERR_FIND_LIBRARY_GAME	1001030	查找游戏大厅游戏库游戏失败
ERR_UPDATE_GAMEHALL_GAMES	1001031	更新游戏大厅游戏失败
ERR_INVALID_PLAYER_NAME	1001032	无效的玩家用户名
ERR_INVALID_PLAYER_PWD	1001033	无效的玩家密码
ERR_ACCOUNT_NOT_FOUND	102201	账户不存在
ERR_ACCOUNT_DISABLED	102203	被禁用
ERR_ACCOUNT_EXIST	102204	账户已存在
ERR_PAGE_PARAMS	102205	分页参数错误
ERR_DEALERLIST_NOT_FOUND	102206	经销商列表
ERR_DEALER_ADMIN_LIST_NOT_FOUND	102207	经销商管理列表
ERR_DEALER_ADMIN_SIMPLE_LIST_NOT_FOUND	102208	经销商短列表

错误名称	错误码	错误描述
ERR_CREATE_DEALER	102209	添加经销商失败
ERR_CREATE_ROLE_GROUP	102210	添加经销商角色组失败
ERR_CREATE_ROLE	102211	添加经销商角色失败
ERR_CREATE_ADMIN	102212	添加经销商管理员失败
ERR_CREATE_NAME	102213	添加经销商失败-name为空
ERR_CREATE_ACCOUNT	102214	添加经销商失败-account为空
ERR_CREATE_PASSWORD	102215	添加经销商失败-passward为空
ERR_CREATE_PASSWORDCONFIRM	102216	添加经销商失败-passwardConfirm为空
ERR_CREATE_PHONE	102217	添加经销商失败-phone为空
ERR_CREATE_AREA_CODE	102218	添加经销商失败-area_code为空
ERR_CREATE_CURRENCY_ID	102219	添加经销商失败-currency_id为空
ERR_CREATE_CURRENCY_NAME	102220	添加经销商失败-currency_name为空
ERR_CREATE_PASSWORD_NO_SAME	102221	添加经销商失败-两次密码不一样
ERR_CREATE_DEALER_EXISTS_ACCOUNT	102222	添加经销商失败-已经存在该账号
ERR_CREATE_DEALER_ID	102223	添加经销商失败-dealer_id错误
ERR_CREATE_MENUS	102224	添加经销商权限失败-menus错误
ERR_DEALER_DELETE	102225	删除经销商失败
ERR_DEALER_ADMIN_NOT_FOUND	102226	经销商管理员不存在
ERR_DEALER_ROLE_NOT_FOUND	102227	经销商管理角色不存在
ERR_DEALER_ROLE_GROUP_NOT_FOUND	102228	经销商管理员角色组不存在
ERR_DEALER_ADMIN_DELETE	102229	删除经销商管理员失败
ERR_ROLE_DELETE	102230	删除经销商管理员角色失败
ERR_ROLE_FOUND	102231	查找账号权限菜单
ERR_ROLE_UPDATE	102232	更新经销商管理员角色
ERR_DEALER_ROLE_GROUP_UPDATE	102233	更新经销商管理员组角色
ERR_ROLE_MERGE_MENU_UPDATE	102234	合并经销商管理员组角色权限菜单错误
ERR_DEALER_ROLE_LIST_NOT_FOUND	102235	经销商角色表查询不到
ERR_DEALER_ROLE_LIST_COUNT_NOT_FOUND	102236	经销商角色表查询总数错误
ERR_DEALER_ID_UPDATE	102237	经销商更新id错误
ERR_DEALER_UPDATE	102241	经销商更新错误
ERR_DEALER_SUB_ADMIN_LIST_NOT_FOUND	102244	经销商管理子账号列表为空
ERR_AGENCY_SUB_ADMIN_LIST_NOT_FOUND	102245	总社管理子账号列表为空
ERR_AGENT_SUB_ADMIN_LIST_NOT_FOUND	102246	代理人管理子账号列表为空

错误名称	错误码	错误描述
ERR_DEALER_NAME_EXISTS	102247	经销商已存在
ERR_PASSWORD_UPDATE	102238	账号密码修改失败
ERR_ACCOUNT_UPDATE	102239	账号信息更新失败
ERR_ACCOUNT_CREATE	102240	账号创建失败
ERR_AGENCY_ADMIN_LIST_NOT_FOUND	102241	总社管理列表
ERR_AGENT_ADMIN_LIST_NOT_FOUND	102242	代理人管理列表
ERR_AGENCY_CREATE	102243	添加总社失败
ERR_AGENT_CREATE	102244	添加代理人失败
ERR_AGENCY_UPDATE	102245	更新总社失败
ERR_AGENT_UPDATE	102246	更新代理人失败
ERR_AGENCYLIST_NOT_FOUND	102247	总社列表
ERR_AGENTLIST_NOT_FOUND	102248	代理人列表
ERR_AGENCY_COSTSETTING_NOT_FOUND	102249	总社费用查询出错
ERR_CREATE_AGENCY_COSTSETTING_NOT_FOUND	102250	总社费用添加
ERR_CREATE_AGENCY_COSTSETTING	102251	总社费用添加或者编辑失败
ERR_AGENCY_DELETE	102252	删除总社失败
ERR_AGENT_CURRENCY_FOUND	102246	代理人货币列表查询出错
ERR_UPDATE_PLAER_NUM	102247	更新玩家总数失败
ERR_FIND_CURRENCY_IS_USERD	102248	查询货币是否被使用
ERR_CREATE_AGENT_PARENTS	102249	创建多级代理人关系记录失败
ERR_FIND_AGENT_PARENTS	102250	查询多级代理人关系记录失败
ERR_UPDATE_AGENCY_SECRET_KEY	102251	更新总社的secret_key失败
ERR_FIND_AGENT_PARENT_CHILD_TREE	102252	获取多级代理人的菜单失败
ERR_INCORRECT_PASSWORD	102253	不正确的密码
ERR_ACCOUNT_CREATE_PERMISSION	102254	账号创建权限失败
ERR_BETLOG_NOT_FOUND	103201	游戏注单不存在
ERR_BETLOG_DETAIL_NOT_FOUND	103202	注单详情不存在
ERR_BETLOG_COMMON_DETAIL_NOT_FOUND	103203	注单公共详情不存在
ERR_YIYUAN_ORDER_SUMMARY_FOUND	103300	一元购汇总查询错误
ERR_CREATE_ROOM_SUMMARY_FOUND	103301	创房汇总查询错误
ERR_CASHFLOW_NOT_FOUND	103400	流水不存在
ERR_BIZ_ID_NOT_FOUND	103403	biz_id不存在

错误名称	错误码	错误描述
InternalServerError	100901	系统内部错误
InvalidRequest	100902	无效的请求
InsufficientBalance	100903	余额不足
PlayerNotFound	100904	玩家不存在
InTableGameCanNotWitdraw	100905	因为在桌游游戏中所以无法取款
FriendRoomNotFound	100906	好友房不存在
UserCanNotBeKicked	100907	无法踢出玩家

1.3 常见问题

1.3.1 APP内返回

触发原理

1. API游戏点击返回，会有个提示框，提示框点击确定后，游戏会先判定当前打开方式，是直接网页打开，还是包网打开方式。
2. 判定方式： Url链接携带参数 lobby /lobby_web /lobby_h5，
3. 参数是 lobby：判定是原生APP 打开游戏（ios、android）
4. 参数是 lobby_web：判定是 Pc网页 打开游戏
5. 参数是 lobby_h5：判定是 手机网页 打开游戏
6. 游戏会发送消息通知保网关闭或者window.close
7. 游戏关闭:
8. android 端:android重写了web 的window.close，然后android 关闭webview。
9. iOS 端：window['webkit'].messageHandlers.quitGamge.postMessage(''); 游戏发送消息quitGamge。
10. web 端：如果return_url 有值跳转页面： window.location.href = "return_url"； 无值返回上个页面：window.history.back();

打开api url链接需要接入参数

```
链接参数:
eg: https://xxxxxxxxxxx?
game_id=MTAwNzM0Mw==&api_game=aHR0cHM6Ly9hcGktZ2FtZS55c2l0LmFwcA==&api_game_token=xxxxx&language=en&lobby=lobby_300&language=pu&app_canRecharge=MQ==&app_toke
n=xxxxxxv&app_version=MTAwMQ==&app_environment=c2l0&app_deviceId=MTY4Mzc5MDU3NzIwOA==&lobby_url=YXBwLXplcm8ueXNpdC5hcHA=
```

```
//参数如下:
game_id=MTAwNzM0Mw==                    //api游戏id (base64)
api_game=xxxxxxx                        //api游戏地址
api_game_token=xxxxxx                   //api游戏token
language=en                             //pu/tw/en/id/..)
lobby/lobby_h5/lobby_web=xxxxxxx         //有lobby参数就行值随意，api游戏通过有lobby参数会对应不同的弹窗
app_canRecharge=MQ==                    //游戏内是否可充值
app_deviceId=MTY4Mzc5MDU3NzIwOA==       //设备id  （可选）
memory=Mw==                             //内存大小 （可选）
recharge_url=xxxxxx                     //充值url链接(base64 编码)  （可选）
```

android修改：

```
/// webView消息监听
/*--------------------
 - quitGamge          监听退出游戏
 ---------------------*/

//给webview 注入 AndroidApp  android实例
webview.addJavascriptInterface(this, "AndroidApp")

class WebDialogNativeMethod(val dialog: CustomWebDialog, val activity: public_name_AppActivity) {
    /**
     * 关闭当前页面
     */
    @JavascriptInterface
    fun closePage() {
        activity.runOnUiThread {
            dialog.dismiss()
        }
    }
}

/**
 * 重写webview window.open close log 等方法
 */
fun overrideWindowOpen(webView: WebView) {
    webView.loadUrl("javascript:window.close = function() {window.AndroidApp.closePage();}")
}
```

IOS修改：

```
XLWeakScriptMessageDelegate  需要实现 WKScriptMessageHandler协议   不能通过直接直接设置self 的方式来接受代理回调  会有内存泄露  需要包一层

//声明XLWeakScriptMessageDelegate 文件
@protocol WKScriptMessageHandler;

@interface XLWeakScriptMessageDelegate : NSObject<WKScriptMessageHandler>
@property (nonatomic,weak)id<WKScriptMessageHandler> scriptDelegate;
- (instancetype)initWithDelegate:(id<WKScriptMessageHandler>)scriptDelegate;
@end


// XLWeakScriptMessageDelegate 实现
#pragma mark - XLWeakScriptMessageDelegate
@implementation XLWeakScriptMessageDelegate

- (instancetype)initWithDelegate:(id<WKScriptMessageHandler>)scriptDelegate{
    self = [super init];
    if (self) {
        _scriptDelegate = scriptDelegate;
    }
    return self;
}

- (void)userContentController:(WKUserContentController *)userContentController
      didReceiveScriptMessage:(WKScriptMessage *)message{
    [self.scriptDelegate userContentController:userContentController didReceiveScriptMessage:message];
}

@end


/// webView消息监听
/*----------------------
 - quitGamge          监听退出游戏
 ----------------------*/
static NSString * const kScriptMessageQuitGame = @"quitGamge";

/// 增加代理 监听消息
- (void)webConfigAddScriptMessage {
    XLWeakScriptMessageDelegate * delegate = [[XLWeakScriptMessageDelegate alloc] initWithDelegate:self];
    [self.userContentController addScriptMessageHandler:delegate name:kScriptMessageQuitGame];
}

@property (nonatomic,strong) WKUserContentController * userContentController;
//移除代理监听
- (void)dealloc {
    //这里需要注意，前面增加过的方法一定要remove掉。
    [self.userContentController removeScriptMessageHandlerForName:kScriptMessageQuitGame];
}

//WKScriptMessageHandler 协议方法=的回调
#pragma mark - WKScriptMessageHandler
- (void)userContentController:(WKUserContentController *)userContentController
      didReceiveScriptMessage:(WKScriptMessage *)message {
    NSLog(@"message_name:%@\\\\n body:%@\\\\n frameInfo:%@\\\\n",message.name,message.body,message.frameInfo);
    NSString *body = message.body;
    /// 退出游戏
    if ([message.name isEqual:kScriptMessageQuitGame]) {

    }
}
```

1.3.2 移动端接入游戏全屏设置说明

一、主页中iframe嵌入游戏

1. **设置iframe高度：**
2. 需要将iframe的高度设置为大于游戏文档页面的高度大约100px，或者设置为 `100vh` 来适应全屏显示。
3. **监听子游戏的滚动事件：**
4. 在主页面中监听来自子游戏页面的滚动事件，确保游戏能够在全屏模式下滚动至顶部。

```
<script>
window.addEventListener('message', (event) => {
    // 如果收到滚动到顶部的消息
    if (event.data.type === 'fullScreenScroll') {
        window.scrollTo(0, 0); // 页面滚动到顶部
    }
});
</script>
```

二、新建窗口打开游戏

- 如果通过新建窗口打开游戏，**无需额外处理**，系统会自动处理全屏功能。

三、关闭游戏内的滑动全屏功能

- 可以在启动游戏中增加参数 `close_full=1` 来关闭游戏内的滑动全屏功能。例如：

```
{
    "game_id": 3003111,
    "language": "en",
    "is_free": 0,
    "third_name": "xxx",
    "agent_id": 123,
    "return_url": "",
    "recharge_url": "",
    "terminal": "",
    "extra": "",
    "launch_params": {
        "close_full": "1"
    },
    "client_ip": "xxx.xxx.xxx.xxx"
}
```

1.3.3 游戏旋转提示层说明

- 部分游戏只能竖屏或者横屏游戏，在不支持的的情况下会出现旋转提示遮罩层
- 触发条件：

a. 游戏加载后 0.6s 内多次检测横竖状态，并且根据游戏类型展示旋转提示遮罩层。

b. 当web页面触发resize 或者 orientationchange 事件时，检测横竖屏状态，并且根据游戏类型展示旋转提示遮罩层。

c. 每隔1s检测一次横竖屏状态，当提示层展示的时候会关闭。

- 判定横竖状态:

```
<script>
   var width = window.innerWidth || document.documentElement.clientWidth || document.body.clientWidth;
   var height = window.innerHeight || document.documentElement.clientHeight || document.body.clientHeight;
   width > height 为横屏，否则为竖屏。
</script>
```

- 注意： 有时页面加载完成时机早于横竖锁定，导致页面0.6s 内获取到未旋转锁定前的值，出现误判，旋转tip层无法关闭的情况，增加了兜底每隔1s检测一次横竖屏状态，当提示层展示的时候会关闭。

1.3.4 FQA

Q: 转账钱包启动游戏流程是什么?

A: 调用接口流程: 创建玩家 -> 存入玩家金币 -> 启动游戏 -> 玩家退出游戏 -> 取出玩家金币

Q: 金币的单位是什么?

A: 金币在系统中都是整数，所以金币的值为当前货币的最小单位。例如：USD有两位精度，"$0.1"对应的金币就是"0.1 * 100 = 10"，其中100 = pow(10, 精度)。

Q: 免转钱包启动游戏流程是什么?

A: 调用接口流程: 创建玩家 -> 启动游戏 -> 玩家退出游戏

Q: 免转钱包结算金币的结算公式是什么?

A: 结算金币 = jp_bonus + sum(win_amount) + back_amount。`win_amount` 在捕猎游戏中可能有多条，需要把每一条的 `win_amount` 求和。`back_amount` 是退回结算的本金。

Q: 免转钱包下注/加注的逻辑是什么，为什么有 `amount` 和 `min_amount`？

A: `amount` 最大金额包含预扣金额，`min_amount` 最小金额为单次投注金额。在捕猎游戏中有预扣下注金币，最多预扣200发。所以 `amount` 与 `min_amount` 不相等。在扣除金币时，如果当前余额小于 `min_amount`，扣除失败。如果当前余额大于 `amount`，返回扣除 `amount`，否则扣除当前余额即可。

Q: 免转钱包结算金币中 `is_order_finish` 什么时候是 `true`，什么时候是 `false`？一个游戏回合会调用多次吗?

A: 只有在捕猎游戏中，`is_order_finish` 才可能是 `false`。一次押注需要多次结算，并且只能写一次游戏记录。前面的结算是 `false`，最后的结算是 `true`。例如，捕猎一次押注只能有对应的一次游戏记录，但是需要结算多次，这就有这种情况。

Q: 免转钱包 `order_id`、`bet_id`、`settlement_id` 之间是什么关系?

A: `order_id` 是一局游戏记录的 ID，`bet_id` 是下注 ID，每调用一次下注接口 `bet_id` 都不相同，`bet_id` 应保持幂等。每调用一次结算接口 `settlement_id` 都不相同，`settlement_id` 应保持幂等。一局游戏可能会出现多次下注和结算，下注和结算并不是一一对应关系。例如，捕猎可能会出现多次下注一次结算，所以 `bet_id` 和 `settlement_id` 没有直接关系。`order_id` 可以对应多个 `bet_id` 和多个 `settlement_id`。`is_order_finish` 为 `true` 表示订单已经全部结算。

Q: 需签名的 `value` 值不是简单值，是一个对象怎么处理?

A: 如果 `value` 是一个对象，需要将对象转换为 JSON 字符串，其中对象的 key 也需要保持 ASCII 升序排列。大多数编程语言在将对象转换为 JSON 字符串时都是升序的。如果签名不通过，需检查转换出来的 JSON 字符串是否符合规则。

Q: 什么时候会触发下注回滚?

A: 1. 桌游游戏可以下注多个区域且有续投功能的。如果钱不够所有区域的总和，就会触发回滚。2. 多人一起玩游戏需要同时扣钱时，如果其中一个扣钱失败，会导致所有人的回滚。

Q: 游戏玩法哈希、时间戳和创世哈希有什么区别?

A: 1.哈希玩法：是提前公布下一局的哈希密文给玩家，出结果后玩家可以验证是否结果有更改。 2.时间戳玩法：根据玩家启动的时间戳转换成哈希值，计算玩家的结果坐标。 3.创世哈希玩法：玩家上线后生成10000条哈希值，从后往前拿哈希值当成下一局的结果。

Q: 游戏记录的保留时间是多长?

A： 只保留最近60天的游戏记录。

Q: "转账钱包"如何确定某笔“存取玩家金币”是否成功?

A： 可发起第二次请求，如返回结果为“交易ID(biz_id)重复”，则说明上一次请求已成功；如返回结果为“成功”，则说明上一次请求失败，但第二次请求已成功

Q: "免转钱包"中捕猎类游戏如何将下注/加注、结算金币与游戏记录进行匹配关联?

A： 玩家进入捕猎类游戏打出第1发子弹，API会根据“第1发子弹的价值*200”来请求预扣的额度，当玩家退出游戏或当前预扣额度消耗完，API会发起结算请求，同时产生1条游戏记录，该游戏记录的order_id与下注/加注、结算金币中的order_id保持一致

Q: 如何测试“免转钱包-下注回滚”?

A： 可以进入游戏"哈希奖池(游戏ID:23007751)",并进行下注，等待游戏流局，此时我方api将发起"下注回滚"请求

Q: 如何测试“免转钱包-更新金币”?

A： 可以进入游戏"百人二八杠(游戏ID:3000901)"，点击游戏大厅右上角"创房"按钮并成功创建房间，此时我方api将发起"更新金币"请求

Q: 签名错误如何排查?

A： 1、核对是否将请求体中的参数名进行了升序排序 2、核对“免转钱包”签名是否将请求体中除sign外的所有参数名进行了升序排序 3、核对签名时使用的key值是否与我方提供的api密钥一致

Q: 结算金币和更新金币的区别是什么?

A： 结算金币——对应前面1笔或多笔相同round_id的下注/加注的结算 更新金币——玩家在游戏内出现了“小费、活动、JP、游戏内功能奖励、创房消耗/续房消耗、创房收益、创房消耗回退”这些场景，而发起的请求

Q: 免转钱包签名验证时请求参数名称升序排序示例

A：
```
back_amount=10000&coin_code=BRL&details=[{"all_bets":1000,"amount":1000,"bet_id":"111","coded_quantity":100,"index":1,"is_finish":true,"win_amount":100},{"all_bets":0,"amount":1000,"bet_id":"222","coded_quantity":200,"index":2,"is_finish":true,"win_amount":0}]&game_id=1000101&order_id=111111&room_type=0&round_id=1111111&settlement_id=121212121&key=key1234567
```

Q: "免转钱包-更新金币-游戏内功能奖励(80016)"什么情况下会出现负值?

A： 可以进入游戏“财神捕鱼(游戏id：2000301)”中，当捕获“猫娘”后将获得“猜大小”小游戏，玩家第1次猜是免费的，如果输了玩家可以选择继续猜，这时我们api就会通过接口回调请求amount值为负数

Q: api白名单是否支持添加ipv6地址?

A： 支持

Q: 玩家在游戏中购买免费旋转，中途离开游戏后的处理机制?

A： 5分钟内不返回游戏，会自动帮玩家使用完剩余的免费旋转并结算；如果切换到其他游戏，会自动帮玩家使用完剩余的免费旋转并结算

2. 变更记录