Game Platform Publisher Manual Document

Product Name: Game Platform

Service Name:

Publisher Manual

Team: []

Document Last Updated Date: (25-March-2025)

Confidentiality

Document ID & Revision History

Version	Date	Description of Changes	Author
v1.0.0	2025-1-22	Initial draft	

Detailed change log:

v1.0.0

- Initial version, includes game lists, tournament interfaces, and webhook interfaces.

Table of Contents

1.API 签名和验签

1.1 基础概念

1.1.1 什么是APIKey?

APIKey是一种用于身份验证和授权的密钥，通常由服务端生成并分发给客户端。客户端在进行API请求时，使用APIKey参与签名生成，以确保请求的完整性和安全性。APIKey本身不需要传递给服务端，只需在客户端参与签名计算即可。

1.1.2 签名和验签

1. **客户端**：使用APIKey和其他请求参数生成签名。
2. **服务端**：使用相同的APIKey和接收到的请求参数重新计算签名，并与客户端传递的签名进行比对，以验证请求的合法性。

1.1.3 获取APIKey

首先请先联系平台管理员或者商务对接人获取您（公司或组织）的账号和密码，使用 `GPCLI` 脚手架工具登录您的账号

```
./gpcli.exe login <UserID> <Password>
```

如果账号和密码都正确，会显示您的APIKey：

```
.\gpcli.exe login User-XXX Pwd-XXX
Login successful!
Organization APIKey: abcdx8baztbtOjq2hk52v69Odmdgz123
Please keep your organization's APIKey safe and do not disclose it to third
parties.
```

1.2 签名生成流程

1.2.1 准备请求参数

在生成签名之前，客户端需要准备以下信息：

- **APIKey**：登录账号后分发的密钥，用于参与签名生成。
- **UserID**：客户端的唯一标识符，就是登录账号时使用的UserID。
- **Timestamp**：当前时间戳，单位为秒，用于防止重放攻击。
- **Nonce**：随机生成的字符串，确保每次请求的签名唯一。

1.2.2 生成签名

- **拼接字符串**：
  将 ClientID、Timestamp、Nonce 和 APIKey 按照固定顺序拼接成一个字符串。
  示例：`message = clientID + timestamp + nonce + apiKey`;
- **计算签名**
  使用SHA-256算法对拼接后的字符串进行哈希计算，生成签名。
  示例：signature = crypto.SHA256(message).toString(crypto.enc.Hex);

1.2.3 设置请求头

将生成的签名及相关参数添加到请求头中：

- X-Timestamp：时间戳。
- X-Client-ID：客户端ID。
- X-Nonce：随机数。
- X-Signature：生成的签名。

1.3 示例代码

1.3.1 Javascript示例代码

```
const crypto = require('crypto-js');

const apiKey = "your_api_key";
const userID = "your_user_id";
const timestamp = Math.floor(Date.now() / 1000).toString();
const nonce = crypto.randomBytes(16).toString('hex');

// 拼接字符串
const message = userID + timestamp + nonce + apiKey;
// 计算签名
const signature = crypto.SHA256(message).toString(crypto.enc.Hex);

// 设置请求头
const headers = {
    'X-Timestamp': timestamp,
    'X-Client-ID': clientID,
    'X-Nonce': nonce,
    'X-Signature': signature
};

// 发起请求
fetch('https://api.example.com/endpoint', {
    method: 'POST',
    headers: headers,
    body: JSON.stringify({})
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

1.3.2 Golang示例代码

```
apiKey := "your_api_key"
clientID := "your_user_id"
timestamp := time.Now().Unix()
nonce := generateNonce(16)
// 拼接字符串
message := fmt.Sprintf("%s%d%s%s", clientID, timestamp, nonce, apiKey)
// 计算签名
hash := sha256.Sum256([]byte(message))
signature := hex.EncodeToString(hash[:])
// 设置请求头
headers := map[string]string{
    "X-Timestamp": fmt.Sprintf("%d", timestamp),
    "X-Client-ID": clientID,
    "X-Nonce":     nonce,
    "X-Signature": signature,
}
// TODO: Send HTTP request
// ...
```

1.4 注意事项

1. **APIKey的安全性**：APIKey是身份验证的关键，务必妥善保管，避免泄露。
2. **时间戳的有效性**：服务端会校验时间戳的有效性，通常允许的时间偏差为±5分钟。
3. **Nonce的唯一性**：每次请求的Nonce应确保唯一，防止重放攻击。
4. **签名的计算**：确保客户端和服务端使用相同的算法和参数顺序计算签名，否则会导致验签失败。

2.API接口调用

所有的API请求都是以HTTP的形式请求的，而且必须是以HTTPS的形式

2.1 请求地址

环境	请求地址
测试环境	https://www.ltonlinegame.com/
生产环境	https://www.ltonlinegame.com/

2.2 请求头说明

每个API的HTTP请求需要包含以下HTTP头：

请求头	是否必需	说明	示例值
X-Client-ID	是	客户端ID（用户ID）	user_a123456789
X-Signature	是	请求签名	签名字符串
X-Timestamp	是	请求时间戳（Unix时间戳）	1641234567
X-Nonce	是	随机字符串，用于防重放攻击	550e8400-e29b-41d4-a716-446655440000

2.3 响应格式

API响应统一使用JSON格式，基本结构如下：

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	mixed	响应数据，可能是字符串、数字、数组或对象

响应示例：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "id": "1234567890",
        "name": "示例数据",
        "created_at": "2024-01-21T10:30:00Z"
    }
}
```

3. 游戏接口

3.1 获取游戏列表

获取渠道商配置的游戏信息，注意：返回的游戏列表只包含已经上线的游戏。

3.1.1 请求地址

接口名称	URL地址	HTTP方法
game.list	`/service/jsonapi/game/list`	GET

3.1.2 请求参数

无

3.1.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	游戏列表，每个元素为游戏信息对象

游戏信息对象（data数组中的元素）字段说明：

字段名	类型	说明
id	int64	游戏记录ID，系统自动生成
gameId	string	游戏ID，唯一标识符
gameType	string	游戏算法ID，标识游戏类型
serialNo	int	游戏SN号，每个游戏都有一个唯一的SN号
developerId	string	开发者ID，游戏开发商的唯一标识
version	string	游戏发布版本号
uri	string	游戏访问的相对路径

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "id": 2,
            "gameId": "slot_zhaocaijinbao",
            "gameType": "gp_slot_zhaocaijinbao",
            "serialNo": 1,
            "developerId": "devsz",
            "version": "2025.01.10.0004",
            "uri": "games/devsz/slot_zhaocaijinbao/client"
        },
        {
            "id": 6,
            "gameId": "slot_mayan",
            "gameType": "gp_slot_mayan",
            "serialNo": 2,
            "developerId": "devsz",
            "version": "2025.01.16.0001",
            "uri": "games/devsz/slot_mayan/client"
        }
    ]
}
```


错误响应：


```
{
    "code": 1,
    "message": "Invalid request",
    "data": null
}
```


3.1.4 注意事项

1. 该接口只返回状态为"上线"（status=2）的游戏列表
2. 游戏配置字段（config）为JSON格式字符串，具体配置项根据不同游戏类型有所不同
3. 建议实现适当的缓存机制，避免频繁请求该接口
4. 返回的游戏列表按照创建时间倒序排列

3.2 获取游戏版本列表

获取指定游戏的版本列表。

3.2.1 请求地址

接口名称	URL地址	HTTP方法
game.versions	`/service/jsonapi/game/versions`	POST

3.2.2 请求参数

字段名	类型	是否必需	说明
gameId	string	是	游戏ID
page	int	否	数据分页的当前页码, 默认是0
size	int	否	每页数量，默认是10

请求示例：

```
{
  "page": 0,
  "size": 3,
  "gameId": "slot_alice"
}
```

3.2.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	版本列表，字符串数组

响应示例：

```
{
    "code": 0,
    "message": "",
    "data": [
        "2025.03.05.0002",
        "2025.03.04.0001",
        "2025.02.23.0001"
    ]
}
```

3.3 获取游戏详情

获取渠道商配置的游戏信息，注意：返回的游戏列表只包含已经上线的游戏。

3.3.1 请求地址

接口名称	URL地址	HTTP方法
game.info	`/service/jsonapi/game/info`	GET

3.3.2 请求参数

`url: {host}/service/jsonapi/game/info?gameId={GameID}`

字段名	类型	是否必需	说明
gameId	string	是	游戏ID

3.3.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	游戏详情信息

游戏信息对象（data数组中的元素）字段说明：

字段名	类型	说明
id	int64	游戏记录ID，系统自动生成
gameId	string	游戏ID，唯一标识符
gameType	string	游戏算法ID，标识游戏类型
serialNo	int	游戏SN号，每个游戏都有一个唯一的SN号
developerId	string	开发者ID，游戏开发商的唯一标识
version	string	游戏发布版本号
uri	string	游戏访问的相对路径
status	int32	游戏状态： -1: 禁用 0: 未发布 1: 已发布 2: 上线
config	string	游戏配置信息，JSON格式字符串
createAt	string	记录创建时间，ISO8601格式，如："2024-01-21T10:30:00Z"

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": {
        "id": 8,
        "gameId": "slot_alice",
        "gameType": "slot_alice",
        "serialNo": 3,
        "developerId": "devsz",
        "version": "2025.02.23.0001",
        "uri": "games/devsz/slot_alice/client",
        "status": 1,
        "config":"{\"backend\":{\"type\":\"slot_alice\"}",
        "createAt": "2025-01-06T04:47:09+08:00"
    }
}
```


错误响应：


```
{
    "code": 1,
    "message": "Invalid request",
    "data": null
}
```


3.4 获取游戏历史记录

获取游戏历史记录，可根据玩家ID、游戏ID、时间段等条件查询。

3.4.1 请求地址

接口名称	URL地址	HTTP方法
game.list	`/service/jsonapi/game/records`	POST

3.4.2 请求参数

字段名	类型	是否必需	说明
userIdentity	string	否	用户唯一标识符，用于标识用户，没有该参数表示查询所有用户的记录
action	string	否	操作类型，`prev` 表示上一页，`next` 表示下一页，默认值next
cursor	int64	否	游标ID，用于分页查询，如果 `action` 为 `prev`，获取id比cursor大的数据；如果 `action` 为 `next`，获取id比cursor小的数据
developerId	string	否	开发者ID，没有该参数表示查询所有游戏的记录
channelId	string	否	渠道ID，没有该参数表示查询所有渠道的记录
version	string	否	版本ID，没有该参数表示查询所有游戏版本的记录
size	int	否	每页的数，默认是10量
gameId	string	否	游戏ID，没有该参数表示查询所有游戏的记录
startTime	int64	否	开始时间，以时间戳（秒）形式表示
endTime	int64	否	结束时间，以时间戳（秒）形式表示

请求示例:

```
{
  "action": "next",
  "cursor": 0,
  "size": 2,
  "userIdentity": "6",
  "developerId": "",
  "channelId": "",
  "versionId": "",
  "gameId": "slot_alice",
  "startTime": 0,
  "endTime": 0
}
```

3.4.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	游戏记录列表

游戏记录对象（data数组中的元素）字段说明：

字段名	类型	说明
id	int	记录ID
gameId	string	游戏ID
gameType	string	游戏算法ID
userIdentity	string	用户ID
userOrg	string	用户所属渠道ID
transactionId	string	交易ID
startingCredit	bigint	开始积分
endingCredit	bigint	结束积分
baseWin	bigint	基础赢取积分
freeWin	bigint	免费赢取积分
bonusWin	bigint	奖励赢取积分
jackpotWin	bigint	奖池赢取积分
totalWin	bigint	总赢取积分
totalBet	bigint	总下注积分
baseBet	int	基础下注
line	int	彩线数
multiple	int	倍数
playTime	int	游戏时间
createAt	string	创建记录时间
version	string	游戏版本
developerId	string	游戏开发者ID

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "id": 13282,
            "gameId": "slot_alice",
            "gameType": "slot_alice",
            "userIdentity": "6",
            "channelId": "chnsz",
            "transactionId": "1741330279372-093jvgn99kt9",
            "startingCredit": 632300,
            "endingCredit": 0,
            "baseWin": 0,
            "freeWin": 0,
            "bonusWin": 0,
            "jackpotWin": 0,
            "totalWin": 0,
            "totalBet": 100,
            "baseBet": 100,
            "line": 1,
            "multiple": 1,
            "playTime": 1741330280,
            "createAt": "2025-03-07T06:51:20+08:00",
            "version": "2025.03.05.0002",
            "developerId": "devsz"
        },
        {
            "id": 13281,
            "gameId": "slot_alice",
            "gameType": "slot_alice",
            "userIdentity": "6",
            "channelId": "chnsz",
            "transactionId": "1741330234455-46adw07tu51",
            "startingCredit": 628600,
            "endingCredit": 632400,
            "baseWin": 3800,
            "freeWin": 0,
            "bonusWin": 3800,
            "jackpotWin": 0,
            "totalWin": 7600,
            "totalBet": 100,
            "baseBet": 100,
            "line": 1,
            "multiple": 1,
            "playTime": 1741330266,
            "createAt": "2025-03-07T06:51:06+08:00",
            "version": "2025.03.05.0002",
            "developerId": "devsz"
        }
    ]
}
```

错误响应：

```
{
    "code": 1,
    "message": "Invalid request",
    "data": null
}
```

3.5 获取未完成的游戏列表

获取当前品牌（Brand）下处于进行中或由于钱包问题而出现错误的游戏列表

3.5.1 请求地址

接口名称	URL地址	HTTP方法
game.incompleteGames	`/service/jsonapi/game/incompleteGames`	GET

3.5.2 请求参数

无

3.5.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	未完成游戏列表

未完成游戏（data数组中的元素）字段说明：

字段名	类型	说明
userIdentity	string	用户唯一标识符，用于标识用户
gameId	string	游戏ID
betAmount	number	投注金额
totalWin	number	游戏总赢金额
gameStatus	number	游戏状态,2:进行中,3:下注失败,4：入账失败
error	string	错误信息

响应示例：


```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userIdentity": "user_a123456789",
            "gameId": "slot_zhaocaijinbao",
            "betAmount": 8800,
            "totalWin": 0,
            "gameStatus": 2,
            "error": ""
        },
        {
            "userIdentity": "user_a123456790",
            "gameId": "slot_zhaocaijinbao",
            "betAmount": 8800,
            "totalWin": 0,
            "gameStatus": 3,
            "error": "下注失败"
        }
    ]
}
```


3.6 获取特定玩家的未完成游戏列表

返回一个指定玩家未完成的游戏列表，通常包括进行中的游戏和因某些问题（如系统故障或网络中断）而未能完成的游戏

3.6.1 请求地址

接口名称	URL地址	HTTP方法
game.playerResumeGames	`/service/jsonapi/game/playerResumeGames`	GET

3.6.2 请求参数

字段名	类型	是否必需	说明
userIdentity	string	是	用户唯一标识符，用于标识用户

3.6.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	未完成游戏列表

未完成游戏（data数组中的元素）字段说明：

字段名	类型	说明
userIdentity	string	用户唯一标识符，用于标识用户
gameId	string	游戏ID
betAmount	number	投注金额
totalWin	number	游戏总赢金额
gameStatus	number	游戏状态,2:进行中,3:下注失败,4：入账失败
error	string	错误信息

请求示例：

```
{
    "userIdentity": "user_a123456789"
}
```

响应示例：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userIdentity": "user_a123456789",
            "gameId": "slot_zhaocaijinbao",
            "betAmount": 8800,
            "totalWin": 0,
            "gameStatus": 2,
            "error": ""
        },
        {
            "userIdentity": "user_a123456789",
            "gameId": "slot_mayan",
            "betAmount": 50,
            "totalWin": 0,
            "gameStatus": 3,
            "error": "下注失败"
        }
    ]
}
```

3.7 将特定玩家的所有游戏标记为过期

标记玩家所有游戏为过期：这个接口会把一个玩家的所有正在进行中的或未完成的游戏状态标记为“过期”。
批量处理未完成的游戏：通常当玩家的游戏出现故障、系统问题，或需要进行清理时，运营团队可以使用此接口来批量处理玩家的游戏。

3.7.1 请求地址

接口名称	URL地址	HTTP方法
game.expirePlayerGames	`/service/jsonapi/game/expirePlayerGames`	POST

3.7.2 请求参数

字段名	类型	是否必需	说明
userIdentity	string	是	用户唯一标识符，用于标识用户

3.7.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	过期游戏列表

过期游戏（data数组中的元素）字段说明：

字段名	类型	说明
gameId	string	游戏ID
isExpired	bool	是否过期处理成功
error	string	错误信息，仅当isExpired为false时有值返回

请求示例:

```
{
    "userIdentity": "user_a123456789"
}
```

响应示例：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "gameId": "slot_zhaocaijinbao",
            "isExpired": true
        },
        {
            "gameId": "slot_mayan",
            "isExpired": false,
            "error": "内部错误"
        }
    ]
}
```

3.8 将所有游戏标记为过期

标记品牌下所有游戏为过期：通过这个接口，可以将某个品牌所有进行中的或未完成的游戏标记为“过期”

3.8.1 请求地址

接口名称	URL地址	HTTP方法
game.expireGames	`/service/jsonapi/game/expireGames`	POST

3.8.2 请求参数

无

3.8.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述，成功时为"success"，失败时为错误描述
data	array	过期游戏列表

过期游戏（data数组中的元素）字段说明：

字段名	类型	说明
userIdentity	string	用户唯一标识符，用于标识用户
gameId	string	游戏ID
isExpired	bool	是否过期处理成功
error	string	错误信息，仅当isExpired为false时有值返回

响应示例：


```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userIdentity": "user_a123456789",
            "gameId": "slot_zhaocaijinbao",
            "isExpired": true
        },
        {
            "userIdentity": "user_a123456790",
            "gameId": "slot_mayan",
            "isExpired": false,
            "error": "内部错误"
        }
    ]
}
```

4. 钱包接口

4.1 钱包转账（充值）

向指定钱包转入金额。

4.1.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
wallet.deposit	`/service/jsonapi/wallet/deposit`	POST	application/json

4.1.2 请求参数

字段名	类型	是否必需	说明
walletId	string	是	钱包ID
amount	int64	是	转账金额（正整数）
transactionId	string	是	外部交易ID，需保证唯一性

请求示例:

```
{
    "WalletId": "W-guest-test",
    "Amount": 1000,
    "transactionId": "{{UUID}}"
}
```

4.1.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	object	响应数据对象

data对象字段说明：

字段名	类型	说明
balance	object	余额信息
balance.id	int	保留字段
balance.total	int	总余额
balance.available	int	总可用余额
creditChange	array	积分变动
creditChange[].amount	int	变动金额
creditChange[].metadata	map	元数据

响应示例：

```
{
    "code": 0,
    "message": "",
    "data": {
        "balance": {
            "total": 2010000,
            "available": 2010000
        },
        "creditChange": [
            {
                "amount": 1000000
            }
        ]
    }
}
```

4.1.4 注意事项

1. 所有金额相关的字段均为整数，以最小货币单位计算（分）
2. transactionId 必须保证全局唯一，建议使用 UUID 格式

4.2 钱包提现

从指定钱包提取金额。

4.2.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
wallet.withdraw	`/service/jsonapi/wallet/withdraw`	POST	application/json

4.2.2 请求参数

字段名	类型	是否必需	说明
walletId	string	是	钱包ID
amount	int64	是	提现金额
type	int32	是	提现类型： 0: 全部提出 1: 按amount金额提出
transactionId	string	是	外部交易ID，需保证唯一性

请求示例：

```
{
    "walletId": "W-390",
    "amount": 10,
    "type": 0,
    "transactionId": "{{UUID}}"
}
```

4.2.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	object	响应数据对象

data对象字段说明：

字段名	类型	说明
balance	object	余额信息
balance.id	int	保留字段
balance.total	int	总余额
balance.available	int	总可用余额
balance.credits	array	
balance.credits[].total	int	
balance.credits[].type	int	
balance.credits[].expiresAt	int	
balance.credits[].order	int	
balance.credits[].metadata	map	
creditChange	array	
creditChange[].type	int	
creditChange[].amount	int	
creditChange[].metadata	map	

响应示例：

```
{
    "code": 0,
    "message": "",
    "data": {
        "balance": {},
        "creditChange": [
            {
                "amount": 2010000
            }
        ]
    }
}
```

4.2.4 注意事项

1. 所有金额相关的字段均为整数，以最小货币单位计算（分）
2. transactionId 必须保证全局唯一，建议使用 UUID 格式

4.3 查询钱包余额

查询指定钱包的当前余额。

4.3.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
wallet.query	`/service/jsonapi/wallet/query`	POST	application/json

4.3.2 请求参数

字段名	类型	是否必需	说明
walletId	string	是	钱包ID

请求示例：

```
{
    "WalletId": "W-390"
}
```

4.3.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	object	响应数据对象

data对象字段说明：

字段名	类型	说明
id	int	保留字段
total	int	总余额
available	int	总可用余额

响应示例：

```
{
    "code": 0,
    "message": "",
    "data": {
        "id": "W-390",
        "total": 1000000,
        "available": 1000000
    }
}
```

4.3.4 注意事项

1. 所有金额相关的字段均为整数，以最小货币单位计算（分）
2. transactionId 必须保证全局唯一，建议使用 UUID 格式

5. 锦标赛接口

5.1 锦标赛概述

锦标赛是一个多轮次的积分竞赛系统。每个锦标赛包含一个或多个轮次，每个轮次都有其特定的积分规则和限制条件。玩家通过参与游戏获取积分，根据积分进行排名。

主要特点：

- 多轮次支持：锦标赛可以包含多个轮次，每个轮次可以有不同的规则
- 灵活的积分规则：支持基于下注、获胜和倍数等多种输入参数进行积分计算
- 完善的限制机制：可以设置时间段限制、场次限制和用户限制
- 多样的排名方式：支持累计积分、最高积分和最新积分等多种排名方式

5.2 创建锦标赛

创建一个锦标赛

5.2.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.list	`/service/jsonapi/tournament/create`	POST	application/json

5.2.2 请求参数

字段名	类型	是否必需	说明
version	string	是	配置版本号,当前为"1.0"
name	string	是	锦标赛名称
start	int64	是	锦标赛开始时间戳(秒)
end	int64	是	锦标赛结束时间戳(秒)
rounds	array	是	轮次配置数组

rounds数组中每个轮次对象的字段说明：

字段名	类型	是否必需	说明
name	string	是	轮次名称
start	int64	是	轮次开始时间戳(秒)
end	int64	是	轮次结束时间戳(秒)
initial	object	是	初始积分配置
prize	object	否	奖励配置
limitations	object	否	限制条件配置
strategy	object	是	积分计算规则

initial对象字段说明：

字段名	类型	是否必需	说明
type	string	是	积分类型:"fixed"(固定积分)或"based"(基于排名)
value	int32	是	初始积分值
tiers	array	否	排名积分配置,仅type="based"时需要

prize对象字段说明：

字段名	类型	是否必需	说明
type	string	是	奖励类型,如"credit"
top	int32	是	获奖名次范围
tiers	array	是	奖励等级配置

limitations对象字段说明：

字段名	类型	是否必需	说明
availability	object	否	时间段限制配置
matches	object	否	场次限制配置
user	object	否	用户限制配置

matches限制用户参赛的场次，user限制用户玩的时间，可以参考下面这段配置：

```
'matches': {
    'enabled': true, // 是否启用场次限制
    'limits': [
        { 'type': 'total', 'value': 10 }, // 总共允许比赛局数
        { 'type': 'daily', 'value': 10 }, // 每日最多允许的比赛局数
        { 'type': 'weekly', 'value': 10 }, // 每周最多允许的比赛局数
        { 'type': 'latest', 'days': 3, 'value': 10 }, // 最近几天允许的比赛局数
    ]
},
'user': {
    "enabled": true,
    'time_limit': 600, // 限制用户玩的时间(秒)，从开始时间算起
    'type': 'daily' // 限制类型，默认是全局限制，其他有 "hourly"、"daily"、"weekly"
}
```

strategy对象字段说明：

字段名	类型	是否必需	说明
type	int32	是	积分计算类型:0(累计)、1(最高)、2(最新)
rules	object	是	积分规则配置

rules对象字段说明：

字段名	类型	是否必需	说明
bet	object	否	下注积分规则
win	object	否	获胜积分规则

每个规则对象(bet/win)的字段说明：

字段名	类型	是否必需	说明
enabled	bool	是	是否启用该规则
type	string	是	计算方式:"rate"(比例)或"tiered"(梯度)
rate	float	否	type="rate"时的比例值
tiers	array	否	type="tiered"时的梯度配置

tiers数组中每个对象的字段说明：

字段名	类型	是否必需	说明
range	array	是	数值范围,[min,max]
value	int32/float	是	对应的积分/奖励值

请求示例：

```
{
    "version": "1.0",
    "name": "tourney-test",
    "start": 1735660800,
    "end": 1767196800,
    "rounds": [
        {
            "name": "round-one",
            "start": 1735660800,
            "end": 1767196800,
            "initial": { "type": "fixed", "value": 1000 },
            "prize": {
                "type": "credit",
                "top": 5,
                "tiers": [
                    { "range": [1,1], "value": 100000 },
                    { "range": [2,2], "value": 80000 },
                    { "range": [3,5], "value": 50000 }
                ]
            },
            "strategy": {
                "type": 0,
                "rules": {
                    "bet": { "enabled": true, "type": "rate", "rate": 0.2 },
                    "win": { "enabled": true, "type": "rate", "rate": 1 }
                }
```

```
            },
            "limitations": {
                "user": { "enabled": true, "time_limit": 600, "type": "hourly" }
            }
        }
    ]
}
```

5.2.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	object	锦标赛信息

data数组中的锦标赛对象字段说明：

字段名	类型	说明
id	int64	锦标赛记录ID
tourneyId	string	锦标赛ID，唯一标识符
name	string	锦标赛名称
startTime	int64	锦标赛开始时间戳（秒）
endTime	int64	锦标赛结束时间戳（秒）
rounds	array	轮次列表
rounds[].roundId	string	轮次ID
rounds[].name	string	轮次名称
rounds[].tourneyId	string	锦标赛ID
rounds[].startTime	int64	轮次开始时间戳（秒）
rounds[].endTime	int64	轮次结束时间戳（秒）

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": {
        "tourneyId": "461dd4a3-456d-4b9a-983c-d79a897ca418",
        "name": "tourney-test",
        "startTime": 1735660800,
        "endTime": 1767196800,
        "rounds": [
            {
                "roundId": 1,
                "tourneyId": "461dd4a3-456d-4b9a-983c-d79a897ca418",
                "name": "round-one",
                "startTime": 1735660800,
                "endTime": 1767196800
            }
        ]
    }
}
```


5.2.4 注意事项

- 所有轮次的时间范围必须在锦标赛的整体时间范围内
- 轮次之间的时间段不能重叠
- 使用based类型初始积分的轮次必须有上一轮的排名数据
- 如果启用required_last_round，用户必须参与上一轮才能参加当前轮
- 时间戳均使用秒级Unix时间戳
- 积分计算规则中的rate值表示比例，需要小于等于1
- 奖励配置中的range必须连续且不重叠

5.3 获取锦标赛列表

获取指定渠道的锦标赛列表，支持分页查询。

5.3.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.list	`/service/jsonapi/tournament/list`	POST	application/json

5.3.2 请求参数

字段名	类型	是否必需	说明
offset	int32	是	偏移量，默认为0
limit	int32	是	每页数量限制

请求示例：

```
{
    "offset": 0,
    "limit":10
}
```

5.3.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	array	锦标赛列表数据

data数组中的锦标赛对象字段说明：

字段名	类型	说明
id	int64	锦标赛记录ID
tourneyId	string	锦标赛ID，唯一标识符
name	string	锦标赛名称
rounds	int32	总轮次数量
currentRound	int32	当前轮次序号，从1开始
startTime	int64	锦标赛开始时间戳（秒）
endTime	int64	锦标赛结束时间戳（秒）
channelId	string	渠道ID
status	int32	锦标赛状态： 0: 未开始 1: 已开始 2: 正在关闭 3: 已结束 4: 禁用
createAt	string	记录创建时间，ISO8601格式

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "id": 16,
            "tourneyId": "3a17cc70-296f-472c-958f-68ea1e220927",
            "name": "tourney-one",
            "rounds": 2,
            "currentRound": -1,
            "startTime": 1736352000,
            "endTime": 1736524800,
            "channelId": "chnsz",
            "status": 3
        },
        {
            "id": 14,
            "tourneyId": "3481598c-50fa-49b1-abec-d7734b294d9b",
            "name": "tourney-one",
            "rounds": 2,
            "currentRound": -1,
            "startTime": 1736352000,
            "endTime": 1736524800,
            "channelId": "chnsz",
            "status": 3
        }
    ]
}
```


5.3.4 注意事项

1. 列表默认按创建时间倒序排列
2. 分页参数 offset 如果小于0，会被自动设置为0
3. 返回的锦标赛信息会自动按照当前用户的渠道ID进行过滤
4. config 字段为 JSON 格式字符串，具体配置项根据不同类型的锦标赛有所不同
5. 时间戳字段（start_time、end_time）使用秒级时间戳

5.4 获取锦标赛详情

获取锦标赛详情信息

5.4.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.info	`/service/jsonapi/tournament/info`	POST	application/json

5.4.2 请求参数

字段名	类型	是否必需	说明
tourneyId	string	是	锦标赛ID

请求示例：

```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9"
}
```

5.4.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	array	锦标赛信息

data数组中的锦标赛对象字段说明：

字段名	类型	说明
id	int64	锦标赛记录ID
tourneyId	string	锦标赛ID，唯一标识符
name	string	锦标赛名称
rounds	int32	总轮次数量
currentRound	int32	当前轮次序号，从1开始
startTime	int64	锦标赛开始时间戳（秒）
endTime	int64	锦标赛结束时间戳（秒）
channelId	string	渠道ID
status	int32	锦标赛状态： 0: 未开始 1: 已开始 2: 正在关闭 3: 已结束 4: 禁用
createAt	string	记录创建时间，ISO8601格式
rounds	array	轮次列表
rounds[].roundId	string	轮次ID
rounds[].name	string	轮次名称
rounds[].tourneyId	string	锦标赛ID
rounds[].leaderboardId	string	排行榜ID
rounds[].startTime	int64	轮次开始时间戳（秒）
rounds[].endTime	int64	轮次结束时间戳（秒）
rounds[].createAt	string	创建时间，ISO8601格式

响应示例：


```
{
    "id": 17,
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
    "name": "tourney-test",
    "currentRound": 1,
    "config": "{...}",
    "startTime": 1735660800,
    "endTime": 1767196800,
    "channelId": "chnsz",
    "status": 1,
    "createAt": "2025-01-22T03:06:32+08:00",
    "rounds": [
        {
            "id": 27,
            "name": "round-one",
            "roundId": 1,
            "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
            "leaderboardId": "ade661e9-c021-4522-868c-d9337b90ed0c",
            "startTime": 1735660800,
            "endTime": 1767196800,
            "createAt": "2025-01-22T03:06:32+08:00"
        }
    ]
}
```


5.4.4 注意事项

1. 列表默认按创建时间倒序排列
2. 分页参数 offset 如果小于0，会被自动设置为0
3. 返回的锦标赛信息会自动按照当前用户的渠道ID进行过滤
4. config 字段为 JSON 格式字符串，具体配置项根据不同类型的锦标赛有所不同
5. 时间戳字段（start_time、end_time）使用秒级时间戳

5.5 获取锦标赛关联的游戏列表

获取渠道的关联了锦标赛的游戏列表，支持分页查询。

5.5.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.games	`/service/jsonapi/tournament/games`	POST	application/json

5.5.2 请求参数

字段名	类型	是否必需	说明
offset	int32	是	偏移量，默认为0
limit	int32	是	每页数量限制

请求示例：

```
{
    "offset": 0,
    "limit":10
}
```

5.5.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	array	锦标赛游戏列表

data数组中的锦标赛对象字段说明：

字段名	类型	说明
id	int64	锦标赛记录ID
tourneyId	string	锦标赛ID，唯一标识符
gameId	string	游戏ID
name	string	锦标赛名称
rounds	int32	总轮次数量
currentRound	int32	当前轮次序号，从1开始
startTime	int64	锦标赛开始时间戳（秒）
endTime	int64	锦标赛结束时间戳（秒）
channelId	string	渠道ID
status	int32	锦标赛状态： 0: 未开始 1: 已开始 2: 正在关闭 3: 已结束 4: 禁用
createAt	string	记录创建时间，ISO8601格式

响应示例：


```
{
    "code": 0,
    "message": "",
    "data": [
        {
            "id": 17,
            "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
            "name": "tourney-test",
            "rounds": 1,
            "currentRound": 1,
            "config": "",
            "startTime": 1735660800,
            "endTime": 1767196800,
            "channelId": "",
            "status": 1,
            "createAt": "2025-01-22T03:06:32+08:00",
            "gameId": "slot_zhaocaijinbao"
        }
    ]
}
```


5.5.4 注意事项

1. 列表默认按创建时间倒序排列
2. 分页参数 offset 如果小于0，会被自动设置为0
3. 返回的锦标赛游戏列表会自动按照当前用户的渠道ID进行过滤
4. 时间戳字段（startTime、endTime）使用秒级时间戳

5.6 绑定游戏到锦标赛

将指定游戏绑定到锦标赛

5.6.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.list	`/service/jsonapi/tournament/bind`	POST	application/json

5.6.2 请求参数

字段名	类型	是否必需	说明
tourney_id	string	是	锦标赛ID
game_id	string	是	游戏ID

请求示例：


```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
    "gameId":"slot_zhaocaijinbao"
}
```

5.6.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述

响应示例：

```
{
    "code": 0,
    "message": "success"
}
```

5.6.4 注意事项

1. code不为0，请查看message里面的错误信息

5.7 解绑游戏和锦标赛

将指定游戏从锦标赛中解除绑定关系

5.7.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.list	`/service/jsonapi/tournament/unbind`	POST	application/json

5.7.2 请求参数

字段名	类型	是否必需	说明
tourney_id	string	是	锦标赛ID
game_id	string	是	游戏ID

请求示例：

```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
    "gameId":"slot_zhaocaijinbao"
}
```

5.7.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述

响应示例：

```
{
    "code": 0,
    "message": "success"
}
```

5.7.4 注意事项

1. code不为0，请查看message里面的错误信息

5.8 关闭锦标赛

关闭指定的锦标赛

5.8.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.close	`/service/jsonapi/tournament/close`	POST	application/json

5.8.2 请求参数

字段名	类型	是否必需	说明
tourneyId	string	是	锦标赛ID

请求示例：

```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9"
}
```

5.8.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述

响应示例：

```
{
    "code": 0,
    "message": "success"
}
```

5.9 删除锦标赛

删除指定的锦标赛

5.9.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.delete	`/service/jsonapi/tournament/delete`	POST	application/json

5.9.2 请求参数

字段名	类型	是否必需	说明
tourneyId	string	是	锦标赛ID

请求示例：

```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9"
}
```

5.9.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述

响应示例：

```
{
    "code": 0,
    "message": "success"
}
```

5.10 获取锦标赛排名

获取锦标赛排行榜信息

5.10.1 请求地址

接口名称	URL地址	HTTP方法	Content-Type
tournament.ranks	`/service/jsonapi/tournament/ranks`	POST	application/json

5.10.2 请求参数

字段名	类型	是否必需	说明
tourneyId	string	是	锦标赛ID
currentRound	int32	是	当前轮次序号，从1开始
offset	int32	是	偏移量，默认为0
limit	int32	是	每页数量限制

请求示例：

```
{
    "tourneyId": "8e17e7e5-afd5-4abf-8773-6f5d3638bff9",
    "currentRound": 1,
    "offset": 0,
    "limit":10
}
```

5.9.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	array	锦标赛排名数据

data数组中的锦标赛对象字段说明：

字段名	类型	说明
id	int64	已结束的排行榜 ID
leaderboardId	string	排行榜 ID
player	string	玩家 ID
score	float64	玩家积分
rank	int64	玩家排名
createAt	string	创建时间（ISO8601 格式）

响应示例：

```
{
    "code": 0,
    "message": "success"
}
```

6. 用户接口

6.1 注册登录用户

您必须在启动游戏之前在我们的游戏平台数据库中创建玩家或将其登录到我们的游戏平台。
如果用户名不存在，则会创建一个新玩家。
如果用户名已存在且密码匹配，玩家将被登录，并返回一个会话令牌（Session Token）。您将使用此返回的令牌来启动游戏。

6.1.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
user.login	`/service/jsonapi/user/login`	POST	application/json

6.1.2 请求参数

字段名	类型	是否必需	说明
identifier	string	是	用户Id(最大128字符)
credential	string	是	用户密码(最大256字符)
authType	string	否	认证方式：默认是账号密码
linkId	string	否	链接账号ID，如果是账号密码登录，则不需要传

请求示例：

```
{
    "identifier": "guest-test",
    "credential": "123456"
}
```

6.1.3 响应参数

字段名	类型	说明
code	int	返回码，0表示成功，其他表示失败
message	string	信息描述
data	object	用户数据

data数组中的锦标赛对象字段说明：

字段名	类型	说明
accessToken	string	用户令牌
userId	string	用户ID
groupId	string	用户所属的渠道ID
expireTime	int64	令牌失效时间
walletId	string	钱包ID
walletBalance	int64	钱包余额

6.2 修改用户信息

更新用户的密码信息

6.2.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
user.update	`/service/jsonapi/user/update`	POST	application/json

6.2.2 请求参数

字段名	类型	是否必需	说明
identifier	string	是	用户Id(最大128字符)
credential	string	是	用户密码(最大256字符)

请求示例：

```
{
    "identifier": "guest-test",
    "credential": "123456"
}
```

6.2.3 响应参数

字段名	类型	说明
code	int	返回码，0 表示成功，其他表示失败
message	string	信息描述

响应示例:

```
{
    "code": 0,
    "message": "成功"
}
```

6.3 注销用户

更新用户的密码信息

6.3.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
user.logout	`/service/jsonapi/user/logout`	POST	application/json

6.3.2 请求参数

字段名	类型	是否必需	说明
identifier	string	是	用户Id(最大128字符)
credential	string	是	用户密码(最大256字符)

请求示例:

```
{
    "identifier": "guest-test",
    "credential": "123456"
}
```

6.3.3 响应参数

字段名	类型	说明
code	int	返回码，0 表示成功，其他表示失败
message	string	信息描述

响应示例:

```
{
    "code": 0,
    "message": "成功"
}
```

7. Webhook 接口

Webhook 是一种允许应用程序实时接收事件通知的机制。通过 Webhook，第三方系统可以在特定事件发生时，向预先配置的 URL 发送 HTTP 请求，从而触发相应的处理逻辑。Webhook 通常用于实现系统间的异步通信，例如在钱包变动时，及时通知相关第三方系统。

Webhook 的核心优势在于其实时性和灵活性。相比传统的轮询机制，Webhook 能够在事件发生时立即通知订阅者，减少了资源浪费和延迟。同时，Webhook 支持多种事件类型，订阅者可以根据业务需求选择性地订阅特定事件。

通过 Webhook 事件通知机制，第三方系统可以实时获取关键事件信息，从而快速响应业务变化，提升系统的整体效率和用户体验。

7.1 订阅事件

第三方系统可以通过订阅事件通知来接收特定事件的发生信息。订阅时需指定 Webhook 的接收地址以及需要订阅的事件类型列表。

7.1.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
webhook.subscribe	`/service/webhook/subscribe`	POST	application/json

7.1.2 请求参数

字段名	类型	是否必需	说明
url	string	是	Webhook 接收地址
events	[]string	是	订阅的事件类型列表

请求示例:

```
{
    "url": "https://example.com/webhook/receiver",
    "events": ["wallet.balance.update", "wallet.transaction.success"]
}
```

7.1.3 响应参数

字段名	类型	是否必需	说明
code	int	是	返回码，0 表示成功，其他表示失败
message	string	是	信息描述

响应示例:

```
{
    "code": 0,
    "message": "订阅成功"
}
```

7.2 取消订阅

第三方系统可以通过取消订阅接口停止接收所有事件通知。取消订阅后，系统将不再向该 URL 发送任何事件通知。

7.2.1 请求地址

接口名称	URL 地址	HTTP 方法
webhook.unsubscribe	`/service/webhook/unsubscribe`	GET

7.2.2 请求参数

无

7.2.3 响应参数

字段名	类型	是否必需	说明
code	int	是	返回码，0 表示成功，其他表示失败
message	string	是	信息描述

响应示例:

```
{
    "code": 0,
    "message": "取消订阅成功"
}
```

7.3 发布事件

发布事件接口用于主动触发特定事件，并将事件数据发送给所有订阅了该事件的 Webhook 接收端。通过此接口，第三方可以触发事件通知，确保订阅者能够及时获取最新的业务状态变化。

7.3.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
webhook.publish	`/service/webhook/publish`	POST	application/json

7.3.2 请求参数

字段名	类型	是否必需	说明
event	string	是	通知的事件名称
payload	map	是	事件内容

请求示例:

```
{
    "event": "wallet.balance.update",
    "payload": {
        "wallet_id": "wallet-001",
        "balance": 1000.50,
        "currency": "USD"
    }
}
```

7.3.3 响应参数

字段名	类型	是否必需	说明
code	int	是	返回码，0 表示成功，其他表示失败
message	string	是	信息描述

7.4 第三方处理事件

当订阅的事件触发时，系统会向第三方配置的 Webhook URL 发送事件通知。第三方系统需要正确处理事件通知并返回响应。

7.4.1 请求地址

接口名称	URL 地址	HTTP 方法	Content-Type
webhook.event	`用户自定义的 Webhook URL`	POST	application/json

7.4.2 请求参数

字段名	类型	是否必需	说明
subscriberId	int	是	订阅 ID
event	string	是	通知的事件名称
createdAt	string	是	事件发生的时间（ISO8601 格式）
payload	map	是	事件内容，每个事件的内容不一致，请参考具体的事件文档

请求示例:

```
{
    "subscriberId": 123,
    "event": "wallet.balance.update",
    "createdAt": "2023-10-01 12:00:00",
    "payload": {
        "walletId": "wallet-001",
        "balance": 1000.50,
        "currency": "USD"
    }
}
```

7.4.3 响应参数

字段名	类型	是否必需	说明
code	int	是	返回码，0 表示成功，其他表示失败
message	string	是	信息描述

响应示例:

```
{
    "code": 0,
    "message": "事件处理成功"
}
```

7.4.4 注意事项

1. **安全性**:
    - 确保 Webhook 接收端点的安全性，建议使用 HTTPS 协议。
    - 对请求进行身份验证，例如通过签名或 Token 验证请求来源。
2. **重试机制**:
    - 如果 Webhook 接收端点在接收到通知后未能成功处理，服务端可能会进行多次重试。
    - 建议接收端点实现幂等性处理，以避免重复处理相同事件。
3. **事件类型**:
    - 确保订阅的事件类型与业务需求匹配，避免不必要的通知。
    - 支持的事件类型请参考本文档后面的通知接口
4. **响应超时**:
    - Webhook 接收端应在 5 秒内返回响应，否则服务端可能会认为请求失败并触发重试机制。
5. **错误处理**:
    - 如果 Webhook 接收端返回的 `code` 不为 0，服务端会记录错误日志并触发重试。
    - 建议接收端记录错误信息以便排查问题。
6. **日志记录**:
    - 建议在 Webhook 接收端记录所有接收到的请求和响应，便于后续审计和问题排查。