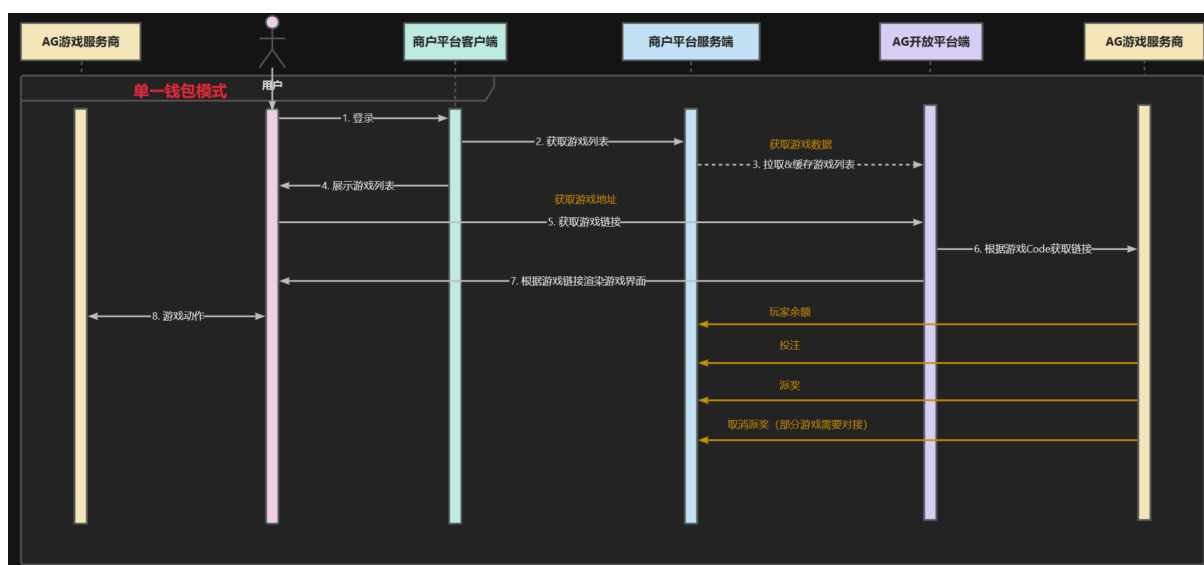


AG游戏开放平台接口文档-V5.0.0(单一钱包)

1. 业务流程



2. 接口说明

2.1. 接口统一为http post请求, 请求入参类型为application/json, reqTraceId为请求唯一标识, 跟随参数一同传递

2.2. 请求 Header

请求AG平台的所有接口, 都需要加签, AG 平台的所有接口, 都需要验签。加签字段位于 header 当中:

headerName	参数值	数据类型	示例	备注
------------	-----	------	----	----

X-MERCHANT-CODE	商户code	varchar(32)	5ae3e2a24b5b428896a9038bedd1c396	从 AG 平台获取
X-TIMESTAMP	当前请求时间戳 (utc时间)	bigint	1744897560043	
X-NONCE	随机字符串	varchar(32)	7c83a1c5f989498d8c73a2d54d66657f	
X-SIGN	接口请求数据签名	varchar(256)	920fb61116e04a9e2c09264399960a91db04ff35cd10018fc1b6d87a25a10165	具体签名规则参见 2.4 签名规则
X-CONTENT-PROCESSING-TYPE	签名算法类型	varchar(64)	HmacSHA256	固定传 HmacSHA256

2.3. 请求域名

AG游戏域名分测试环境和线上环境, 具体域名参照AG平台的开户文档

2.4. 签名规则

1. 商户在平台获取秘钥
2. 将以下字段按照字段序号排序, 拼接成加签字符串

字段序号	字段名称	备注
1	merchanrCode	商户编码, 平台方获取
2	timestamp	当前时间戳, 要求和请求头中保持一致
3	nonce	随机字符串, 要求和请求头中保持一致

4	signType	签名算法, 要求和请求头中保持一致
5	body	实际传输数据(如果存在的话) <ol style="list-style-type: none"> 1. 数据传输格式为json 2. 签名数据为统一结果包装后的数据 3. 加签数据一定要和实际发送数据一致(每个字符, 每个字节, 完全一致)

- 例如: 商户 Code 为 123123, 当前时间戳为 1744897560043, 随即字符串为 7c83a1c5f989498d8c73a2d54d66657f, 签名算法为 HmacSHA256, 请求体为 {"merchantSession": "9259c473c8a94f68a7ec1e766d33b297", "currencyCode": "BR_BRL"}
 - 拼接后的加签字符串为 12312317448975600437c83a1c5f989498d8c73a2d54d66657fHmacSHA256{"merchantSession": "9259c473c8a94f68a7ec1e766d33b297", "currencyCode": "BR_BRL"}
3. 使用 HmacSHA256 算法与商户密钥对其进行签名。商户密钥在开户时由 AG 提供, 请妥善存储。
- 以上述加签字符串为例, 签名后的结果为: 920fb61116e04a9e2c09264399960a91db04ff35cd10018fc1b6d87a25a10165
4. 将所有参与签名的字段放入请求 Header 或 Body 当中。
5. 签名生成示例 (Java 代码) :

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.net.URI;
import java.net.http.HttpClient;
```

```

import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.HexFormat;

public class Demo {

    static String apiUrl = "http://xxxxx.xxxxx.xxxxx/xxxxx/game/v3/game/launchUrl";
    static String merchantCode = "xxxxxxxxxxxxx";
    static String secretKey = "xxxxxxxxxxxxx";

    public static void main(String[] args) throws Exception {
        postJson(
            ""
            {
                "gameCode": "126",
                "playerSession": "playerSession-U10000001",
                "playerUniqueld": "U10000001",
                "currencyCode": "US_USD",
                "language": "en",
                "rtp": 96,
                "balance": "9999.00",
                "subMerchantCode": "site",
                "terminalType": "PC",
                "returnUrl": "https://www.example.com/"
            }
            ""
        );
    }

    public static void postJson(String jsonBody) throws Exception {

        String timestamp = String.valueOf(System.currentTimeMillis());
        String nonce = String.valueOf(Math.random());
        String contentProcessingType = "HmacSHA256";

        String spliceResult = merchantCode + timestamp + nonce + contentProcessingType;

        String sign = hmacSHA256Sign(secretKey, spliceResult + jsonBody);
    }
}

```

```

HttpClient client = HttpClient.newHttpClient();

HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(apiUrl))
    .header("Content-Type", "application/json")

    .header("X-MERCHANT-CODE", merchantCode)
    .header("X-TIMESTAMP", timestamp)
    .header("X-NONCE", nonce)
    .header("X-SIGN", sign)
    .header("X-CONTENT-PROCESSING-TYPE", contentProcessingType)

    .POST(HttpRequest.BodyPublishers.ofString(jsonBody, StandardCharsets.UTF_8))
    .build();

HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

System.out.println("Response Code: " + response.statusCode());
System.out.println("Response Body: " + response.body());
}

public static String hmacSHA256Sign(String secretKey, String data) throws Exception {
    Mac mac = Mac.getInstance("HmacSHA256");
    SecretKeySpec secretKeySpec = new
SecretKeySpec(secretKey.getBytes(StandardCharsets.UTF_8), "HmacSHA256");
    mac.init(secretKeySpec);
    return hexString(mac.doFinal(data.getBytes(StandardCharsets.UTF_8)));
}

public static String hexString(byte[] data) {
    return HexFormat.of().formatHex(data);
}
}

```

2.5. ag平台接口返回的标准格式如下：

```
{
```

```
"code": C10000,
"msg": "success",
"data": {
}
}

// 错误示例
{
  "success": false,
  "code": "C10005",
  "msg": "签名错误",
  "data": null
}
```

参数名	描述	数据类型	是否必须
code	错误码	String	是
msg	错误信息	String	是
data	返回的数据体	Sting	是

2.6. 错误码说明

Code	Msg
C10000	Request succeeded
C10001	基础服务异常
C10002	请求参数错误
C10003	无效的请求头
C10004	签名错误

C20001	商户Code不存在
G10001	游戏服务异常
G20001	playerId为空
G20002	游戏ID不存在
G20003	游戏已下架
G30001	当前游戏用户会话已过期
G30002	商户余额不足
G30003	玩家余额不足
G40001	三方服务异常

2.7. 货币编码说明

地区	货币编码(currency)
巴西	BR_BRL
英国	GB_GBP
丹麦	DK_DKK
德国	DE_EUR

西班牙	ES_EUR
芬兰	FI_EUR
法国	FR_EUR
印度尼西亚	ID_IDR
意大利	IT_EUR
荷兰	NL_EUR
挪威	NO_NOK
波兰	PL_PLN
葡萄牙	PT_EUR
泰国	TH_THB
土耳其	TR_TRY
越南	VN_VND
缅甸	MM_MMK
新加坡	SG_SGD
尼日利亚	NG_NGN
美国	US_USD
巴基斯坦	PK_PKR
哥伦比亚	CO_COP
菲律宾	PH_PHP

墨西哥	MX_MXN
阿根廷	AR_ARS
肯尼亚	KE_KES
西班牙	ES_EUR
智利	CL_CLP
南非	ZA_ZAR

2.8. 语言编码

语言代码(lang)	中文语言名称
en	英语
ar	阿拉伯语
az	阿塞拜疆语
bg	保加利亚语
bn	孟加拉语
cs	捷克语
da	丹麦语
de	德语
el	希腊语
en-stkus	英语(STKUS变体, 注意:这不是一个广泛认可的语言代码)

es	西班牙语
et	爱沙尼亚语
fa	波斯语
fi	芬兰语
fr	法语
hi	印地语ā
hu	匈牙利语
hy	亚美尼亚语
id	印度尼西亚语
it	意大利语
ja	日语
ko	韩语
lo	老挝语
lt	立陶宛语
mn	蒙古语
my	缅甸语
ms	马来西亚语
nl	荷兰语
no	挪威语

pl	波兰语
pt	葡萄牙语
pt-br	巴西葡萄牙语
ro	罗马尼亚语
ru	俄语
sh	塞尔维亚-克罗地亚语(或波斯尼亚-克罗地亚-塞尔维亚语)
si	锡兰语(或僧伽罗语)
sk	斯洛伐克语
sq	阿尔巴尼亚语
sv	瑞典语
th	泰语
tr	土耳其语
uk	乌克兰语
ur	乌尔都语
uz	乌兹别克语
vi	越南语

2.9. 商户需要主动对接实现的接口

[1、第三节:游戏API 3.1~3.5](#)

[2、第五节:调控RTP-API 5.1~5.4](#)

[3、第六节:交易相关api 6.1](#)

[4、第七节:商户相关api 7.1](#)

2.10.商户平台需提供接口实现,由**AG**平台主动回调

[1、第四节:单一钱包API 4.1~4.5](#)

3. 游戏**API**(**AG**平台方提供)

3.1. 游戏厂商(无需对接)

3.1.1. 接口地址

POST /game/v5/providers

3.1.2. 请求参数

说明:该接口无需传入参数,直接调用即可获取所有游戏提供商信息

```
{
  "reqTraceId": "TRACE_XXX"
}
```

3.1.3. 响应结果

```
{
  "code": "C10000",
  "msg": "成功",
  "data": {
    "providers": [
      {
        "name": "PP EN",
        "currencyCode": "PHP,BRL",
        "code": "PP",
        "categoryCode": "SLOTS,LIVE"
      }
    ]
  }
}
```

3.1.4. 字段说明

字段名称	类型	描述
name	string	提供商名称
currencyCode	string	支持的币种(多个用逗号分隔)
code	string	提供商编码
categoryCode	string	支持的分类(多个用逗号分隔)

3.1.5. 成功响应示例

```
{
  "code": "C10000",
  "msg": "成功",
  "data": {
    "providers": [
      {
        "name": "PP EN",
        "currencyCode": "PHP,BRL",
        "code": "PP",
        "categoryCode": "SLOTS,LIVE"
      }
    ]
  }
}
```

3.2. 游戏分类

3.2.1. 接口地址

GET /game/v5/categories

3.2.2. 请求参数

```
{
  "reqTraceId": "TRACE_XXX",
  "currencyCode": "BR_BRL",
  "lang": "pt"
}
```

3.2.3. 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	可选	币种编码, 如 BR_BRL
lang	string	可选	语言编码, 如 pt(葡萄牙语)

3.2.4. 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "categoryDataList": [
      {
        "categoryName": "AG_Hot PG",
        "categoryCode": "AG_HOT_PG"
      }
    ],
    "totalSize": 12
  }
}
```

3.2.5. 字段说明

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据
totalSize	integer	必填	分类总数

providers数组项说明:

字段名	类型	说明
categoryName	string	分类名称
categoryCode	string	分类编码

3.2.6. 成功响应示例

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "categoryDataList": [
      {
        "categoryName": "AG_Hot PG",
        "categoryCode": "AG_HOT_PG"
      },
      {
        "categoryName": "AG_New PG",
        "categoryCode": "AG_NEW_PG"
      }
    ],
    "totalSize": 2
  }
}
```

3.2.7. 失败响应示例

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

3.3 游戏数据

3.3.1 接口地址

GET /game/v5/games

3.3.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "categoryCode": "AG_HOT_PG",
  "currencyCode": "BR_BRL",
  "lang": "pt",
  "pageIndex": 1
}
```

3.3.3 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	可选	币种编码, 如BR_BRL
lang	string	必填	语言编码, 如pt(葡萄牙语)
categoryCode	string	必填	分类编码, 如AG_HOT_PG
pageIndex	int	必填	页码

3.3.4 响应结果

js

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "categoryCode": "AG_HOT_PG",
    "gameItemList": [
      {
        "gameName": "Fortune Dragon",
        "gameCode": "FORTUNE_DRAGON",
        "status": 1,
        "gameType": "SLOT"
      }
    ],
    "totalSize": 15
  }
}
```



```
}
```

3.3.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据
categoryCode	string	必填	分类编码
totalSize	integer	必填	该游戏分类下的游戏总数

gameItemList 数组项说明

字段名	类型	说明
gameName	string	游戏名称
gameCode	string	游戏ID
status	int	游戏状态, 0-下线, 1-上线, 2-维护中
gameType	string	游戏类型

3.3.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "categoryCode": "AG_HOT_PG",
```

```
"gameItemList": [  
  {  
    "gameName": "Fortune Dragon",  
    "gameCode": "FORTUNE_DRAGON",  
    "status": 1,  
    "gameType": "SLOT"  
  },  
  {  
    "gameName": "Fortune Tiger",  
    "gameCode": "FORTUNE_TIGER",  
    "status": 1,  
    "gameType": "SLOT"  
  }  
],  
"totalSize": 2  
}
```

3.3.7 错误响应示例

json

```
{  
  "success": false,  
  "code": "C40000",  
  "msg": "参数错误",  
  "data": null  
}
```

3.3.8 使用场景

- 获取指定分类下的游戏列表
- 构建游戏分类页面的游戏展示列表

3.4 强制玩家下线

3.4.1 接口地址

text

POST /game/v5/player/force/logout

3.4.2 请求参数

json

```
{  
  "reqTraceId": "TRACE_XXX",  
  "playerId": "PLAYER_123456"  
}
```

3.4.3 参数说明

参数名	类型	可选/必填	说明
playerId	string	必填	玩家ID

3.4.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "playerId": "PLAYER_123456",
    "logoutStatus": "SUCCESS",
    "message": "Player forced logout successfully"
  }
}
```

3.4.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
playerId	string	必填	玩家ID

logoutStatus	string	必填	下线状态 (SUCCESS/FAILED)
message	string	可选	操作结果描述

3.4.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "playerId": "PLAYER_123456",
    "logoutStatus": "SUCCESS",
    "message": "Player forced logout successfully"
  }
}
```

3.4.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

3.4.8 使用场景

- 管理员强制玩家下线
- 玩家账户异常时的安全控制
- 系统维护前的用户清理

3.5 获取游戏链接

3.5.1 接口地址

text

POST /game/v5/game/url

3.5.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "gameCode": "FORTUNE_DRAGON",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL",
  "language": "pt",
  "terminalType": "PHONE",
  "returnUrl": "https://merchant.com/return",
  "ipAddress": "192.168.1.1",
  "subMerchantCode": "SUB_MERCHANT_001",
  "nickName": "玩家昵称",
  "avatarUrl": "https://example.com/avatar.jpg"
}
```

3.5.3 参数说明

参数名	类型	可选/必填	说明
gameCode	String	必填	游戏code
playerId	String	必填	商户玩家唯一id
currencyCode	String	必填	货币编码, 详情见 1.7 货币编码说明
language	String	必填	语言
terminalType	String	可选	游戏终端类型, 取值范围: PHONE / PC, 默认为 PHONE
returnUrl	String	可选	运营商网站的 URL, 用于游戏里的返回按钮链接
ipAddress	String	必填	用户所在地的IP地址, 可以是IPv4或IPv6格式
subMerchantCode	String	可选	子商户编码, 不能包含下横线

nickName	string	可选	玩家昵称
avatarUrl	string	可选	玩家头像URL

returnUrl 说明：

利用 returnUrl 关闭游戏并接收自定义参数，需要游戏平台前端实现对应功能
returnUrl 对应游戏平台的一个中间页面和游戏平台同源可直接访问游戏平台JS代码，实现退出游戏

示例：

假设游戏平台地址：<https://www.demo.com>

假设游戏平台页面为：

```
<html>
<head>
  .....
  <title></title>
</head>
<body>
  .....
  // 游戏iframe
  <iframe src="gameUrl"></iframe>
  // 游戏退出时iframe会重定向到returnUrl, returnUrl中的页面会来调用平台退出游戏的方法
  .....
  <script>
    // 假设游戏平台原有的退出游戏方法为exitGame
    window.exitGame = function (params) {
      // 关闭游戏iframe
    };
  </script>
  .....
</body>
</html>
```

在游戏平台前端增加一个中间页面例如：<https://www.demo.com/exitGame>

```
<html>
<head>
  .....
  <title></title>
</head>
<body>
<script>
  // 游戏退出时，当前页面加载到游戏平台iframe中
  // 和游戏平台同源，直接调用游戏平台原有的退出游戏方法
  // 假设游戏平台原有的退出游戏方法为exitGame
  window.parent.exitGame();

  // 可携带调用接口时自定义的search
  // window.parent.exitGame(new URL(window.location.href).searchParams);
</script>
</body>
</html>
```

3.5.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "gameCode": "FORTUNE_DRAGON",
    "playerId": "PLAYER_123456",
    "gameUrl": "https://game.example.com/play?token=abc123"
  }
}
```

3.5.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
gameCode	string	必填	游戏code
playerId	string	必填	商户玩家唯一id
url	string	必填	游戏链接地址
expireTime	string	可选	链接过期时间

3.5.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "url": "https://game.example.com/play?token=abc123&lang=pt",
    "expireTime": "2024-12-25T10:30:00Z"
  }
}
```

3.5.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

3.5.8 使用场景

- 玩家进入游戏时获取游戏链接
- 根据不同参数生成定制化游戏入口
- 支持多语言多货币的游戏访问

4. 单一钱包API(商户型需要实现)

1. 商户开户选择<单一钱包>模式需要实现以下接口供AG平台调用
2. 商户在开户时提供回调地址的域名, 则实现回调URL的全路径为:回调地址域名 + / url地址
3. 商户需要按照平台统一<响应格式>返回回调处理结果
4. 接口统一返回包装 (以下需要实现接口返回结果均使用该类型进行包装)
 - a. 接口返回的Content-Type 要求类型为: application/json

4.1 玩家余额

4.1.1 接口地址

text

POST {MERCHANT-URL}/wallet/balance

4.1.2 请求参数

json

```
{
  "reqTraceId": "xxxx",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL"
}
```

4.1.3 参数说明

参数名	类型	可选/必填	说明
reqTraceId	string	必填	请求追踪id
playerId	string	必填	玩家ID, 如 PLAYER_123456
currencyCode	string	必填	货币编码, 如 BR_BRL

4.1.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "balance": "1000.00"
  }
}
```

4.1.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功

code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
balance	string	必填	玩家余额

4.1.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "balance": "1000.00"
  }
}
```

4.1.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

4.1.8 使用场景

- 查询玩家当前余额信息

4.2 玩家信息

4.2.1 接口地址

text

POST {MERCHANT-URL}/player/info

4.2.2 请求参数

json

```
{
  "reqTraceId": "xxxx",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL"
}
```

4.2.3 参数说明

参数名	类型	可选/必填	说明
reqTraceId	string	必填	请求追踪id
playerId	string	必填	玩家ID, 如 PLAYER_123456
currencyCode	string	必填	货币编码, 如 BR_BRL

4.2.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "nickName": "PlayerOne",
    "avatarUrl": "https://example.com/avatar/playerone.png"
  }
}
```

4.2.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功

code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
nickName	string	必填	玩家昵称
avatarUrl	string	必填	玩家头像URL

4.2.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "nickName": "PlayerOne",
    "avatarUrl": "https://example.com/avatar/playerone.png"
  }
}
```

4.2.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

4.2.8 使用场景

- 获取玩家基本资料用于游戏界面展示

4.3 投注(无需对接)

4.3.1 接口地址

text

POST {MERCHANT-URL}/wallet/bet

4.3.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL",
  "gameCode": "FORTUNE_DRAGON",
  "roundId": "ROUND_789012",
  "betId": "BET_345678",
  "betAmount": "100.00",
  "timestamp": "2024-12-25T10:30:00Z",
  "transactionId": "TXN_789012"
}
```

4.3.3 参数说明

参数名	类型	可选/必填	说明
reqTraceId	string	必填	追踪ID, 如 TRACE_XXX
playerId	string	必填	玩家ID, 如 PLAYER_123456
currencyCode	string	必填	货币编码, 如 BR_BRL
gameCode	string	必填	游戏编码, 如 FORTUNE_DRAGON
roundId	string	必填	牌局ID, 如 ROUND_789012

betId	string	必填	订单ID, 如 BET_345678
betAmount	string	必填	投注金额, 如 100.00
timestamp	string	必填	交易时间(时区), 如 2024-12-25T10:30:00Z
transactionId	string	必填	派奖订单号(每笔记录唯一), 如 TXN_789012

4.3.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantBetId": "MBET_987654",
    "balance": "900.00"
  }
}
```

4.3.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功

msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
merchantBetId	string	必填	商户投注订单号
balance	string	必填	商户最新余额

4.3.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantBetId": "MBET_987654",
    "balance": "900.00"
  }
}
```

4.3.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

4.3.8 使用场景

- 玩家在游戏中进行投注操作
- 实时扣减玩家账户余额
- 生成投注订单记录

4.4 派奖

4.4.1 接口地址

text

POST {MERCHANT-URL}/wallet/win

4.4.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL",
  "gameCode": "FORTUNE_DRAGON",
  "roundId": "ROUND_789012",
  "transactionId": "TXN_789012",
  "betId": "BET_345678",
  "betAmount": "100.00",
  "winAmount": "500.00",
  "isFree": false,
  "isEnd": true,
  "betTime": 1703498999000,
  "settledTime": 1703499000000,
  "type": "bet_win"
}
```

4.4.3 参数说明

参数名	类型	可选/必填	说明
playerId	string	必填	玩家ID, 如PLAYER_123456
currencyCode	string	必填	货币编码, 如BR_BRL
gameCode	string	必填	游戏编码, 如FORTUNE_DRAGON
transactionId	string	必填	派奖订单号(每笔记录唯一), 如TXN_789012
roundId	string	必填	牌局ID, 如ROUND_789012

betId	string	必填	投注订单号, 和投注单进行关联。当不存在单独的投注单时, 与派奖单号相同
betAmount	string	必填	投注金额, 如100.00
winAmount	string	必填	用户派奖金额, 如500.00
isFree	boolean	必填	用于指示下注为免费旋转下注的状态
isEnd	boolean	必填	本轮牌局是否结束
betTime	long	必填	此交易的初始请求的Unix时间戳(毫秒)
settledTime	long	必填	此交易的结算时间的Unix时间戳(毫秒)
type	string	必填	类型: win(仅派奖)或bet_win(投注与派奖合并)

4.4.4 注意事项

运营商需要实现该接口, 该接口用于回传玩家游戏派奖信息。

注意: 部分快节奏游戏(如 SLOT 类) 的投注与派奖合并在该派奖信息回调当中:

- 当 type 为 win 仅派奖时, 说明该投注信息已经通过投注信息回调接口通知过, 无需扣除玩家投注金额。
- 当 type 为 bet_win 时, 说明该派奖信息合并了投注的数据, 需要一同处理玩家的投注与派奖两类信息

4.4.5 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantBetId": "MBET_987654",
    "balance": "1400.00"
  }
}
```

```
}  
}
```

4.4.6 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
merchantBetId	string	必填	商户投注订单号
balance	string	必填	商户最新余额

4.4.7 成功响应示例

json

```
{  
  "success": true,  
  "code": "C10000",  
  "msg": "Request succeeded",  
  "data": {  
    "merchantBetId": "MBET_987654",  
    "balance": "1400.00"  
  }  
}
```

4.4.8 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
{
  "code": "G30003",
  "msg": "余额不足",
}
```

4.4.9 使用场景

- 运营商接收游戏派奖信息
- 处理玩家投注与派奖合并的快节奏游戏
- 实时更新玩家账户余额
- 记录游戏交易流水

4.5 取消派奖|投注(无需对接)

4.5.1 接口地址

text

POST {MERCHANT-URL}/wallet/cancel

4.5.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "playerId": "PLAYER_123456",
  "currencyCode": "BR_BRL",
  "gameCode": "FORTUNE_DRAGON",
  "betId": "BET_345678",
  "merchantBetId": "MBET_987654",
  "betAmount": "100.00",
  "cancelType": "BET",
  "transactionId": "TXN_789012"
}
```

4.5.3 参数说明

参数名	类型	可选/必填	说明
playerId	string	必填	玩家ID, 如PLAYER_123456
currencyCode	string	必填	货币编码, 如BR_BRL
gameCode	string	必填	游戏编码, 如 FORTUNE_DRAGON
betId	string	必填	投注订单号
merchantBetId	string	必填	商户投注单号
betAmount	string	必填	投注金额, 如100.00
cancelType	string	必填	取消类型: BET(取消投注)/WIN(取消派奖)
transactionId	string	必填	派奖订单号(每笔记录唯一), 如 TXN_789012

4.5.4 注意事项

仅部分游戏存在取消操作

4.5.5 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantBetId": "MBET_987654",
    "betId": "BET_345678",
    "balance": "1000.00",
    "currencyCode": "BR_BRL"
  }
}
```

4.5.6 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
merchantBetId	string	必填	商户投注单号
betId	string	必填	投注订单号
balance	string	必填	商户最新余额
currencyCode	string	必填	货币编码

4.5.7 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantBetId": "MBET_987654",
    "betId": "BET_345678",
    "balance": "1000.00",
    "currencyCode": "BR_BRL"
  }
}
```

4.5.8 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

4.5.9 使用场景

- 处理游戏中的取消操作
- 恢复玩家账户余额
- 维护游戏交易记录一致性

5. 调控RTP-API

5.1 修改子商户RTP

5.1.1 接口地址

text

POST /game/v5/set/sub/merchant/rtp

5.1.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "subMerchantCode": "SUB_MERCHANT_001",
  "rtp": 95,
  "currencyCode": "BR_BRL"
}
```

5.1.3 参数说明

参数名	类型	可选/必填	说明
subMerchantCode	string	必填	子商户编码, 如 SUB_MERCHANT_001

rtp	integer	必填	RTP修改值, 如95
currencyCode	string	必填	货币编码, 如 BR_BRL

5.1.4 响应结果

json

<pre>{ "success": true, "code": "C10000", "msg": "Request succeeded", "data": { "changedRtpValue": 95 } }</pre>

5.1.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
changedRtpValue	integer	必填	RTP修改值

5.1.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "changedRtpValue": 95
  }
}
```

5.1.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

5.1.8 使用场景

- 调整子商户游戏RTP值
 - 根据运营策略动态调整游戏回报率
-

5.2 修改用户RTP

5.2.1 接口地址

text

POST /game/v5/set/player/rtp

5.2.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "currencyCode": "BR_BRL",
  "rtp": 96,
  "playerId": "PLAYER_123456"
}
```


5.2.3 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	必填	货币编码, 如 BR_BRL
rtp	integer	必填	RTP值, 如96
playerId	string	必填	玩家ID, 如 PLAYER_123456

5.2.4 注意事项

商户传递的玩家ID需要保证唯一, 平台会将相同玩家ID视作同一个玩家数据

5.2.5 响应结果

json

<pre>{ "success": true, "code": "C10000", "msg": "Request succeeded", "data": { "changedRtpValue": 96 } }</pre>

5.2.6 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
changedRtpValue	integer	必填	修改后的RTP值

5.2.7 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "changedRtpValue": 96
  }
}
```

5.2.8 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

5.2.9 使用场景

- 为特定玩家设置个性化RTP值
- 测试玩家游戏体验调整
- 根据玩家行为优化游戏回报率

5.3 查看生效RTP

5.3.1 接口地址

text

GET /game/v5/rtp/effective

5.3.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "currencyCode": "BR_BRL",
  "subMerchantCode": "SUB_MERCHANT_001",
  "playerId": "PLAYER_123456"
}
```

5.3.3 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	必填	货币编码，如BR_BRL
subMerchantCode	string	可选	子商户编码，如SUB_MERCHANT_001
playerId	string	可选	玩家ID，如PLAYER_123456

5.3.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "effectiveType": "PLAYER",
    "effectiveRtpValue": 96,
    "merchantRtpValue": 95,
    "subMerchantRtpValue": 95,
    "playerRtpValue": 96
  }
}
```

5.3.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
effectiveType	string	必填	当前生效类型(MERCHANT/商户, SUB_MERCHANT/子商户, PLAYER/玩家)
effectiveRtpValue	integer	必填	当前生效RTP值
merchantRtpValue	integer	必填	商户RTP值
subMerchantRtpValue	integer	必填	子商户RTP值
playerRtpValue	integer	必填	玩家RTP值

5.3.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
```

```
"effectiveType": "PLAYER",
"effectiveRtpValue": 96,
"merchantRtpValue": 95,
"subMerchantRtpValue": 95,
"playerRtpValue": 96
}
}
```

5.3.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

5.3.8 使用场景

- 查看当前生效的RTP策略
- 确认RTP优先级应用情况
- 调试RTP配置问题
- 核实玩家实际游戏回报率设置

5.4 RTP变更记录

5.4.1 接口地址

text

GET /game/v5/rtp/change/history

5.4.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "currencyCode": "BR_BRL",
  "subMerchantCode": "SUB_MERCHANT_001",
  "playerId": "PLAYER_123456",
  "startTime": "2025-09-31T23:59:59.777Z",
}
```

```
"endTime": "2025-09-01T23:59:59.777Z",
"cursor": "1ABC",
"pageSize": 20
}
```

5.4.3 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	必填	货币编码, 如BR_BRL
subMerchantCode	string	可选	子商户编码, 如SUB_MERCHANT_001
playerId	string	可选	玩家ID, 如PLAYER_123456
playerIds	list[string]	可选	玩家ID集合, 如["A_PLAYER_1", "B_PLAYER_2"]
startTime	string	必填	查询开始时间, 如2024-12-01T00:00:00Z
endTime	string	必填	查询结束时间, 如2024-12-25T23:59:59Z
cursor	string	可选	游标, 如果为空则查询第一页, 否则查询游标之后的数据
pageSize	int	必填	默认每页20条数据, 最大100

5.4.4 注意事项

仅支持查询最近一个月的变更记录

5.4.5 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "nextPageCursor": "1ABC",
    "rtpChangeList": [
      {
        "merchantCode": "MERCHANT_001",
        "subMerchantCode": "SUB_MERCHANT_001",
        "playerId": "PLAYER_123456",
        "beforeRtp": 95,
        "currentRtp": 96,
        "changeTime": "2024-12-25 10:30:00 000"
      }
    ]
  }
}
```

5.4.6 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
nextPageCursor	string	选填	下页游标
rtpChangeList	array	必填	RTP变更记录列表

rtpChangeList 数组项说明

字段名	类型	必填	说明
merchantCode	string	必填	商户编码
subMerchantCode	string	必填	子商户编码
playerId	string	必填	玩家ID
beforeRtp	integer	必填	变更前RTP值
currentRtp	integer	必填	当前RTP值
changeTime	string	必填	变更时间

5.4.7 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "rtpChangeList": [
      {
        "merchantCode": "MERCHANT_001",
        "subMerchantCode": "SUB_MERCHANT_001",
        "playerId": "PLAYER_123456",
        "beforeRtp": 95,
        "currentRtp": 96,
        "changeTime": "2024-12-25T10:30:00Z"
      },
      {
        "merchantCode": "MERCHANT_001",
        "subMerchantCode": "SUB_MERCHANT_001",
        "playerId": "PLAYER_123456",
        "beforeRtp": 94,
        "currentRtp": 95,
        "changeTime": "2024-12-20T15:45:00Z"
      }
    ]
  }
}
```


5.4.8 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

5.4.9 使用场景

- 查询RTP变更历史记录
- 审计RTP调整操作

5.5批量修改用户RTP

5.5.1 接口地址

text

POST /game/v5/batch/set/player/rtp

5.5.2 请求参数

json

```
{
  "reqTraceId": "TRACE_XXX",
  "currencyCode": "BR_BRL",
  "rtp": 96,
  "playerIds": ["A_PLAYER_1", "B_PLAYER_2"]
}
```

5.5.3 参数说明

参数名	类型	可选/必填	说明
currencyCode	string	必填	货币编码，如 BR_BRL

rtp	integer	必填	RTP值, 如96
playerIds	list[string]	必填	最多支持50个, 玩家ID集合

5.5.4 注意事项

- 商户传递的玩家ID需要保证唯一, 平台会将相同玩家ID视作同一个玩家数据
- 批量设置玩家RTP功能采用异步处理机制, 系统接收请求后立即返回受理成功响应, RTP变更将在后台异步处理后生效, 可通过RTP变更记录接口确认最新生效RTP信息

5.5.5 响应结果

```
json
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {}
}
```

5.5.6 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应结果

5.5.7 成功响应示例

```
json
{
  "success": true,
```

```
"code": "C10000",  
"msg": "Request succeeded",  
"data": {}  
}
```

5.5.8 错误响应示例

```
json  
{  
  "success": false,  
  "code": "C40000",  
  "msg": "参数错误"  
}
```

5.5.9 使用场景

- 为特定玩家设置个性化RTP值
- 测试玩家游戏体验调整
- 根据玩家行为优化游戏回报率

6. 交易相关API

6.1 投注记录

6.1.1 接口地址

text

GET /game/v5/game/record

6.1.2 请求参数

json

```
{  
  "reqTraceId": "TRACE_XXX",  
  "pageNum": 1,  
  "pageSize": 20,  
  "reqData": {  
    "startTime": "2024-12-01T00:00:00Z",  
    "endTime": "2024-12-25T23:59:59Z"  
  }  
}
```

6.1.3 参数说明

参数名	类型	可选/必填	说明
pageNum	int	必填	页码
pageSize	int	必填	每页容量
startTime	string	必填	查询开始时间, 如 2024-12-01T00:00:00Z
endTime	string	必填	查询结束时间, 如 2024-12-25T23:59:59Z

6.1.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "gameRecordList": [
      {
        "orderNo": 123456789,
        "playerUniqueId": "PLAYER_123456",
        "betAmount": "100.00",
        "winAmount": "500.00",
        "betTime": "2024-12-25T10:30:00Z",
        "winTime": "2024-12-25T10:35:00Z",
        "gameCode": "FORTUNE_DRAGON",
        "currencyCode": "BR_BRL",
        "roundId": "ROUND_789012"
      }
    ]
  }
}
```

6.1.5 字段说明

响应根字段

字段名	类型	必填	说明
-----	----	----	----

success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
gameRecordList	array	必填	游戏记录列表

gameRecordList 数组项说明

字段名	类型	必填	说明
orderNo	Long	必填	订单号
playerUniqueld	string	必填	玩家ID
betAmount	string	必填	投注金额
winAmount	string	必填	派奖金额
betTime	string	必填	投注时间
winTime	string	必填	派奖时间
gameCode	string	必填	游戏code
currencyCode	string	必填	地区/货币唯一编码
roundId	string	必填	牌局ID

6.1.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "gameRecordList": [
      {
        "orderNo": 123456789,
        "playerUniqueId": "PLAYER_123456",
        "betAmount": "100.00",
        "winAmount": "500.00",
        "betTime": "2024-12-25T10:30:00Z",
        "winTime": "2024-12-25T10:35:00Z",
        "gameCode": "FORTUNE_DRAGON",
        "currencyCode": "BR_BRL",
        "roundId": "ROUND_789012"
      },
      {
        "orderNo": 123456790,
        "playerUniqueId": "PLAYER_123456",
        "betAmount": "50.00",
        "winAmount": "0.00",
        "betTime": "2024-12-25T11:00:00Z",
        "winTime": "2024-12-25T11:05:00Z",
        "gameCode": "FORTUNE_TIGER",
        "currencyCode": "BR_BRL",
        "roundId": "ROUND_789013"
      }
    ]
  }
}
```

6.1.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

6.1.8 使用场景

- 查询游戏投注记录

- 核对玩家游戏交易明细
- 进行游戏数据分析
- 处理玩家争议记录

7. 商户API

7.1 基础信息

7.1.1 接口地址

text
POST /game/v5/merchant/info

7.1.2 请求参数

json

```
{
  "reqTraceId": "xxxx"
}
```

7.1.3 参数说明

参数名	类型	可选/必填	说明
reqTraceId	string	可选	请求追踪id

7.1.4 响应结果

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantName": "测试商户",
    "merchantCode": "MERCHANT_001",
    "apiType": "TRANSFER_WALLET",
    "singleControlInGlobal": true,
  }
}
```

```
"callbackUrl": "https://callback.example.com",
"whitelips": "192.168.1.1,192.168.1.2",
"walletInfos": [
  {
    "billingMode": "QUANTITY",
    "rtp": 95,
    "currencyCode": "BR_BRL",
    "balance": "1000.00"
  }
]
}
```

7.1.5 字段说明

响应根字段

字段名	类型	必填	说明
success	boolean	必填	请求是否成功
code	string	必填	响应码, C10000表示成功
msg	string	必填	响应消息
data	object	必填	响应数据

data 对象字段

字段名	类型	必填	说明
merchantName	string	必填	商户名称
merchantCode	string	必填	商户code
apiType	string	必填	API类型 (SINGLE_WALLET/TRANSFER_WALLET)
singleControllnG lobal	boolean	必填	单控是否计入大盘

callbackUrl	string	必填	回调地址
whitelips	string	必填	白名单ip
walletInfos	array	必填	商户钱包地区信息

walletInfos 数组项说明

字段名	类型	必填	说明
billingMode	string	必填	计费模式 (MONTHLY/QUANTITY)
rtp	integer	必填	RTP值
currencyCode	string	必填	货币编码
balance	string	必填	余额

7.1.6 成功响应示例

json

```
{
  "success": true,
  "code": "C10000",
  "msg": "Request succeeded",
  "data": {
    "merchantName": "测试商户",
    "merchantCode": "MERCHANT_001",
    "apiType": "TRANSFER_WALLET",
    "singleControlInGlobal": true,
    "callbackUrl": "https://callback.example.com",
    "whitelips": "192.168.1.1,192.168.1.2",
    "walletInfos": [
      {
        "billingMode": "QUANTITY",
        "rtp": 95,
        "currencyCode": "BR_BRL",
        "balance": "1000.00"
      }
    ]
  }
}
```

7.1.7 错误响应示例

json

```
{
  "success": false,
  "code": "C40000",
  "msg": "参数错误",
  "data": null
}
```

7.1.8 使用场景

- 查询商户基本信息及各地区钱包配置
- 获取商户支持的货币类型及对应余额
- 查询商户API类型和相关配置信息

8. 参考

商户平台签名机制文档

签名生成规则

签名字段排序规则

按照字段序号对以下参数进行排序并拼接成加签字符串：

字段序号	字段名称	备注
1	merchantCode	商户编码，平台方获取
2	timestamp	当前时间戳，要求和请求头中保持一致
3	nonce	随机字符串，要求和请求头中保持一致

4	signType	签名算法, 要求和请求头中保持一致
5	body	实际传输数据(如果存在的话), 数据传输格式为json

加签字符串生成规则

1. 按照字段序号顺序拼接各字段值
2. 签名数据必须与实际发送数据完全一致(每个字符、每个字节)
3. 如果没有请求体, 则只拼接前四个字段

示例说明

示例参数

- 商户**Code**: 123123
- 时间戳: 1744897560043
- 随机字符串: 7c83a1c5f989498d8c73a2d54d66657f
- 签名算法: HmacSHA256
- 请求体:

```
{"merchantSession": "9259c473c8a94f68a7ec1e766d33b297", "currencyCode": "BR_BRL"}
```

拼接后的加签字符串

text
12312317448975600437c83a1c5f989498d8c73a2d54d66657fHmacSHA256{"merchantSession": "9259c473c8a94f68a7ec1e766d33b297", "currencyCode": "BR_BRL"}

签名结果

text
920fb61116e04a9e2c09264399960a91db04ff35cd10018fc1b6d87a25a10165

Java签名实现示例

```

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.util.HexFormat;

public class SignatureDemo {
    static String apiUrl =
"http://xxxxx.xxxxx.xxxxx/xxxxx/game/v3/game/launchUrl";
    static String merchantCode = "xxxxxxxxxxxxxx";
    static String secretKey = "xxxxxxxxxxxxxx";

    public static void main(String[] args) throws Exception {
        postJson("""
        {
            "gameCode": "126",
            "playerSession": "playerSession-U10000001",
            "playerUniqueld": "U10000001",
            "currencyCode": "US_USD",
            "language": "en",
            "rtp": 96,
            "balance": "9999.00",
            "subMerchantCode": "site",
            "terminalType": "PC",
            "returnUrl": "https://www.example.com/"
        }
        """);
    }

    public static void postJson(String jsonBody) throws Exception {
        String timestamp = String.valueOf(System.currentTimeMillis());
        String nonce = String.valueOf(Math.random());
        String contentTypeProcessingType = "HmacSHA256";

        String spliceResult = merchantCode + timestamp + nonce +
contentProcessingType;
        String sign = hmacSHA256Sign(secretKey, spliceResult + jsonBody);

        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(apiUrl))
            .header("Content-Type", "application/json")
            .header("X-MERCHANT-CODE", merchantCode)
            .header("X-TIMESTAMP", timestamp)
            .header("X-NONCE", nonce)
            .header("X-SIGN", sign)

```

```

        .header("X-CONTENT-PROCESSING-TYPE",
contentProcessingType)
        .POST(HttpRequest.BodyPublishers.ofString(jsonBody,
StandardCharsets.UTF_8))
        .build();

    HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
    System.out.println("Response Code: " + response.statusCode());
    System.out.println("Response Body: " + response.body());
}

    public static String hmacSHA256Sign(String secretKey, String data) throws
Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new
SecretKeySpec(secretKey.getBytes(StandardCharsets.UTF_8),
"HmacSHA256");
        mac.init(secretKeySpec);
        return hexString(mac.doFinal(data.getBytes(StandardCharsets.UTF_8)));
    }

    public static String hexString(byte[] data) {
        return HexFormat.of().formatHex(data);
    }
}

```

重要注意事项

1. 完整性要求: 所有参与签名的字段必须同时放入请求 `Header` 或 `Body` 中
2. 密钥安全: 商户密钥在开户时由 AG 提供, 请妥善存储
3. 数据一致性: 签名数据与实际发送数据必须完全一致
4. 参数一致性: 时间戳、随机字符串、签名算法在请求头和签名字符串中必须保持一致

Z2. 响应码

游戏服务相关错误

响应码	消息	说明
G10001	游戏服务异常	游戏服务出现内部异常

G10002	调用游戏服务异常	调用游戏服务时发生异常
--------	----------	-------------

业务参数相关错误

响应码	消息	说明
G20001	playerId为空	玩家ID参数不能为空
G20002	游戏CODE不存在:	指定的游戏代码不存在
G20003	游戏已下架:	请求的游戏已被下架
G20004	子商户已被禁用	操作的子商户处于禁用状态
G20005	钱包不存在	玩家或商户钱包不存在
G20006	玩家不存在:	指定的玩家不存在
G20007	子商户不存在:	指定的子商户不存在
G20008	查询余额异常:	查询账户余额时发生异常
G20009	RTP范围错误:, 范围应为: {1}-{2}	设置的RTP值超出允许范围
G20010	游戏维护中:	请求的游戏正在维护中
G20011	游戏分类不存在:	指定的游戏分类不存在
G20012	游戏分类已禁用:	请求的游戏分类已被禁用
G20013	商户信息查询失败:	查询商户信息时失败

账户与交易相关错误

响应码	消息	说明
G30001	当前游戏用户会话已过期	玩家会话已超时需要重新登录

G30002	商户余额不足	商户账户余额不足以完成操作
G30003	玩家余额不足	玩家账户余额不足以完成操作
G30004	游戏启动参数不存在:	指定的游戏启动参数不存在
G30005	包月调用总次数不足	商户包月服务调用次数已用完
G30006	当前商户不支持此API模式	商户配置不支持当前请求的API模式
G30007	玩家账户不存在:	指定的玩家账户不存在
G30008	玩家账户余额查询异常:	查询玩家账户余额时发生异常
G30009	玩家账户余额不足:	玩家账户余额不足以完成当前操作
G300010	玩家提现失败:	玩家提现操作失败
G300011	玩家开户失败:	为玩家创建账户时失败
G300012	玩家充值失败:	玩家充值操作失败
G300013	玩家账户投注失败:	玩家投注操作失败
G300015	玩家全部提现失败:	玩家全部提现操作失败
G300016	商户订单已存在:	商户提交的订单号已存在
G300017	没有找到RTP信息	系统中未找到相关的RTP配置信息