PP 充值平台联运游戏服务端对接文档

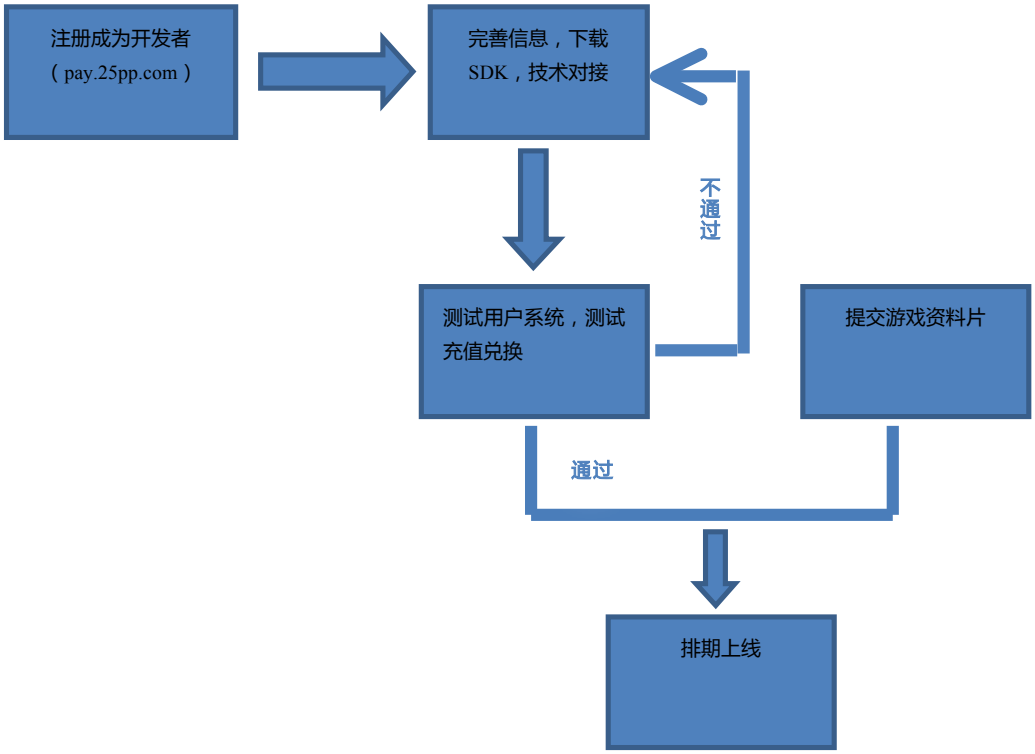
2015-07-07

目录

1. 服务端对接概述

服务端的对接分为两部分，登录验证和充值购买验证。登录验证通过后才能进入游戏，充值购买验证是指玩家在游戏中购买道具时调用 PP 的充值购买接口，PP 处理订单成功后通知游戏商，游戏商接收通知并验证数据一致后处理游戏订单的过程。

1.1 游戏上架流程



2. 登录验证

登录验证接口协议已调整，在游戏是混服和使用新接口的前提下，CP 在九游发行 Android 版本，在 PP 助手发行 iOS 版本，两个版本的游戏角色是互通的。2014-08-28 之后接入的游戏直接走新接口，存量游戏若升级新接口则有几点需要注意：

1、CP 是否以 PP 旧接口返回的 userid 关联游戏角色？

新接口未返回 username，CP 必须以 userid 做游戏角色关联，否则升级后会找不到角色。

2、CP 登录验证是二进制还是字符串方式？

服务端只能存在一套验证逻辑，不能新旧接口同时使用，所以若存在二进制验证包，则游戏需要做强更。

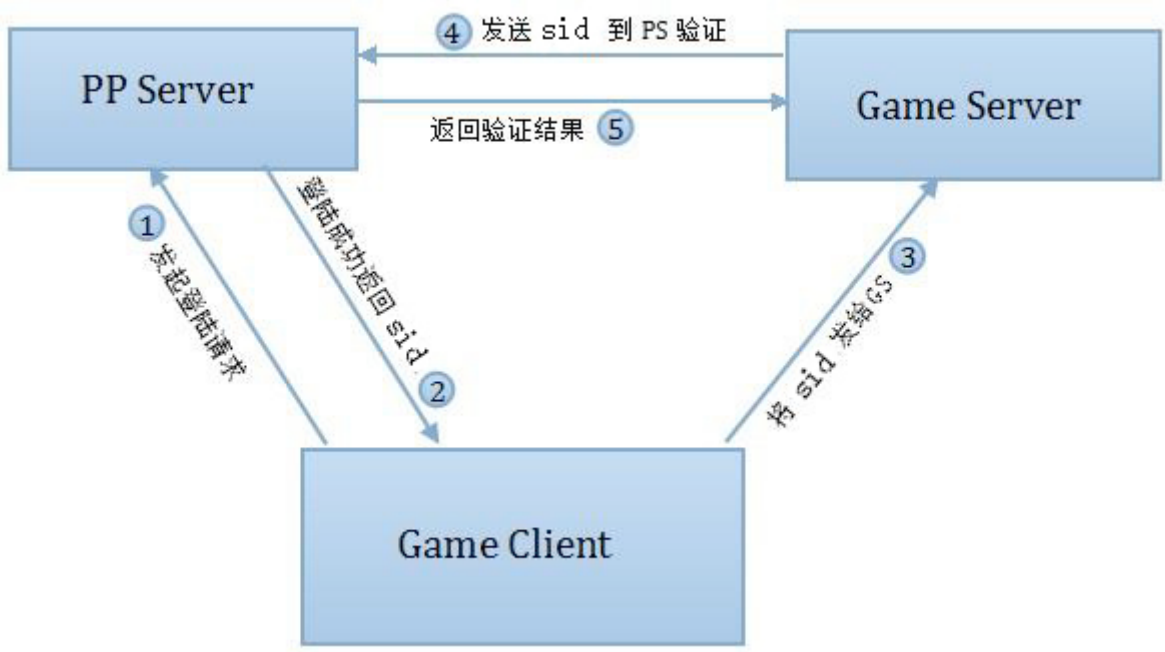
3、支付回调时，CP 是根据自己的订单来处理发货的吗，有没有依赖 PP 的 user_id、username？

游戏角色关联由 userid 变成了 creator+accountId ,如果发货有依赖 PP 返回的 user_id 或 username 则发货会存在找不到角色的情况。游戏需以自身订单号来处理发货逻辑。

4、若存量游戏升级新接口则必须通知 PP 运营人员，更多问题请到对接讨论组中提出。

为了实现两个平台的游戏角色互通，需要标记存量游戏使用新接口还是旧接口（存量游戏默认使用旧接口），以便做不同的逻辑处理。如果升级为新接口而未标记，则可能出现游戏角色丢失的情况。

2.1 登陆流程图



2.2 登陆验证流程说明

第一步：PPSDK 向 PP 服务端发起登陆请求。

第二步：PP 服务端验证用户名密码后返回 sid 给 PPSDK，PPSDK 将 sid 传给游戏客户端。（sid 每次返回都不一样，有效期 30 秒，验证一次后失效。）

第三步：游戏客户端向游戏服务端发送 sid。

第四步：游戏服务端把第三步接收到的 sid 发送给 PP 服务端验证。

第五步：PP 服务端返回 sid 的验证结果。

2.3 登陆验证方法说明

1) **请求地址：**
http://passport_i.25pp.com:8080/account?tunnel-command=2852126760
2) **调用方式：**HTTP POST
3) **接口描述：**

游戏客户端通过 PPSDK 获取到 sid，传到游戏服务端，游戏服务端到 PP 服务端验证用户会话 sid 的有效性，获取用户的 accountId、creator 和 nickName，此时游戏服务端不在以接入平台来关联游戏角色，而是用 creator 来关联游戏角色。

若 creator 为 PP，按接入 PP 助手来处理游戏角色；

若 creator 为 JY，按接入九游来处理游戏角色。

请一定注意，新的验证接口协议已调整，使用 creator+accountId 做唯一标识！

4) **请求方：**游戏服务端
5) **响应方：**PP 服务端
6) **请求内容（json 格式）：**

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	int	Y	Unix 时间戳或其他自定义内容， 不要超过 10 位例：1330395827
service	接口服务名称	string	Y	account.verifySession
data	请求数据	json	Y	json 格式
game	game 参数	json	Y	json 格式，gameId 值为 PP 的 AppId。格式如下：{gameId：76}
encrypt	加密方法	string	N	请求数据加密方法，目前只支持"MD5"。如果发起方传此参数，值必须是"MD5"；如果发起方不传此参数，接收方将其值默认为"MD5"。
sign	签名参数	string	Y	MD5(签名内容+AppKey)， 签名后的字符串统一转小写。签名内容示例：sid=ghkrpofaisdpfqkmweklvmnqwkeg23sd
请求数据(对应 data,采用 json 格式)				
sid	当前用户会话标识	string	Y	游戏客户端登录后从 SDK 取得，sid 必须是 32 字节的字符串才响应。

例子：

```
HTTP 请求的 body 内容：
{
    "id": 1330395827,
    "service": "account.verifySession",
    "data": {
        "sid": "80a5fe53d3540300005a17e308a4b1fb"
    },
    "game": {
        "gameId": 12345
    },
    "encrypt": "md5",
    "sign": "6e9c3c1e7d99293dfc0c81442f9a9984"
}
sign 的签名规则：MD5(sid=... + AppKey)（去掉 + 替换 ... 为实际 sid 值）
```

7） 返回内容（json 格式）：

字段名称	字段说明	类型	必填	备注
id	请求的唯一标识	int	Y	Unix 时间戳或其他自定义内容，与请求内容中 id 值相同。例：1330395827
state	响应状态	json	Y	
data	响应数据	json	Y	
响应状态（对应 state,采用 json 格式）				
code	响应码	int	Y	
msg	结果描述	string	Y	
响应数据(对应 data,采用 json 格式)				
accountId	帐号标识	string	Y	最长为 32 个字符
creator	角色创建者	string	Y	JY：九游。PP：PP 助手
nickName	用户昵称	string	Y	

例子：

```
{
    "id": 1330395827,
    "state": {
        "code": 1,
        "msg": "OK"
    },
    "data": {
        "accountId": "U11626774a4e39c16cf7mmsnz5002une",
        "creator": "PP",
        "nickName": "口水哥"
    }
}
```

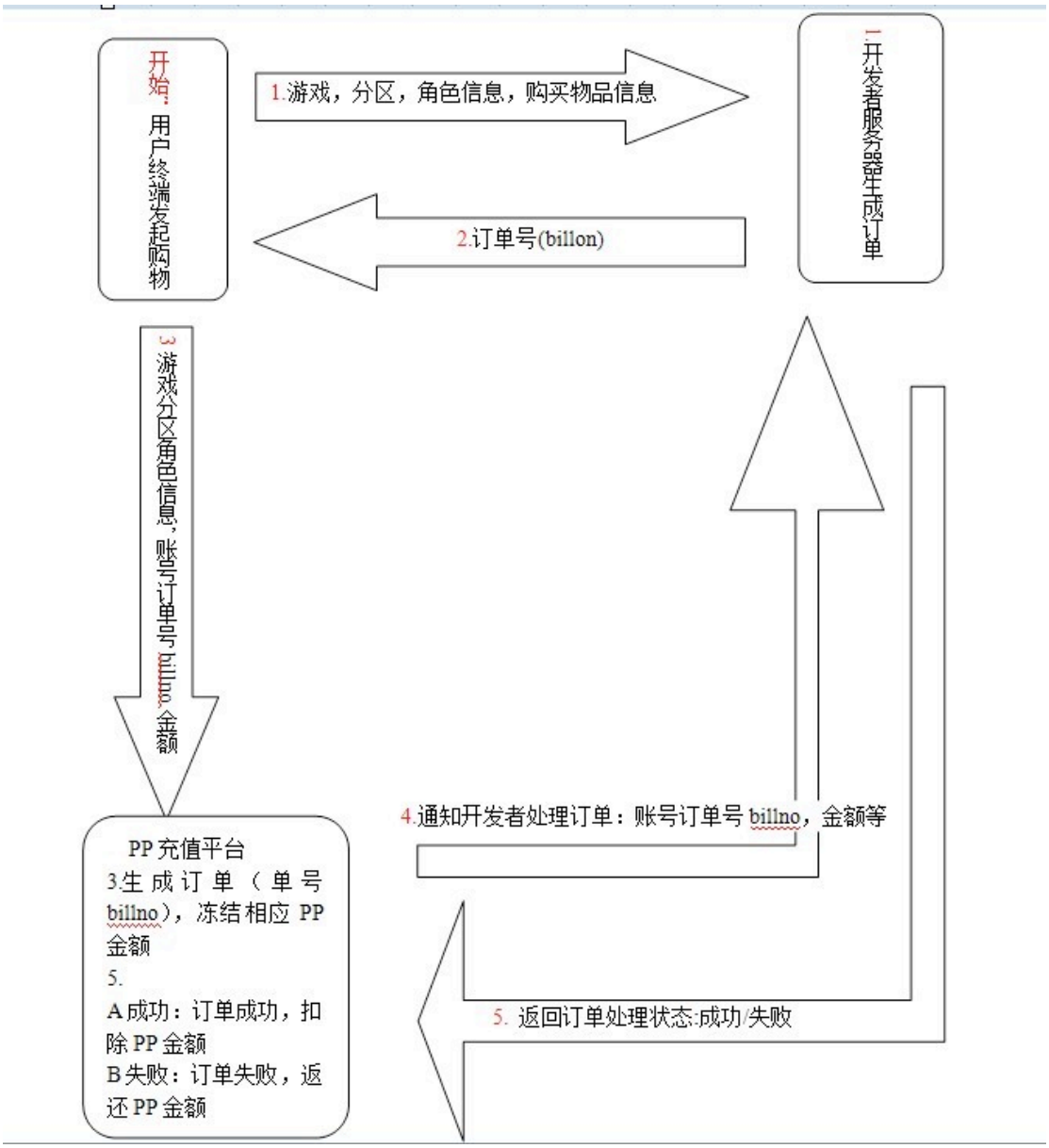
响应码说明（state.code）:

响应码	说明
1	成功
10	请求参数错误
11	用户未登录
99	网络繁忙，请稍后重试
175	接口调用受限（如返回该响应码，请联系 pp 技术处理）

3. 充值购买验证

参考例子：./服务端 DEMO/充值购买验证/

3.1 充值购买流程图



第一步：玩家发起购物（客户端中完成）

第二步：游戏服响应生成订单并返回订单号。

第三步：客户端调用 PP 服接口（客户端中完成）

第四步：PP 服生成订单并发通知给游戏服。

第五步：游戏服验证通知并处理订单，成功返回 success 给 PP 服。

3.2 充值购买方法说明

流程图第三步有两种方式(在客户端完成，服务端不用处理)

1 游戏内购买，PP 币兑换

请查看SDK客户端对接文档中的exchangeGoods 接口

以下是必需参数

billno	varchar(30)	厂商订单号(最多不超过 30 位)
amount	double(8,2)	兑换 PP 币数量=整数
zone	int(10)	厂商应用分区 id(值为 0 忽略)
roleid	varchar(30)	厂商应用角色 id(值为 0 忽略，最多不超过 30 位)
billno_title	varchar(30)	订单标题

流程图第四步为 PP 服通知游戏服

数据通过 POST 方式发送到通知地址，下面的表显示了游戏服接收到的参数.

字段名	长度	描述	是否必须
order_id	bigint(20)	兑换订单号	是
billno	varchar(30)	厂商订单号（原样返回给游戏服）	是
account	varchar(50)	通行证账号	是
amount	double(8,2)	兑换 PP 币数量	是
status	tinyint(1)	状态: 0 正常状态 1 已兑换过并成功返回	是
app_id	int(10)	厂商应用 ID（原样返回给游戏服）	是
uuid	char(40)	设备号（返回的 uuid 为空）	是
roleid	varchar(30)	厂商应用角色 id（原样返回给游戏服）	是
zone	int(10)	厂商应用分区 id（原样返回给游戏服）	是
sign	text	签名(RSA 私钥加密)	是

流程图第五步游戏服验证通知并处理订单

1、把接收到的原始数据 urldecode 处理。（php 的$_POST 接收可以忽略这步）

2、使用 base64_decode 对上一步处理过的 sign 进行转换

3、对接收到的数据 sign 使用开发者中心的公钥进行解码（rsa 解码）

4、sign 解析出来的数据与订单中的数据一致，即视为验证通过

5、status 做重复判断，当等于 0 时，表示订单需要进行兑换处理

6、回调事件结果处理完成后，只有处理成功必须返回：success，其它状态返回：fail

说明 1：处理接收到 PP 服的通知数据

通知数据示例

order_id=2012110900000364&billno=8888888888888&account=pp123456&amount=10&status=0&app_id=93&sign=0G4%2F2zwBkuc%2FJDY8Rkn8iyUMxNVWa7ayBG%2FiCQVZCqjBzyD%2F07Z9RI93sLTjJN7XSBVhGD%2FUylE%2FfcTMr5EbQJZDC%2FZ7JNDbJidCHUdUnRkt1YFuDk7tVMlv%2Br2CvtAJv9eLM5tbtrkZwImJt4qZcubbYkEEKVSDv4GXtx2aLo8%3D

1、游戏服接收到的 sign 是 server 端通过私钥对加密后进行 base64_encode 编码生成的字符串

2、解密代码片段

使用 base64_decode 后 ,使用公钥对 sign 进行解密(不同语言的 RSA 解密请参考./服务端 DEMO/充值购买验证)，得到为 json 格式数据

```php
$privatebackdata = base64_decode($_POST['sign']);

$MyRsa = new MyRsa;
$data = $MyRsa->publickey_decodeing($privatebackdata, MyRsa::public_key);
$rs = json_decode($data,true);
```

$data[‘order_id’] =2012110900000364
$data[‘billno’] =888888888888
$data[‘account] =pp123456
$data[‘amount’] =10
$data[‘status’] =0
$data[‘app_id’] =93

解析出来的数据与游戏订单中的数据一致，即视为验证通过。(参考例子：./服务端 DEMO/充值购买验证)

【注意】:

由于存在多次发送通知的情况，因此“游戏服务器”必须能够正确处理重复的通知。当接收到通知时，需要检查系统内对应业务数据的状态，以判断该通知是否已经处理过。在对业务数据进行状态检查或处理之前，需要采取数据加锁或时间戳判断等方式进行并发控制。

说明 2：返回与队列通知

请游戏开发商接到通知后返回 success，其他任何字符（包括 fail）或者空数据视为失败，失败记录将会进入重发队列。

失败通知延时后会在 10 秒 30 秒 1 分钟 5 分钟 10 分钟 半小时 1 小时 2 小时 4 小时 6 小时 8 小时 10 小时 12 小时 24 小时各通知一次。

说明 3：关于测试状态、正式状态、测试币发放如下：

1.游戏在开发者后台分为“测试状态”与“正式状态”，每个新游戏默认一开始为测试状态，游戏状态修改请联系我司运营同事。

2.测试状态：只能显示和使用测试币，不能使用 RMB，只能测试兑换，不可测试充值和充值兑换，测试状态下充值兑换不会发通知到游戏服务器

正式状态：只能显示和使用 RMB，不能使用测试币，可测试全部兑换流程；我司只发放测试币，不会发放 PP 币，如需测试充值，请将游戏转成正式状态，自行充值 PP 币测试

3.可用普通 pp 账号进行所有测试（自己申请，绝对不能用开发者账号测试），账号名必须带有 中文、英文、数字（如“测试账号 123abc”，以测试中文有效性），请自行申请，如需测试币请把账号发到讨论组。

充值购买验证 PHP 例子代码片段：

```
<?php
include('Rsa.php');
include('MyRsa.php');
//注意：接收到的是原始数据要 urldecode 解码。
//如：$data = file_get_contents("php://input","r");
$privatebackdata = base64_decode($_POST['sign']);
$MyRsa = new MyRsa;
        $data = $MyRsa->publickey_decodeing($privatebackdata, MyRsa::public_key);
        $rs = json_decode($data,true);
        if(empty($rs)||empty($_POST)) return false;
        if($rs["billno"] == $_POST['billno'] && $rs["amount"] == $_POST['amount']) {
            //验证通过
            //--------业务处理----------
            echo "success";
        }else{
            echo "fail";
        }
?>
```