

魅族游戏平台接入手册

V 1.2.0

版本	日期	说明
1.0	2014/5/13	初稿完成
1.1	2014/11/24	1. 新增 SDK2.0 的 Gamebar 功能 2. 优化支付流程 3. 优化 SDK 接口 4. 修复 BUG，优化用户体验
1.2	2015/1/9	1. 新增快捷注册功能 2. 新增话费支付 3. 修复 BUG 及兼容性问题

目 录

一、魅族游戏平台	3
1.1. 平台介绍	3
1.2. 魅族联网手游 SDK 简介	3
1.3. Flyme 账户介绍	3
二、联网手游合作流程概述	4
2.1. 合作流程	4
2.2. 商务联系方式	4
三、 联网手游接入流程概述	5
3.1. 流程概述	5
3.2. 获取游戏参数	5
3.3. 接入须知	7
四、客户端接入说明	7
4.1. 流程概述	7
4.1.1. 登陆流程	8
4.1.2. 支付流程	9
4.2. SDK 接入方法	10
4.2.1. 开发环境配置	10
4.2.2. SDK 初始化	10
4.2.3. 登录调用代码	10
4.2.4. 支付调用代码	11
4.2.5. 游戏退出调用代码	13
4.3 悬浮窗口 Gamebar	13
4.3.1 Gamebar 功能	13
4.3.2 接入过程	13
4.3.3 主动显示和隐藏	14
五、服务端接入说明	15
5.1. 接入概述	15
5.1.1. SESSION 校验流程	15
5.1.2. 购买流程说明	16
5.1.3. 创建 CP 服务端订单	16
5.2. 协议说明	20
5.2.1. 通信协议	20
5.2.2. 数据协议	20
5.3. 接口说明	21
5.3.1. SESSION 校验	21
5.3.2. 查询订单接口	22
5.3.3. 通知游戏方发货	23
六、发布联网手游	24
6.1. 完善游戏信息	24
6.2. 游戏数据统计查询	24
七、 常见问题	25

一、魅族游戏平台

1.1. 平台介绍

魅族 FlymeOS，作为最早的 Android 深度定制手机操作系统之一，以用户体验为核心，追求极致的魅力，在国内拥有相当优秀的口碑，拥有非常庞大的用户群及高忠诚度的粉丝。

旗下 App 分发渠道，有**魅族应用中心**、**魅族游戏中心**、**魅族官网 Flyme**、**应用中心 web 端**，目前（2014 年 10 月）拥有 1500 万活跃用户，日活跃用户 550 万，日下载量最高达 780 万次以上。

据调研样本，魅族 FlymeOS 用户类型较为丰富，具有魅族产品相同气质的用户较多，对应用、游戏的质量有较高的追求，对高质量 App 有非常高的粘度，同时用户均具有较高的付费习惯和 ARPU 值。

1.2. 魅族联网手游 SDK 简介

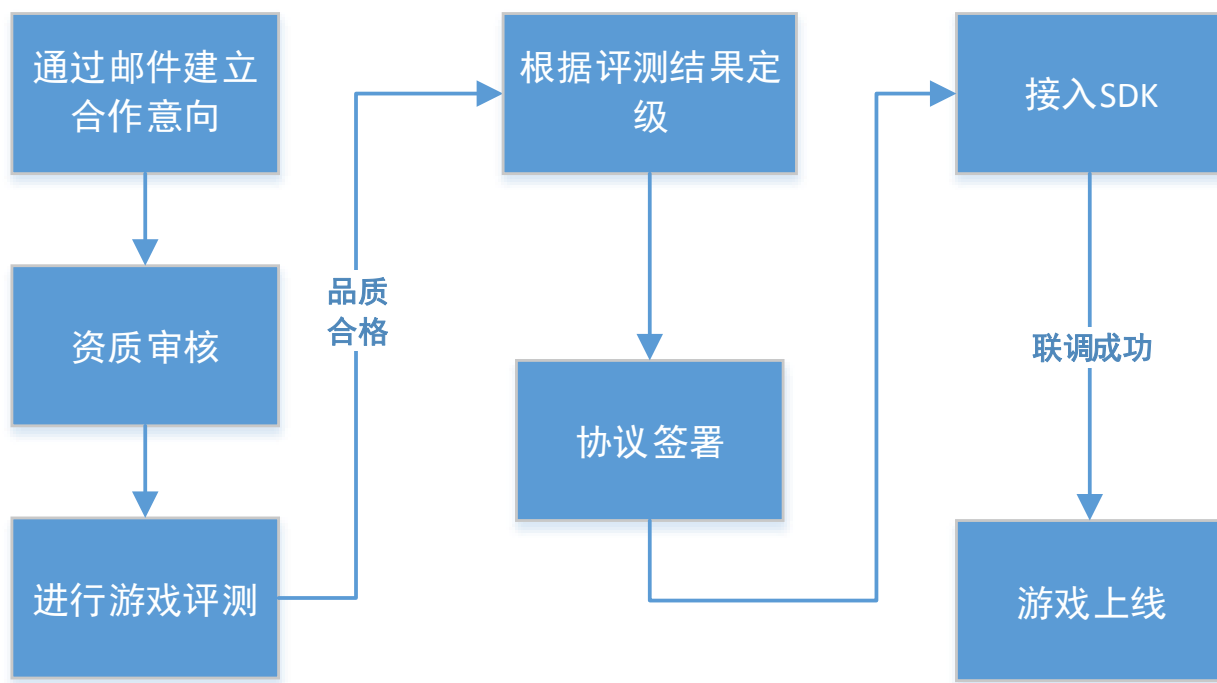
魅族联网手游 SDK 是为联网手游打造的高质量运营开发包，为游戏提供完善的 Flyme 账户体系，含登录、注册及充值支付等基础功能，同时为联运和独代的游戏提供了强大的活动公告、礼包中心、游戏攻略及游戏论坛等运营工具；后续将不断完善，为玩家及游戏厂商提供更多的贴心服务。

1.3. Flyme 账户介绍

Flyme 账号为魅族所有产品的通行证，打通魅族论坛以及 FlymeOS 的用户中心、应用中心、游戏中心、充值中心、音乐、视频、图库、联系人及云服务等账户，完善的账户管理体系，安全、便捷的充值支付，可以为联网手游提供全面的保障。

二、联网手游合作流程概述

2.1. 合作流程



2.2. 商务联系方式

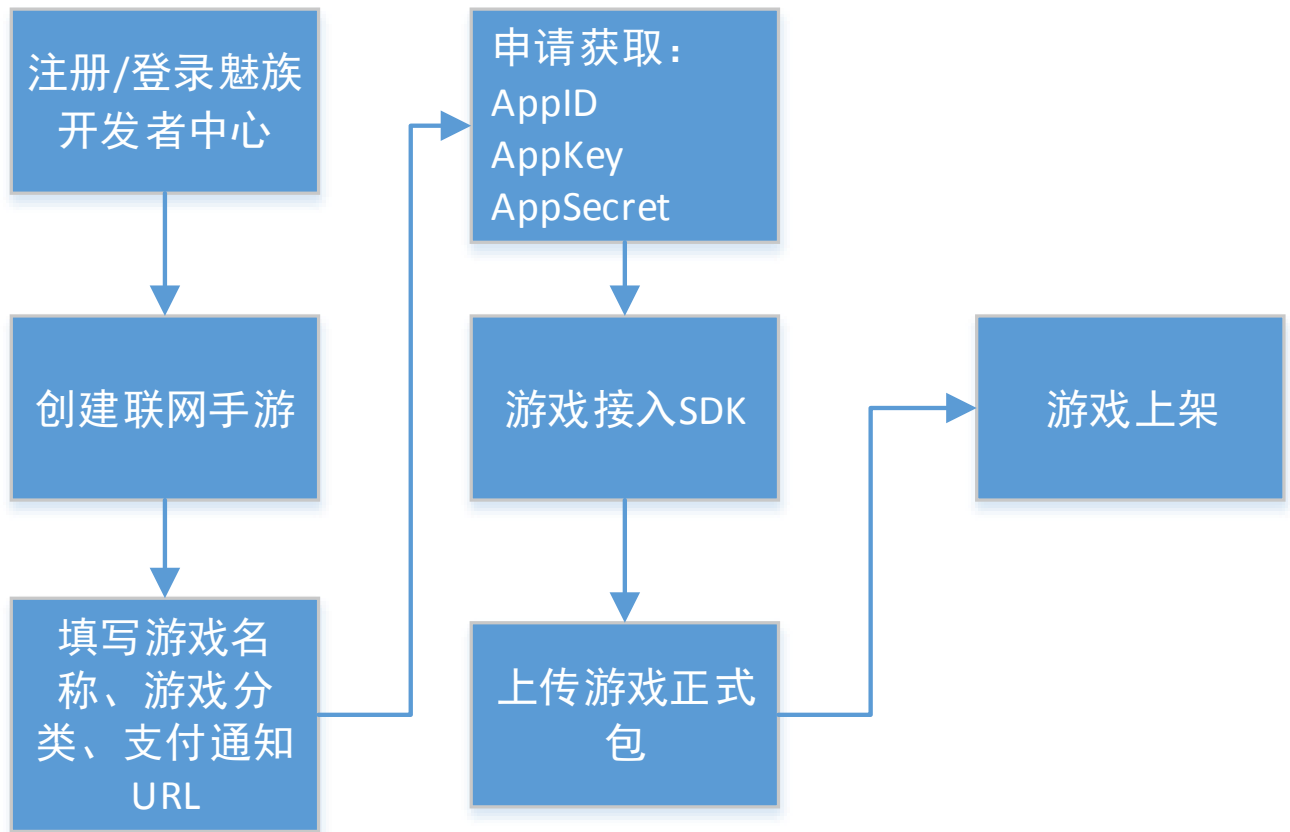
请发送您的联系方式、合作简述至以下任一邮箱地址：

37@meizu.com

37game@meizu.com

三、联网手游接入流程概述

3.1. 流程概述



参数说明:

- AppID: App 的唯一 ID
- AppKey: App 的唯一标识
- AppSecret: App 的密钥，魅族开发者平台分配给开发者，接口调用时用作签名
- 支付回调 URL: 玩家在游戏支付后得到支付结果，魅族游戏中心反馈到游戏服务器中的 URL

3.2. 获取游戏参数

3.2.1 创建联网手游

3.2.1.1 说明

目前魅族开发者平台仅为接入魅族联网手游 SDK 的游戏开放上传包之前申请获取

AppID、AppKey、AppSecret 的渠道，若确定申请接入 SDK，请在发布应用入口点击【创建联网手游】，非接入魅族手游 SDK 的同个此渠道申请，将不予审批通过。

3.2.1.2 操作

- ① 登录“魅族开发者”中心
- ② 进入管理页面，点击【发布应用】



- ③ 在【发布应用】点击【创建联网手游】



3.2.2 填写游戏基础资料

3.2.2.1 说明

申请创建接入魅族手游 SDK 的联网手游，只需要填写基本游戏的基本信息和支付回调 URL，即可创建游戏 App 信息记录，获取魅族开发者平台分配的游戏参数

3.2.2.2 操作

- ① 在创建联网手游页面，输入游戏名称（必填项）
- ② 选择游戏类型（必填项），填写支付通知 URL（必填项）
- ③ 点击【确定】执行创建

④ 此时已获得开发者平台分配的 AppID、AppKey 和 AppSecret


联网手游接入所需参数

游戏名称	卡通农场
AppID	2882303761517185394
AppKey	5331718587394
AppSecret	YUFullcyQL68QgOOiVp0bg==
支付通知URL	http://bbs.mysite.com/qqllogin.php?a=1

[点此下载《MEIZU应用中心联网手游开发SDK》](#)

[返回游戏列表](#)

⑤ 返回游戏列表后,将出现一条状态为“创建中”的记录;此时可以通过点击游戏名称,回到参数页面查看

应用名称	版本	价格	状态	评分	操作	
 卡通农场			创建中		添加版本	删除

3.3. 接入须知

所有联运手游接入时的 package 名称必须以“.mz”结尾。

游戏提交到开发者平台,由开发者平台以魅族私钥重新签名,在魅族开发者平台中测试并发布。

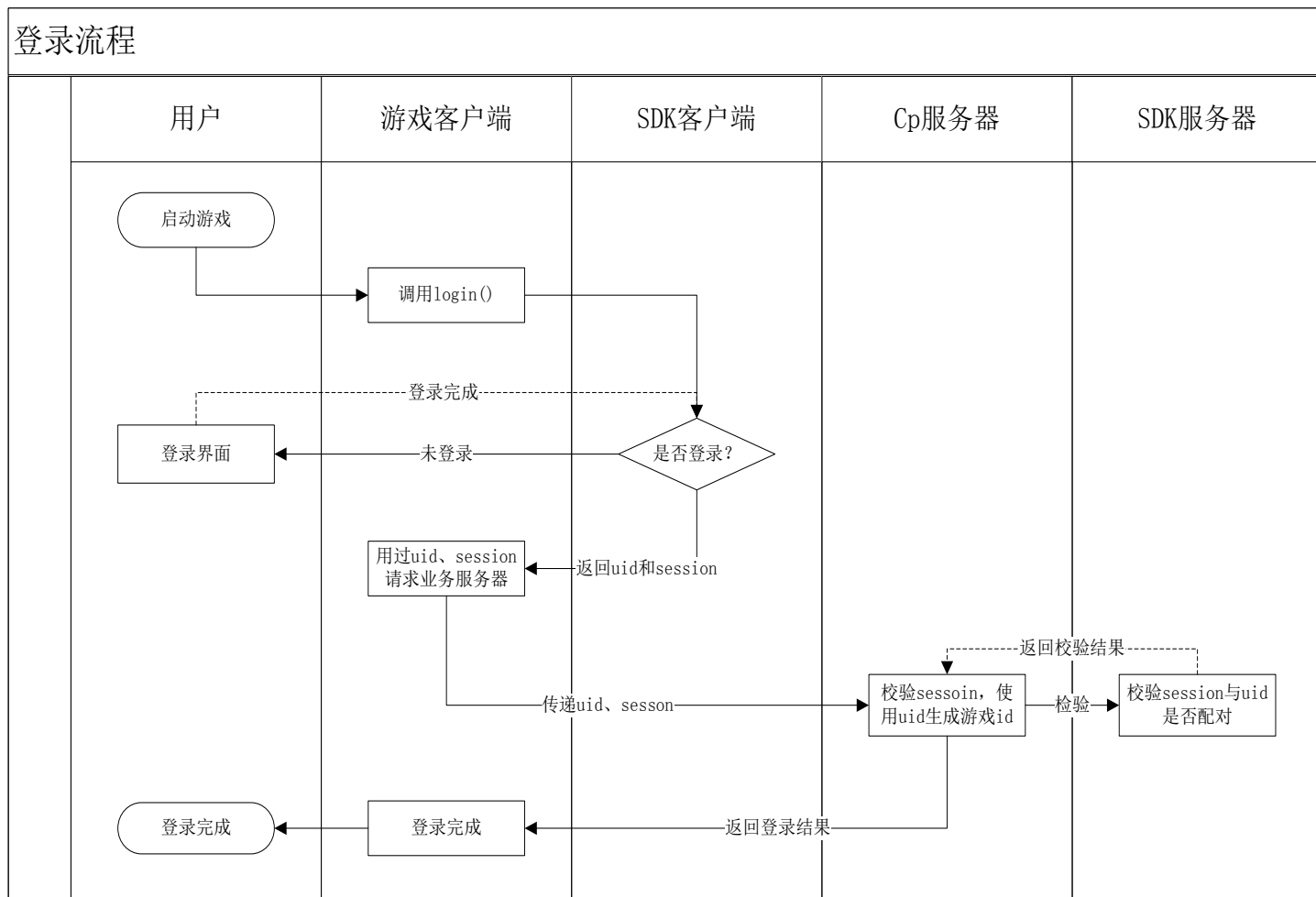
四、客户端接入说明

4.1. 流程概述

魅族游戏 SDK 主要提供两方面核心功能:登陆和支付。游戏每次启动,都需要先调用 login()接口来进行游戏登陆,通过魅族账户的 uid 来创建或者查询游戏的 id 等信息。登陆成功后,根据用户需要,可以调用 payOnline()接口来进行物品购买、充值。

用户登录完成进入游戏后，还能展示 Gamebar（悬浮的操作栏），用户可在游戏中方便地查询、修改账户信息，逛论坛、查攻略等。（注：如果游戏之前已经接入 SDK1.0 版本，即已经实现了登录及支付流程，可以直接查看《4.3 悬浮窗口 Gamebar》一节即可，其他内容与前一版本一致）

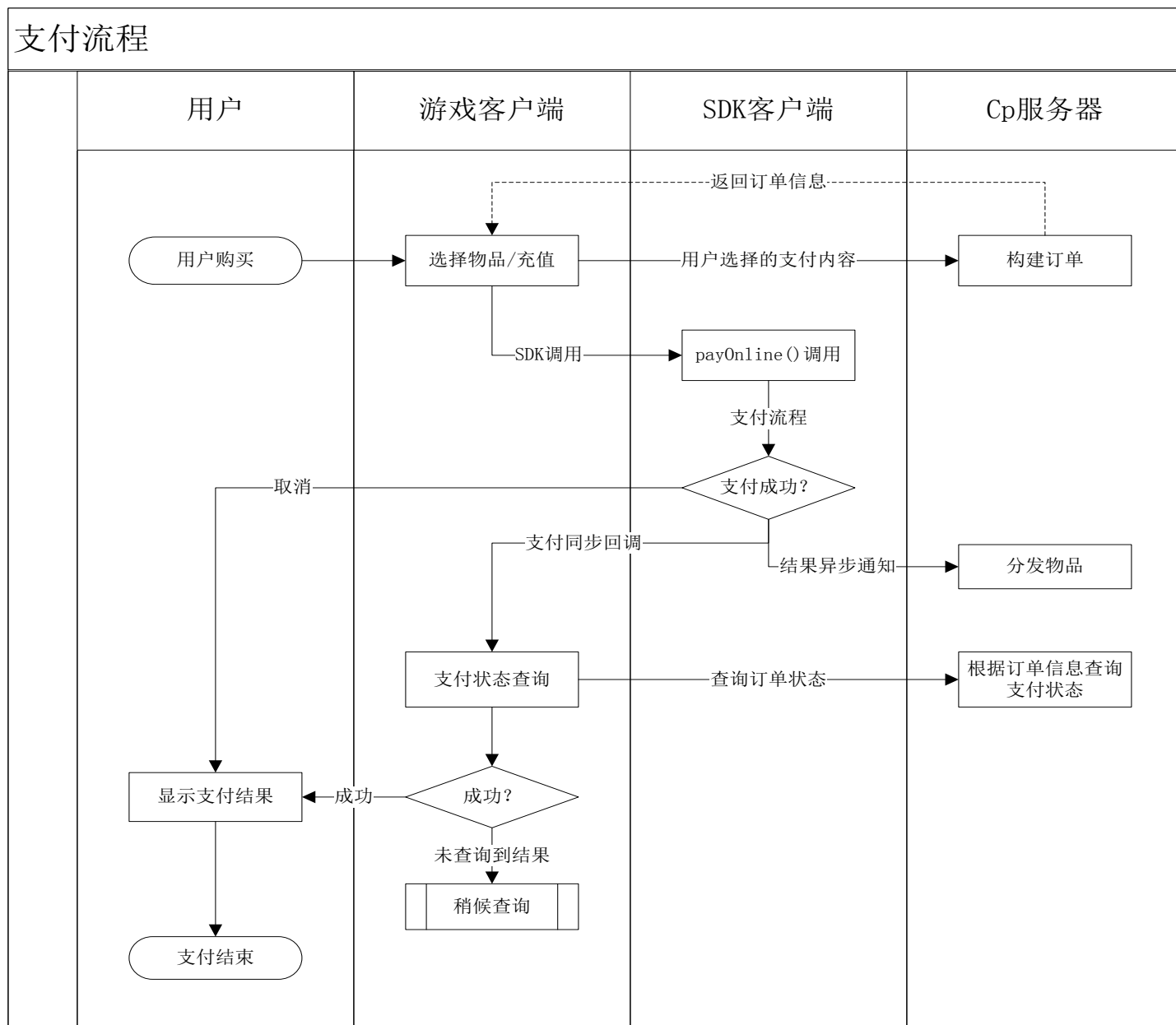
4.1.1. 登陆流程



注意：

- 1、uid 不是 Flyme 账户名，但是跟 Flyme 账户名是一一对应的关系。
- 2、开发者必须使用 uid 作为用户标识。
- 3、返回的 session 与 uid 存在对应关系，需要调用 SDK 服务器接口校验对应关系是否正确。

4.1.2. 支付流程



注意：

- 1、调用 `payOnline()` 接口的参数必须由 CP 服务端返回，因为需要对参数做签名。
- 2、支付结果通知 CP 服务端的回调通知是异步的，在极端情况下，可能存在延时问题。
- 3、游戏客户端收到 SDK 客户端的支付成功回调之后，需到 CP 服务端确认支付结果，以 CP 服务端结果为准。

4.2. SDK 接入方法

4.2.1. 开发环境配置

4.2.1.1. 加入 SDK 的 jar 文件

将提供的 MzGameCenterLib_xxx.jar 放到游戏工程的 libs 目录。

4.2.1.2. 配置应用权限

魅族游戏 SDK 需要使用到以下权限，请在工程的 AndroidManifest.xml 中加入权限声明：

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

4.2.1.3. 引入魅族游戏组件安装包到工程

将提供的 MzGameCenterService_xxx.apk 放到游戏工程的 assets 目录中。

4.2.2. SDK 初始化

使用 SDK 前，需要对 SDK 进行初始化。初始化需要魅族开发者平台申请的 appid 和 appkey 值作为参数。可以在 Application 的 onCreate()方法中调用。

```
MzGameCenterPlatform.init(context, appid, appkey);
```

注意：

init()方法在游戏应用启动时调用一次即可(推荐在 Application 的 onCreate()中调用)，该方法会记录游戏的 appid 和 appkey，没有其它特殊操作，不会阻塞线程。

4.2.3. 登录调用代码

```
// TODO 调用登录接口。注意，该方法必须在应用的主线程中调用。
MzGameCenterPlatform.login(MainActivity.this, new MzLoginListener() {
    @Override
    public void onLoginResult(int code, MzAccountInfo accountInfo, String errorMsg) {
        // TODO 登录结果回调。注意，该回调跑在应用主线程，不能在这里做耗时操作
```

```

switch(code){
    case LoginResultCode.LOGIN_SUCCESS:
        // TODO 登录成功，拿到uid 和 session到自己的服务器去校验session合法性
        mUid = accountInfo.getUid();
        displayMsg("登录成功！\r\n 用户名：" + accountInfo.getName() + "\r\n Uid：" +
accountInfo.getUid() + "\r\n session：" + accountInfo.getSession());

        break;

    case LoginResultCode.LOGIN_ERROR_CANCEL:
        // TODO 用户取消登陆操作

        break;

    default:
        // TODO 登陆失败，包含错误码和错误消息。
        // TODO 注意，错误消息(errorMsg)需要由游戏展示给用户，提示失败原因
        displayMsg("登录失败：" + errorMsg + ", code=" + code);

        break;
}
}
});

```

4.2.4. 支付调用代码

// TODO 以下信息需要客户端通过网络访问自己的服务器获取，客户端不需要也不能在本地生成。

// TODO 此处各字段值对应CP服务端生成订单的信息(后面的绿色字对应服务端的字段)，详情请参考

5.1.3 创建服务端订单。其中cp_order_id、uid、sign、sign_type、uid不能为空

```

String orderId = "xxx"; // cp_order_id (不能为空)

String sign = "xxx"; // sign (不能为空)

String signType = "xxx"; // sign_type (不能为空)

int buyCount = 1; // buy_amount
String cpUserInfo = "xxx"; // user_info
String amount = "10"; // total_price
String productId = "xxx"; // product_id
String productSubject = "xxx"; // product_subject
String productBody = "xxx"; // product_body

```

```

String productUnit = "xxx"; // product_unit

String appid = "xxx"; // app_id (不能为空)

String uid = "xxx"; // uid (不能为空)

String perPrice = "10"; // product_per_price
long createTime = 1385452122; // create_time
int payType = *; // pay_type

MzBuyInfo buyInfo = new MzBuyInfo().setBuyCount(buyCount).setCpUserInfo(cpUserInfo)
    .setOrderAmount(amount).setOrderId(orderId).setPerPrice(perPrice)
    .setProductBody(productBody).setProductId(productId).setProductSubject(productSubject)
    .setProductUnit(productUnit).setSign(sign).setSignType(signType).setCreateTime(createTi
me)

    .setAppid(appid).setUserUid(uid).setPayType(payType);

// TODO 调用支付接口。注意，该方法必须在应用的主线程中调用。
MzGameCenterPlatform.payOnline(MainActivity.this, buyInfo, new MzPayListener() {
    @Override
    public void onPayResult(int code, MzBuyInfo info, String errorMsg) {

        // TODO 支付结果回调，该回调跑在应用主线程。注意，该回调跑在应用主线程，不能在这里
        做耗时操作

        switch(code){
            case PayResultCode.PAY_SUCCESS:

                // TODO 如果成功，接下去需要到自己的服务器查询订单结果

                displayMsg("支付成功：" + info.getOrderId());

                break;
            case PayResultCode.PAY_ERROR_CANCEL:

                // TODO 用户主动取消支付操作，不需要提示用户失败

                break;
            default:

                // TODO 支付失败，包含错误码和错误消息。

                // TODO 注意，错误消息(errorMsg)需要由游戏展示给用户，提示失败原因

                displayMsg("支付失败：" + errorMsg + ", code = " + code);

                break;
        }
    }
});

```

```
}  
}  
});
```

4.2.5. 游戏退出调用代码

游戏退出时，在适当的地方调用退出接口。

```
MzGameCenterPlatform.logout(context);
```

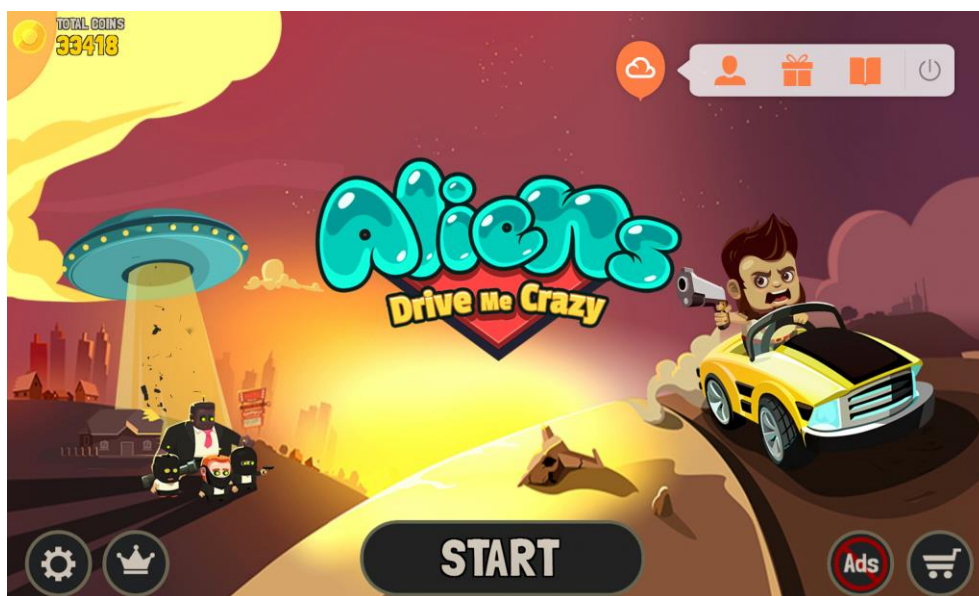
注意：

- 1.logout()方法只需要在游戏结束（用户退出游戏）时调用即可，用户注销账户时不需要调用。
- 2.如果用户选择注销账户，游戏只需要清除自身记录的用户信息即可。
- 3.如需要重新登录、切换账户等操作，游戏只需先清除本身记录的账户信息，然后调用 login()接口，登录流程中用户即可做相关的切换操作。

4.3 悬浮窗口 Gamebar

4.3.1 Gamebar 功能

SDK2.0 在 1.0 的基础上增加了悬浮窗的功能，在游戏登录成功后会出现在游戏的界面中，点击悬浮窗上的功能可打开账号，礼包和攻略等功能，如下图所示：



4.3.2 接入过程

```
// TODO 在游戏的 Activity 中定义 Gamebar 变量
```

```
MzGameBarPlatform mzGameBarPlatform;
```

```
// TODO 在 activity 的各个声明周期中增加对应的 Gamebar 方法
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    // TODO 初始化，可以指定 Gamebar 第一次显示的位置，在游戏退出时会记住用户操作的最后一次位置，再次启动时使用上一次的位置
```

```
    // 第一次显示的位置可以指定四个方向，左上，左下，右上，右下
```

```
    // public static final int GRAVITY_LEFT_TOP = 1;
```

```
    // public static final int GRAVITY_LEFT_BOTTOM = 2;
```

```
    // public static final int GRAVITY_RIGHT_TOP = 3;
```

```
    // public static final int GRAVITY_RIGHT_BOTTOM = 4;
```

```
    mzGameBarPlatform = new MzGameBarPlatform(this, MzGameBarPlatform.GRAVITY_RIGHT_BOTTOM);
```

```
    // TODO 调用 onActivityCreated
```

```
    mzGameBarPlatform.onActivityCreated();
```

```
}
```

```
@Override
```

```
protected void onDestroy() {
```

```
    super.onDestroy();
```

```
    // TODO 调用 onActivityDestroy
```

```
    mzGameBarPlatform.onActivityDestroy();
```

```
}
```

```
@Override
```

```
protected void onResume() {
```

```
    super.onResume();
```

```
    // TODO 调用 onActivityResume
```

```
    mzGameBarPlatform.onActivityResume();
```

```
}
```

```
@Override
```

```
protected void onPause() {
```

```
    super.onPause();
```

```
    // TODO 调用 onActivityPause
```

```
    mzGameBarPlatform.onActivityPause();
```

```
}
```

4.3.3 主动显示和隐藏

悬浮窗情况下是在登录后默认显示的，如要主动隐藏或显示 Gamebar 可使用如下方法：

```
// TODO 主动显示或隐藏 Gamebar:
```

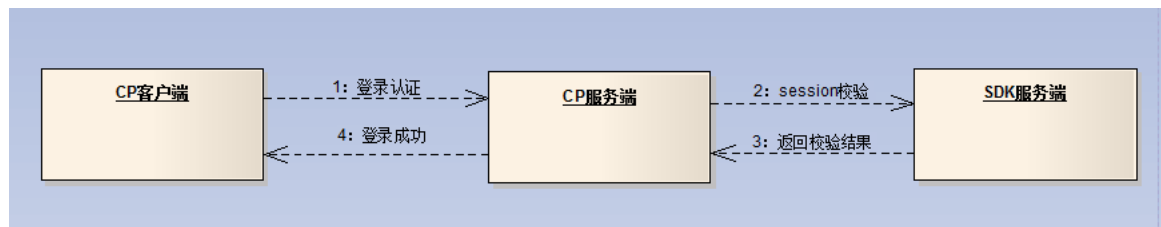
```
mzGameBarPlatform.showGameBar()
mzGameBarPlatform.hideGameBar()
```

五、服务端接入说明

5.1. 接入概述

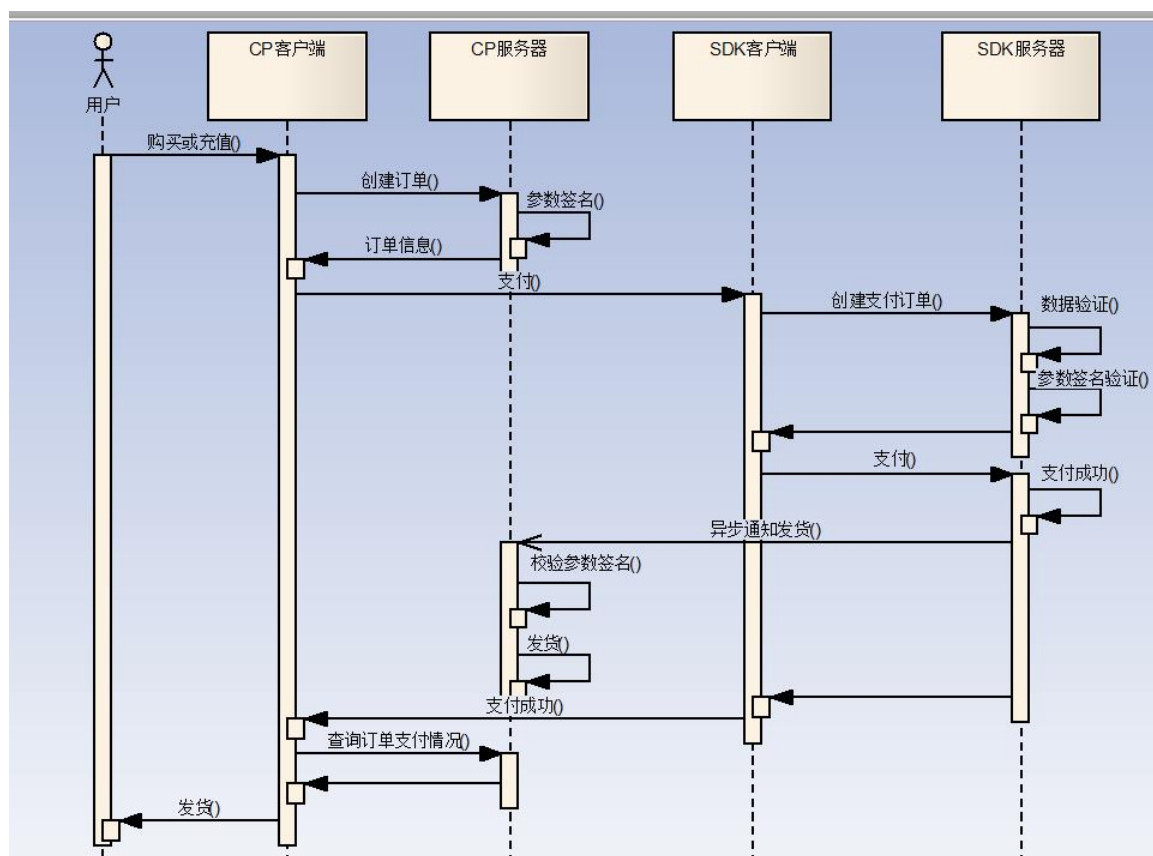
本部分主要提供魅族游戏平台“SDK 服务器”和魅族游戏合作商“CP 游戏服务器”的交互的接口规范。

5.1.1. SESSION 校验流程



SDK 登录成功后，CP 客户端即与 CP 服务器进行登录认证操作。在认证过程中，CP 服务器需要通过 SDK 服务器校验客户端提交的 `session_id` 是否有效。接口详情见 5.3.1

5.1.2. 购买流程说明



- 1: 用户发起购买请求后，首先 CP 客户端将订单信息提交到 CP 服务端创建订单，并对参数按算法进行签名，服务端将订单信息及签名返回给客户端
- 2: 客户端调用 SDK 相关支付方法，SDK 将引导用户完成支付
- 3: 支付成功后，SDK 服务端会异步通知 CP 服务器进行发货，发货地址为 3.2.2.2.2 中配置的”支付通知 URL”,接口详情见 5.3.4 通知游戏方发货
- 4: 本流程省略了 SDK 与魅族支付中心的支付流程
- 5: CP 服务端创建订单见 5.1.3

5.1.3.创建 CP 服务端订单

用户发起购买操作后，首先需要在 CP 服务器完成订单创建和参数签名，然后将订单信息及参数签名一并返回给客户端，再由交由 SDK 客户端以完成支付流程。

接口由 CP 游戏服务端实现，本节主要提供参数签名示例及相关参数说明

签名算法参考 5.2.2 之签名规则

本节分简要和通用两个场景说明参数签名

支付流程请参考 4.1.2 和 5.1.2 相关的支付流程

相关字段的详细说明请参考 5.3 接口说明

5.1.3.1 简要场景

简要场景旨在快速集成游戏 SDK，CP 服务端关注最基本的参数值，其它参数使用默认值返回给客户和参数签名。本方式适用于购买游戏币(金币，钻石)等场景下的充值。

字段名称	字段类型	必填	字段说明
app_id	int	Y	游戏 ID(不能为空)
cp_order_id	String	Y	CP 定单 ID(不能为空)
uid	long	Y	游戏玩家 ID(不能为空)
product_id	String	Y	CP 游戏道具 ID,默认值: "0"
product_subject	String	Y	订单标题,格式为: "购买 N 枚金币"
product_body	String	Y	游戏道具说明, 默认值: ""
product_unit	String	Y	游戏道具的单位, 默认值: ""
buy_amount	int	Y	道具购买的数量, 默认值: "1"
product_per_price	String	Y	游戏道具单价, 默认值: 总金额
total_price	String	Y	总金额
create_time	long	Y	创建时间戳
pay_type	int	Y	支付方式, 默认值: "0" (即定额支付)
user_info	String	Y	CP 自定义信息, 默认值: ""
sign	String	Y	参数签名(不能为空)
sign_type	String	Y	签名算法, 默认值: "md5"(不能为空)

CP 自定义信息 user_info

CP 接入时可以用自定义信息附加相关资料，如角色名称，区服号等，在支付完成后，此信息将回传给 CP 服务端

签名串示例：

```
app_id=464013&buy_amount=1&cp_order_id=2680&create_time=139868782768&pay_type=0&product_body=
&product_id=0&product_per_price=1.0&product_subject=  购    买    500    枚    金    币    &product_unit=
&total_price=1.0&uid=5535004&user_info=:appSecret
```

响应示例:

```
{ "code" : 200,
  "message" : "",
  "redirect" : "",
  "value" : {
    "app_id" : 464013,
    "buy_amount" : 1,
    "cp_order_id" : 2680,
    "create_time" : 139868782768,
    "pay_type" : 0,
```

```

    "product_body" : "",
    "product_id" : "0",
    "product_per_price" : "1.0",
    "product_subject" : "购买500枚金币",
    "product_unit" : "",
    "total_price" : "1.0", //双引号是必须的, 否则存在转化过程丢失0的情况
    "uid" : "2025201",
    "user_info" : "",
    "sign" : "321a14bc32a95c0922d7ef7882345923",
    "sign_type" : "md5"
}

```

如果联调时提示参数签名不正确, 则请比较 CP 服务端的签名串是否与提示的一致, 常见问题请查阅文本后的 FAQ。

5.1.3.2 通用场景

通用场景是指一般需要定制的情况, 包括不定金额支付以更详细的订单信息。

字段名称	字段类型	必填	字段说明
app_id	int	Y	游戏 ID(不能为空)
cp_order_id	String	Y	CP 定单 ID(不能为空)
uid	long	Y	游戏玩家 ID(不能为空)
product_id	String	Y	CP 游戏道具 ID
product_subject	String	Y	订单标题
product_body	String	Y	游戏道具说明
product_unit	String	Y	游戏道具的单位
buy_amount	int	Y	道具购买的数量
product_per_price	String	Y	游戏道具单价
total_price	String	Y	总金额
create_time	long	Y	创建时间戳
pay_type	int	Y	支付方式
user_info	String	Y	CP 自定义信息
sign	String	Y	参数签名(不能为空)
sign_type	String	Y	常量: "md5"(不能为空)

a) 支付方式 pay_type

0: 定额支付, 例如, 订单生成时金额为 100 元, 到账也 100 元

1: 不定金额支付, 支付过程中用户可以修改充值金额, 到账以用户实际充值金额为准

b) 订单标题 product_subject

当 pay_type=0 时:

product_subject= “购买” +数量+单位+产品名称

当 pay_type=1 时, **total_price 不参与签名**, 用户在充值过程中可以修改金额

product_subject= “充值” +游戏名称

c) CP 自定义信息 user_info

CP 接入时可以用自定义信息附加一些相关的资料, 如角色名称, 区服号等, 在支付完成后, 此信息将回传给 CP 服务端

签名串示例:

app_id=464013&buy_amount=1&cp_order_id=2680&create_time=139868782768&pay_type=0&product_body=这里填产品的说明,没有用空串&product_id=2&product_per_price=1.0&product_subject=购买500枚金币&product_unit=枚&total_price=1.0&uid=5535004&user_info=这里填写游戏相关附加信息, 发货时将回传该字段:appSecret

如果联调时提示参数签名不正确, 则请比较 CP 服务端的签名串是否与提示的一致, 常见问题请查阅文本后的 FAQ

凡浮点类型的数据如 total_price, product_per_price 等字段在生成 JSON 时需要以字符串输出, 如下:

```
{ "code": 200,
  "message": "",
  "redirect": "",
  "value": { "app_id": 464013,
    "buy_amount": 1,
    "cp_order_id": 2680,
    "create_time": 139868782768,
    "pay_type": 0,
    "product_body": "这里填产品的说明,没有说明用空串",
    "product_id": "2",
    "product_per_price": "1.0",
    "product_subject": "购买500枚金币",
    "product_unit": "枚",
    "sign": "321a14bc32a95c0922d7ef7882345923",
    "sign_type": "md5",
    "total_price": "1.0", //双引号是必须的, 否则存在转化过程丢失0的情况
    "uid": "2025201",
    "user_info": "这里填写游戏相关附加信息, 发货时将回传该字段"
  }
}
```

```
}
```

5.2. 协议说明

5.2.1. 通信协议

“SDK 服务器”采用 HTTP 协议作为通信协议，“游戏服务器”通过构造 HTTP 请求（POST 方式）向“SDK 服务器”发起接口请求。

手机端或网页端与服务端交互 API 协议数据格式统一如下：

协议JSON对象：

```
{
  "code": "", //必选，返回码
  "message": "", //可选，返回消息，网页端接口出现错误时使用此消息展示给用户，
  //手机端可忽略此消息，甚至服务端不传输此消息。
  "value": "", //必选，返回结果
  "redirect": "" //可选，当returnCode=300 重定向时，使用此URL重新请求
}
```

数据结构由 JSON 对象构成，外层对象为调用约定对象，里层对象为当前调用接口返回的具体数据对象（被放在外层对象 value 里）。

所有 API 需要有一份定义明确的错误码对照表，提供给客户端开发人员使用。

5.2.2. 数据协议

1) 数据格式

请求信息采用 HTTP 协议 Key-Value 方式提交，响应消息的内容使用 JSON 格式表示数据，具体请参考下文的示例。

2) 字符编码

请求与响应内容须采用 utf-8 字符编码。

3) 签名规则

1. 参数排序：将所有参数名按字母顺序进行排序,没有值的参数不要参与签名。
2. 参与签名的数据不要做 URL Encoding，一律以 UTF-8 编码参与签名。
3. 参数拼接：将所有参数按 k1=v1&k2=v2&k3=v3...格式进行拼接，不包含 sign 和 sign_type。
4. 签名：将第 3 步所得字符串+”.”+appSecret 用 md5 计算摘要，得签名字符串。

示例：购买结果通知签名前的串可能如下

```
body=coins&notify_id=23344555&notify_time=2014-04-01 13:12:56&out_trade_no=xxxssd1122
&subject=coins&total_fee=120.00 &trade_status=3:appSecret
```

如上待签名字符串中的值，必须经过 UTF-8 编码，其中 appSecret 从开发者平台的应用详情里获取，appSecret 勿泄露给他人，也不要打包在 APK 包里，否则会有安全风险。

5.3. 接口说明

5.3.1. SESSION 校验

<https://api.game.meizu.com/game/security/checksession>

SDK 登录成功后，CP 客户端即与 CP 服务器进行登录认证操作。在认证过程中，CP 服务器需要通过 SDK 服务器校验客户端提交的 session_id 是否有效

校验 session 接口	请求地址	https://api.game.meizu.com/game/security/checksession		
	调用方式	https post		
	请求方	CP 游戏服务器		
	响应方	游戏中心		
	同步 or 异步	同步		
请求内容：				
字段名称	字段类型	必填	字段说明	
app_id	long	Y	游戏 id	
session_id	String	Y	sessionId	
uid	long	Y	用户 id	
ts	long	Y	timestamp.eg: 1396424644001	
sign_type	String	Y	常量:md5	
sign	String	Y	签数签名	

响应(response):

响应内容:			
协议: https + json			
字段名称	字段类型	必填	字段说明
code	String	Y	200 成功 198001 APP 不存在 198005 用户无效或非法 100000 未知异常
message	String	N	
value	String	N	成功返回如下: value : null

5.3.2. 查询订单接口

<https://api.game.meizu.com/game/order/query>

查询订单接口	请求地址	https://api.game.meizu.com/game/order/query	
	调用方式	http post	
	请求方	CP 游戏服务器	
	响应方	游戏中心	
	同步 or 异步	同步	
请求内容:			
字段名称	字段类型	必填	字段说明
app_id	long	Y	游戏 id
cp_order_id	String	Y	游戏生成的 order_id
ts	long	Y	Unix_timestamp.eg: 1396424644
sign_type	String	Y	常量:md5
sign	String	Y	签数签名

响应(response):

响应内容:			
协议: http+json			
字段名称	字段类型	必填	字段说明
appId	long	Y	游戏 Id
buyAmount	String	N	购买数量
cpOrderId	String	N	游戏定单 Id
orderId	String	Y	SDK 服务器定单 Id
partnerId	String	N	开发者 Id
uid	long	Y	用户 Id
productId	String	Y	产品 Id
productSubject			定单标题
productBody			产品详情
productUnit			产品单位
buyAmount			购买数量
productPerPrice			单价
totalPrice			总价
tradeStatus			交易状态, 1: 待支付 (订单已创建) 2: 支付中 3: 已支付 4: 取消订单 5: 未知
createTime			创建时间
payTime			支付时间
deliverStatus			发货状态, 1: 支付未完成, 2: 待发货: 3: 已发货, 4: 发货失败

游戏服务端需要对app_id,cp_order_id, ts用appSecret进行参数签名。

5.3.3. 通知游戏方发货

申请时自定义配置

请求(request):

查询版本号	请求地址	由游戏开发者配置	
	调用方式	http post	
	请求方	游戏中心	
	响应方	游戏服务端	
	同步 or 异步	同步	
请求内容：			
字段名称	字段类型	必填	字段说明
notify_time	Date	Y	通知的发送时间
notify_id	String(32)	Y	通知 id
order_id	Long	Y	订单 id
app_id	Long	Y	应用 id
uid	long	Y	用户 id
partner_id	long	Y	商户 id
cp_order_id	String	Y	游戏订单 id
product_id	String	Y	产品 id
product_unit	int	N	产品单位
buy_amount	int	N	购买数量
product_per_price	float	n	产品单价
total_price	float	Y	购买总价
trade_status	String	Y	交易状态： 1：待支付（订单已创建） 2：支付中 3：已支付 4：取消订单 5：未知异常取消订单
create_time	Date	Y	订单时间
pay_time	Date	Y	支付时间
pay_type	int	N	支付类型：1 不定金额充值，0 购买
user_info	String	N	用户自定义信息
sign	String	Y	参数签名
sign_type	String	Y	签名类型，常量 md5

响应(response):

响应内容:			
协议: http + json			
字段名称	字段类型	必填	字段说明

code	String	Y	200 成功发货 120013 尚未发货 120014 发货失败 900000 未知异常
message	String	N	
value	String	N	
redirect	String	N	

购买成功后会调用游戏提供商的该接口。如果第一次调用失败，会重复 3 次调用该接口。系统可能会重复通知，商户必须保证对于多次重复调用与一次调用的结果是一致的。游戏服务端需要作参数签名验证，否则存在安全风险。签名示例：

```
app_id=464013&buy_amount=1&cp_order_id=2680&create_time=1413776092239&notify_id=1413776113206&notify_time=2014-10-20
11:35:13&order_id=14102000000298934&partner_id=5458428&pay_time=1413776113219&pay_type=0&product_id=2
&product_per_price=1.0&product_unit=枚&total_price=1.0&trade_status=3&uid=9700193&user_info=这里填写游戏相关附加信息，发货时将回传该字段:appSecret
```

六、发布联网手游

6.1. 完善游戏信息

1. 登录开发者社区，进入游戏应用管理列表
2. 找到之前创建的游戏信息记录，点击【添加版本】到应用信息编辑页面

应用名称	版本	价格	状态	评分	操作
 卡通农场			创建中		添加版本 删除

3. 选择已测试通过的游戏正式包，按照引导完善其他游戏信息资料，点击【完成】提交游戏待审核
4. 然后请留意魅族官方为您的游戏包和信息内容进行审批的结果，审批通过后将自动上线

6.2. 游戏数据统计查询

- 6.2.1 登录开发者社区，点击 统计报告>游戏

七、常见问题

7.1. 已经接入 SDK1.0 版本如何升级

已接入 SDK1.0 版本，即已经实现了登录、支付流程，可以直接查阅《4.3 悬浮窗口 Gamebar》一节即可，其他内容与 SDK1.0 版本一致

7.2. 调试登录时，提示“游戏不存在”或“游戏签名不正确”

SDK 服务端会对参数作必要的校验，校验不通过则有此提示

1. 联系运营检查该游戏是否打开了联调状态
2. 联系运营检查是否创建了该游戏
3. 上架后的游戏需要从游戏中心下载 APK 包进行安装

7.3. 联调时出现“游戏 ID 参数无效”

后台会验证游戏 ID 是否为空，是否为数字，如果验证不通过即有此提示

1. CP 需要检查游戏 ID 是否给定正确的值
2. 部分语言如 PHP 作 POST 提交时需要将表单类型转为 x-www-form-urlencoded, 否则可能取不到参数
3. SDK 服务端仅接收 KEY-VALUE 形式提交的参数, 参数格式不接受 JSON 及数组的形式

7.4. 支付时出现“参数签名不正确”+一串签名串

在支付流程中参数签名校验不通过.

1. 检查是否使用了错误的 app_secret, 在集成时，常有错用 app_key 进行签名
2. 在对参数用 app_secret 签名时，参数没有按要求的顺序生成签名串
3. md5 算法不一致，检查 md5(“中国”)为" c13dceabcb143acd6c9298265d618a9f"
4. 签名的参数和提交的参数是要一致的，此时需要比较服务端的签名串与异常提示的签名串是否存在不同，尤其注意不要漏掉相关参数
5. 浮点类型的数据注意用字符串参与签名及传递，防止丢失末位的 0

7.5. 支付时如何作参数签名

签名规则按 文档 5.2.2 数据协议约定进行签名

具体可以参考 5.3.2 签名串示例，为 NULL 的值用""空串代替,数字类型的用 0，同时客户端也""串或 0 提交参数

7.6. 支付成功，收不到发货回调

支付成功后，SDK 服务端会异步通知 CP 服务端进行发货

1. 检查回调地址是否正确，如拼写错误。
2. 回调地址不支持 ssl 协议，即不能用 https 配置回调地址
3. 回调地址本身服务不可用，需检查网络，域名等相关是否处于可用状态
4. 回调地址不能存在非法字符，&，空格等字符将会被转义，导致回调地址不可用
5. 在联调状态修改回调地址时，需要重新设定回调

7.7. 发货回调如何作签名验证

分发货回调及发货成功返回

1. SDK 服务端发货回调按下面模板签名

app_id=464013&buy_amount=1&cp_order_id=2680&create_time=1413776092239¬ify_id=1413776113206¬ify_time=2014-10-20 11:35:13&order_id=14102000000298934&partner_id=5458428&pay_time=1413776113219&pay_type=0&product_id=2&product_per_price=1.0&product_unit= 个
&total_price=1.0&trade_status=3&uid=9700193&user_info=其它回传附加信息:appSecret

2. 发货成功必须以如格式返回

```
{"code":200,"message":"","redirect":"","value":null}
```

200 成功发货，120013 尚未发货，120014 发货失败，900000 未知异常

7.8. 悬浮窗不显示

1. 先确认已按文档说明在 Activity 的各个生命周期调用了相应的方法
2. 再检查是否是 MIUI 等有管理悬浮窗权限的系统,如果 MIUI 这些系统请检查魅族游戏框架是否具备了悬浮窗权限，打开后即可正常显示

7.9. 游戏退出后悬浮窗还显示在桌面

游戏退出时如果是使用 `System.exit(0)` 或者是 `killProcess` 的方式退出的，请在退出之前主动调用下 `gamebar` 的 `onActivityPause` 和 `onActivityDestroy` 这两个方法。

7.10. 游戏框架安装时出现解析包错误

目前游戏框架只支持安装在 4.2(包括 4.2)以上机器，4.2 以下机器会出现此问题

7.11. 其它常见问题

序号	问题关键字	相应问题解答
1	实名认证	请提供 账号名+公司名
2	调试 SN	指的是魅族手机的序列号，可以随意填写，不影响 SDK 的接入
3	支付通知 URL	即充值回调地址，需要您的服务端技术提供
4	包名	包名的格式要求以 .mz 后缀结束，如：com+公司名+游戏名+.mz
5	角标	游戏 ICON 暂不需要添加角标
6	闪屏	闪屏中只出现与游戏相关的信息