# Meizu 游戏聚合 SDK 接入指导文档

# V2.1.0

| 版本号 | 时间 | 变更内容 |
| --- | --- | --- |
| 1.0.0 | 2016-03-08 | 初始版本 |
| 1.1.0 | 2016-03-22 | 添加数据统计接口 |
| 1.1.4 | 2016-03-23 | 添加 jar 包引入方式，添加支付参数描述，客户端初始化参数补充，角色埋点数据添加工会 |
| 1.1.6 | 2016-04-01 | 修复数据统计 bug |
| 1.1.7 | 2016-05-11 | 添加修改部分接口描述 |
| 1.1.8 | 2016-05-18 | 修改充值回调接口的参数名称（游戏订单 ID 和订单创建时间的字段名称） |
| 1.1.9 | 2016-07-04 | 添加四个数据埋点 |
| 1.1.10 | 2016-07-11 | 修改数据统计接口,添加一些必须的权限和组件 |
| 2.0.0 | 2016-07-29 | 添加新渠道，新接口，优化数据统计接口 |
| 2.0.1 | 2017-01-10 | 添加 7 个数据埋点 |
| 2.0.2 | 2017-03-16 | buyInfo 添加字段角色 id 和角色名称 |
| 2.1.0 | 2017-06-16 | 添加新的扩展接口 |

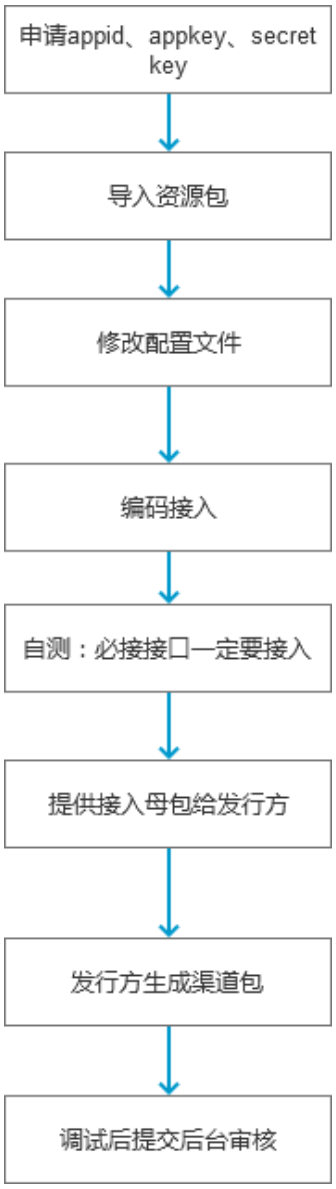
# 目录:

# 1 SDK 客户端使用指南

## 1.1 概述

本文档面向安卓开发者。

本文档用于指导开发者快速接入魅族游戏发行客户端SDK,本 SDK 为安卓游戏提供聚合其它渠道登录，注册（如果渠道商提供）和支付的功能。



## 1.2 接入流程

### 1.2.1 申请 APPID、APPKEY 和 SECRETKEY

接入方需要联系发行方申请相关参数

### 1.2.2 导入资源包

本 SDK 目前支持 Android2.2 及以上的系统版本,为兼容 Android 4.0 及以上的新手机,编 译时请使用 Android4.2 或以上的版本 。
SDK 支持两种方式导入：

#### 1.2.2.1 aar 包导入（Android studio 工程）

将 SDK 包内 aar 包导入的 libs 目录下的文件(夹)放到应用工程的 libs 目录下。

#### 1.2.2.2 Jar 包导入（Eclipse 工程）

将 SDK 包内 Jar 包导入目录下的 libs 目录下的文件(夹)放到应用工程的 libs 目录下，并将 Jar 包导入目录下的 res 目录下的文件放到工程的 res 目录下。

### 1.2.3 配置应用工程的 AndroidManifest.xml

**注意：cp 自己的组件配置务必写全名，因为打出的渠道包包名配置各不相同，这样会找不到 cp 自己的组件。**

#### 1.2.3.1 添加权限

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<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.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="com.meizu.stats.permission.WRITE_USAGESTATS" />
<uses-permission android:name="com.meizu.flyme.permission.UPDATE"/>
```

```
<uses-permission android:name="android.permission.ROOT_RECOVERY_STATE."/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="com.meizu.stats.permission.READ_USAGESTATS"/>
<uses-permission android:name="android.permission.LOCAL_MAC_ADDRESS"/>
```

### 1.2.3.2 在 AndroidManifest.xml 中添加组件

```
<service android:name="com.meizu.flyme.gamepolysdk.update.service.MzUpdateComponentService"
android:exported="false" android:stopWithTask="false"/>
<activity android:name="com.meizu.flyme.gamepolysdk.update.display.KeyguardHelperActivity"
        android:theme="@style/MzucActivityNoDisplay"
        android:exported="false"/>
<receiver android:name="com.meizu.gslb.push.GslbDataRefreshReceiver">
        <intent-filter>
            <action android:name="com.meizu.flyme.gslb.push.broadcast" />
        </intent-filter>
</receiver>
```

### 1.2.3.3 在 AndroidManifest.xml 中添加 meta-data

注意:必须放入<application>元素区块内:

```
<meta-data
                android:name="channel"
                android:value="empty" />
```

### 1.2.3.4 继承 MzPolySdkApplication

如游戏客户端使用默认 application，则需要设置 MzPolySdkApplication 为客户端 application，在 AndroidManifest.xml 中 application 节点配置

```
<application android:name="com.meizu.flyme.gamepolysdk.MzPolySdkApplication" >
```

如有自定义的 application，则需在代码中继承 MzPolySdkApplication。

## 1.2.4 配置应用工程的 build.gradle(仅 Android studio 工程用 aar 包引入需要)

build.gradle 里面添加如下代码

```
repositories{
        flatDir{
            dirs 'libs'
        }
    }
```

dependencies 里面加入 compile(name:'MzPolySDK-release', ext:’aar')

最终的 build.gradle 文件结构大致如下：

```
apply plugin: 'com.android.application'

android {
    .
    .
    .
    repositories{
        flatDir{
            dirs 'libs'
        }
    }
.
.
.
}
dependencies {
    .
    .
    .
    compile(name:'MzPolySDK-release', ext:'aar')
  }
```

## 1.2.5 编码接入

为了保证所有渠道都能成功运行，请在 activity 中调用接口方法，或使用 activity.RunOnUIThread 在 ui 线程中调用接口方法。

### 1.2.5.1 初始化接口【客户端调用】(必接)

```
MzPolySdk.init(Activity activity, String polyAppId, String polyAppKey, String polyAppSecret, MzPolySdk.InitCallback callback);
```

建议在主 Activity 的 onCreate 里面调用。

### 1.2.5.2 Activity 方法处理【客户端调用】(必接)

在游戏主 activity 里面的对应方法中调用如下对应方法：

```
MzPolySdk.getPolySdkInterface().onActivityBackPress(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityResult(Activity activity, int requestCode, int resultCode, Intent data);
MzPolySdk.getPolySdkInterface().onActivityRestart(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityStart(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityNewIntent(Intent intent);
MzPolySdk.getPolySdkInterface().onActivityResume(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityPause(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityStop(Activity activity);
MzPolySdk.getPolySdkInterface().onActivityDestory(Activity activity);
```

### 1.2.5.3 登录调用方法【客户端调用】(必接)

```
MzPolySdk.getPolySdkInterface().login(Activity activity, LoginCallback callback);
```

LoginCallback 用法如下，回调方法都是必须要处理的：

```
private MzPolySdk.LoginCallback loginCallback = new MzPolySdk.LoginCallback() {
    @Override
    public void onLoginSuccess(final PolyUserInfo userInfo) {//登录成功
    }

    @Override
    public void onLoginError(final int code, final String msg) {//登录失败
    }

    @Override
    public void onLogoutSuccess() {//登出成功，一些渠道中会有在悬浮窗中做切换账号的功能，会先处理登出的消息，一般为回到游戏登陆页面，重新调用聚合 sdk 的登录接口，
    }

    @Override
    public void onLogoutError(int code, String msg) {//登出失败
    }

    @Override
    public void onSwitchAccountSuccess(PolyUserInfo userInfo) {
        //一些渠道中会有在悬浮窗中做切换账号的功能，游戏做切换账号的逻辑，一般为回到游戏登陆页面，重新调用聚合 sdk 的登录接口
    }
};
```

### 1.2.5.4 登出接口【客户端调用】(必接)

游戏注销的时候，需要调用此接口。

```
MzPolySdk.getPolySdkInterface().logout(getActivity());
```

### 1.2.5.5 支付调用方法【客户端调用】(必接)

```
MzPolySdk.getPolySdkInterface().pay(Activity activity, BuyInfo buyInfo, PayCallback callback);
```

支付参数如下：

```
BuyInfo buyInfo = new BuyInfo();
buyInfo.orderId = "";//订单 id,必填
buyInfo.buyCount = 1;//购买数量,必填
buyInfo.productName = "";//商品名称,必填
buyInfo.productSubject = "";//商品主题,必填,不确定可以填商品名称
buyInfo.productBody = "";//商品内容,必填,不确定可以填商品名称
buyInfo.productPerPrice = "";//商品单价,必填,以分为单位
buyInfo.productDesc = "";//商品描述,必填,不确定可以填商品名称
buyInfo.totalPrice = "";//商品总价,必填,单位元
buyInfo.productId= "";//商品 id,必填
buyInfo.extendParam= "";//扩展参数，透传给服务端
buyInfo.gameRoleId = "";//角色 id，必填，某些渠道要求传入
buyInfo.gameRoleName = "";//角色名称，必填，某些渠道要求传入
buyInfo.gameServerId = "";//服务器 id，必填，某些渠道要求传入
```

PayCallback 用法如下

```
@Override
    public void onPayed(buyInfo ) {//支付完成，客户端的支付流程已结束，是否到账已服务端回调通知为准
    }
@Override
public void onError(int code, String msg
) {//支付错误
   }
```

### 1.2.5.6 销毁接口【客户端调用】(必接)

在游戏退出时首先调用销毁接口，回调为 true 时再调用游戏自己的退出逻辑，

```
MzPolySdk.destory(Activity activity, DestoryCallback callback);
```

用法如下：

```
MzPolySdk.destory(this, new MzPolySdk.DestoryCallback() {
    @Override
    public void onDestory(boolean isSdkExit) {
        if(isSdkExit){
            finish();
        } else {
            //cancel exit
        }
    }
});
```

### 1.2.5.7 提交角色数据【客户端调用】(必接)

一些渠道要求提交角色的数据，如 UC，华为等。统一通过以下接口提交：

```
setGameRoleData(GameRole gameRoleData, Activity activity, Action action);
```

其中角色信息 GameRole 的填写参考下方的示例。此接口在创建角色，角色进入游戏以及角色升级时都要调用。

Action 类：

```
public enum Action{
    CREATE,
    LOGIN,
    UPGRADE
}
```

角色信息类:

```
GameRole gameRole = new GameRole();
//设置角色 id
gameRole.setRoleId("111");
//设置角色名称
gameRole.setRoleName("肥猪流名字脑残");
//设置角色等级
gameRole.setRoleLevel("0");
//设置角色职业
gameRole.setRoleClass("战士");
//设置角色工会
gameRole.setRoleParty("HQ 工会");
//设置角色所在区服 id
gameRole.setZoneId(1);
//设置角色所在区服名称
gameRole.setZoneName("沧海桑田");
```

### 1.2.5.8 检查渠道是否支持扩展方法【客户端调用】(必接)

```
isChannelFunctionSupported(final int funcType);
```

某些 SDK 具有打开关闭浮动工具栏、进入用户中心、进入论坛等功能，该方法可以用来调用第三方渠道支持的方法，调用前请先用 isChannelFunctionSupported 进行判断，目前判断

渠道是否有退出框必接。
支持的方法列表：
Constans.FuncType

```
public static final class FuncType{
    public static final int UNDEFIEND = 0;
    public static final int ENTER_BBS = 101;
    public static final int ENTER_PLATFORM = 102;
    public static final int SHOW_TOOLBAR = 103;
    public static final int HIDE_TOOLBAR = 104;
    public static final int REAL_NAME_REGISTER = 105;
    public static final int ANTI_ADDICTION_QUERY = 106;
    public static final int SWITCH_ACCOUNT = 107;
    public static final int SHARE = 108;
    //防沉迷
    public static final int ISINDULGE = 109; //必接
    //战斗结束数据上报
    public static final int GAME_BATTLE = 110;
    public static final int EXIT_DIALOG = 1001;//必接
}
```

## 1.2.5.9 调用渠道扩展方法【客户端调用】(必接)

| |
|---|
| callChannelFunction(final Activity activity, final int funcType);//游戏退出时调用 |
| callChannelFunction(final Activity activity, String prop, final int funcType, ExtCallBack extCallBack);（新添加的扩展接口，防沉迷专用，prop 传""） |

代码可以从 demo 中 copy：

```
//进入游戏
if ((!MzPolySdk.getPolySdkInterface().isChannelFunctionSupported(Constans.FuncType.ISINDULGE))) {
    Toast.makeText(getActivity(), "不支持防沉迷！", Toast.LENGTH_SHORT).show();
    getFragmentManager().beginTransaction().replace(R.id.fragment, GameFragment.newInstance()).commitAllowingStateLoss();
} else {
    MzPolySdk.getPolySdkInterface().callChannelFunction(getActivity(), "", Constans.FuncType.ISINDULGE, new MzPolySdk.ExtCallBack() {
        @Override
        public void onSuccess(String msg) {
            //解析年龄age
            try {
                int age = Integer.parseInt(msg);
                if (age < 18) {
                    Toast.makeText(getActivity(), "已实名但未成年，CP开始处理防沉迷", Toast.LENGTH_SHORT).show();

                } else {
                    Toast.makeText(getActivity(), "已实名且已成年，尽情玩游戏吧~~", Toast.LENGTH_SHORT).show();
                }
                getFragmentManager().beginTransaction().replace(R.id.fragment, GameFragment.newInstance()).commitAllowingStateLoss();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onError(boolean isSupport, String msg) {
            if (isSupport) {
                Toast.makeText(getActivity(), msg + "，还可以继续玩游戏", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getActivity(), msg + ",CP自己处理退出游戏", Toast.LENGTH_SHORT).show();
            }
            getFragmentManager().beginTransaction().replace(R.id.fragment, GameFragment.newInstance()).commitAllowingStateLoss();
        }
    });
}
```

## 1.2.5.10 数据统计【客户端调用】

数据统计分为两类，一类是点击事件，另一类是页面事件。

数据统计相关接口调用的前提是接口初始化代码已经执行，即如下代码：MzPolySdk.init(Activity activity, String polyAppId, String polyAppKey, String polyAppSecret, MzPolySdk.InitCallback callback);

具体埋点接口如下，：

以下方法均在 UsageStatsAssist 中

1.点击事件

```
登录游戏区服
/**
 * 登录游戏区服埋点
 * @param accountId
 * @param gseId
 * @param page
 */
public static void onEnterGameZone(String accountId, String gseId, String page)
```

```
点击商城
点击游戏设置
点击游戏角色账号
点击游戏通知
跳过剧情
点击技能菜单项
点击排行榜
/**
 * 普通点击事件埋点
 * @param accountId
```

```
 * @param gseId
 * @param page
 * @param eventType
 * @param roleId  角色 id
 */
public static void onNormalClick(String accountId, String gseId,
            String page, EventType eventType, String roleId)

EventType  类型如下：
public enum EventType{
      CLICK_SETTING,//点击设置
      CLICK_ROLE_ACCOUNT,//点击游戏角色账号
      CLICK_ANNOUCEMENT,//点击游戏通知
      CLICK_SKIP,//点击跳过剧情
      CLICK_SKILL,//点击技能菜单项
      CLICK_RANK,//点击排行榜
      CLICK_MALL,//点击商城
}
```

```
点击领取任务
/**
 *  领取任务埋点
 * @param missionId  任务 id
 * @param missionName  任务名称
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId  角色 id
 */
public static void onGetMission(String missionId, String missionName, String accountId, String gseId,
String page, String roleId)
```

```
完成任务
/**
 *  完成任务埋点
 * @param missionId  任务 id
 * @param missionName  任务名称
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId  角色 id
 */
public static void onMissionFinish(String missionId, String missionName, String accountId,
                                   String gseId, String page, String roleId)
```

```
点击游戏关卡
/**
 *  点击关卡埋点
 * @param gameStageId  关卡 id
 * @param gameStageName  关卡名称
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId  角色 id
```

```
 */
public static void onClickStage(String gameStageId, String gameStageName, String accountId, String
gseId, String page, String roleId)
```

```
完成游戏关卡
/**
 * 完成关卡埋点
 * @param gameStageId 关卡 id
 * @param gameStageName 关卡名称
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId 角色 id
 */
public static void onStageFinsh(String gameStageId, String gameStageName, String accountId,
                                String gseId, String page, String roleId)
```

```
点击技能升级
/**
 * 点击技能升级埋点
 * @param beforeLevel 升级前技能等级
 * @param afterLevel 升级后技能等级
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId 角色 id
 * @param polyAppId 已废弃，填空即可
 */
public static void onSkillUpdateClick(String beforeLevel, String afterLevel, String accountId, String gseId,
String page, String roleId, String polyAppId)

点击礼包
/**
 * 点击礼包埋点
 * @param giftId 礼包 id
 * @param giftName 礼包名称
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId 角色 id
 * @param polyAppId 已废弃，填空即可
*/
public static void onGiftClick(String giftId, String giftName, String accountId, String gseId,
String page, String roleId, String polyAppId)
点击奖励
/**
 * 点击领取奖励埋点
 * @param rewardId 奖励 id
 * @param rewardSource 奖励来源
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId 角色 id
 * @param polyAppId 已废弃，填空即可
 */
public static void onRewardClick(String rewardId, String rewardSource, String accountId, String gseId,
```

```
String page, String roleId, String polyAppId)
```

```
用户退出
/**
 * 退出埋点
 * @param taskId 退出前的任务 id，-1 代表不在任务中
 * @param lastGameLevel 退出前的角色等级
 * @param accountId
 * @param gseId
 * @param page
 * @param roleId 角色 id
 * @param polyAppId 已废弃，填空即可
 */
public static void onLogout(String taskId, String lastGameLevel, String accountId, String gseId, String
page, String roleId, String polyAppId)
```

```
点击新手指引
/**
 * 点击新手指引埋点
 * @param roleId 角色 id
 * @param guideId 新手引导步骤 id
 * @param accountId 账户 id
 * @param gseId 区服 id
 * @param page
 */
public static void onClickGuide(String roleId, String guideId, String accountId, String gseId,
                                String page)
```

```
完成新手指引
/**
 * 完成新手指引埋点
 * @param roleId 角色 id
 * @param accountId 账户 id
 * @param gseId 区服 id
 * @param page
 */
public static void onGuideFinish(String roleId, String accountId, String gseId,
                                 String page)
```

2.页面事件

需要埋点的页面如下：

```
public enum PageType{
    PAGE_LOGIN,//游戏登录页面
    PAGE_MAIN,//游戏主页
    PAGE_COMMODITY_MALL,//游戏商城页面
    PAGE_CHARGE,//游戏支付页面
    PAGE_EQUIPMENT,//游戏道具购买页面
    PAGE_MISSION,//任务页面
    PAGE_GAME_LEVEL,//游戏关卡页面
```

```
    PAGE_ROLE_ACCOUNT,//角色账号页面
    PAGE_SETTING,//设置页面
    PAGE_GAME,//游戏页面
    PAGE_ANNOUCEMENT,//通知页面
    PAGE_RANK,//排行页面
    PAGE_GIFT,//礼物页面
    PAGE_GUIDE,//指引页面
    PAGE_REWARD//奖励页面
}
```

```
页面开始
/**
 * @param accountId
 * @param gseId
 * @param pageType  页面类型
 * @param roleId     角色id
 */
public static void onGamePageStart(String accountId, String gseId, PageType pageType, String roleId)
```

```
页面结束
/**
 * @param accountId
 * @param gseId
 * @param pageType  页面类型
 * @param roleId     角色id
 */
public static void onGamePageStop(String accountId, String gseId, PageType pageType, String roleId)
```

# 2 SDK 服务端接口开发指南

## 2.1 接口说明

**服务器域名：**

https://poly-game.meizu.com/ （线上环境统一以HTTPS请求）。

**返回内容：**

所有接口返回内容都以JSON格式返回。

**签名规则：**

将参与签名的参数以ASCII码的增序排序，若遇到相同的首字母，则比较第二个字母，以此类推。完成排序后，把所有参数以“&”字符连接起来，并在最后以“:”字符将魅族分配的app_key或者app_secret【app_key、app_secret使用那个，请看据体接口文档】连接起来如：

【

**MD5(key1=value1&key2=value2&key3=value3&key4=value4&key5=value5:f32fdc02123a82524eb4ea95e1383d0b)**

**】**

假设app_key= f32fdc02123a82524eb4ea95e1383d0b,上面这段字符串便是待签名的完整串。

**注意：**

1. **值为 null 的参数出现在请求中，则不参与签名，如: product_name=null(不参与签名) 。但值为空字符串的参数需要参与签名，如：product_name=""(参与签名)，空字符串的参数参与签名的格式如下：product_name=。**
2. **据体参与签名的参数请看请求参数的“签名”列，Y=参与签名、N=不参与签名。**

# 2.2 校验 access_token 接口

## 2.2.1 接口描述

游戏客户端通过meizu SDK登录成功后,服务端会返回token等数据。当玩家登录游戏时，游戏服务端需要用客户端传过来的token等数据,来SDK服务端进行token校验,校验通过才允许玩家登录，否则登录失败。

## 2.2.2 接口 URL 及参数说明

**请求地址**：/poly/security/check_session

**请求参数**:

| 请求方式 | GET、POST | | | |
|---|---|---|---|---|
| 参数名 | 字段类型 | 必填 | 签名 | 描述 |
| app_id | int | Y | Y | 游戏 id 由 meizu sdk 分配 |
| token | String | Y | Y | access token |
| ts | long | Y | Y | 时间戳 |
| uid | long | Y | Y | meizu sdk 用户 id |

**返回内容：**

| 参数名 | 字段类型 | 必填 | 签名 | 描述 |
|---|---|---|---|---|
| uid | long | Y | N | meizu sdk 用户 id |

**返回示例**:

```
{
  "code": 200,
  "message": "",
  "value": {
    "uid": 163
  }
}
```


## 2.3 充值回调接口

### 2.3.1 接口描述

SDK服务端订单在支付完成后将通过此接口通知游戏服务端给玩家发货.
游戏服务端在收到回调后对回调信息进行校验，校验通过返回JSON格式:{“code”:200}。

通知重发：游戏方没有返回或者返回结果非{“code”:200}，服务端将对这笔订单进行周期性重发通知.
重发机制：持续周期共1个小时，每2分钟一次.
**注意：**
**1、 为保证可靠性,同笔订单存在连续发送多次回调通知的情况,请“游戏服务端”做好重复发货的处理，收到回调通知后应先判断当前订单是否已经处理过。**
**2、 为防止刷单，请校验双方订单金额信息是否一致。**

### 2.3.2 接口 URL 及参数说明

**请求地址**：支付结果通知地址，由游戏合作商提供，在游戏接入时，游戏合作端提供回调地址给魅族录入到开放平台。

**请求参数**:

| 请求方式 | POST | | | |
|---|---|---|---|---|
| 参数名 | 字段类型 | 必填 | 签名 | 描述 |
| order_id | long | Y | Y | 订单 ID |
| cp_order_id | String | Y | Y | 游戏订单 ID |
| game_server_id | String | Y | Y | 游戏区服 ID |
| app_id | int | Y | Y | 游戏 ID |
| uid | long | Y | Y | 魅族 SDK 用户 ID |
| buy_amount | int | N | Y | 购买数量 |
| product_per_price | double | N | Y | 购买商品单价 |

| | | | | |
|---|---|---|---|---|
| total_price | double | Y | Y | 总金额 |
| status | int | Y | Y | 支付状态，1：支付成功 |
| product_name | String | Y | Y | 产品名称 |
| product_desc | String | N | Y | 产品描述 |
| extend_param | String | N | Y | 扩展字段(商户下单时传的参数,原样返回) |
| cTime | long | Y | Y | 订单创建时间 |
| success_time | long | Y | Y | 支付成功时间 |
| notify_time | long | Y | Y | 通知时间 |
| sign | String | Y | | 签名串(用 app_secret 进行签名) |

**返回内容：**

| 参数名 | 字段类型 | 必填 | 签名 | 描述 |
|---|---|---|---|---|
| code | int | Y | N | 200: 成功; 其它: 失败 |

**返回示例：**

```
{
  "code": 200
}
```

# 2.4 订单信息查询接口

## 2.4.1 接口描述

游戏合作商在收到魅族回调通知后，根据订单号等信息，请求魅族服务器查询订单支付信息，用以验证订单有效性。

## 2.4.2 接口 URL 及参数说明

**请求地址**：/poly/order/query

**请求参数**:

| request | GET、POST | | | |
|---|---|---|---|---|
| 参数名 | 字段类型 | 必填 | 签名 | 描述 |
| app_id | int | Y | Y | 游戏 ID |
| cp_orderid | String | Y | Y | 游戏订单 ID |
| ts | long | Y | Y | 时间戳 |
| sign | String | Y | N | 签名串(用 app_secret 进行签名) |

**返回内容：**

| 字段名称 | 字段类型 | 必填 | 签名 | 字段说明 |
|---|---|---|---|---|
| order_id | int | Y | Y | 订单 ID |
| cp_orderid | String | Y | Y | 游戏订单 ID |
| game_server_id | String | Y | Y | 游戏区服 ID |
| app_id | int | Y | Y | 游戏 ID |
| uid | long | Y | Y | 魅族 SDK 用户 ID |
| buy_amount | int | N | Y | 购买数量 |
| product_per_price | double | N | Y | 购买商品单价 |
| total_price | double | Y | Y | 总价 |
| status | int | Y | Y | 支付状态 |
| deliver_status | int | Y | Y | 通知状态 |
| product_name | String | Y | Y | 产品名称 |
| product_desc | String | Y | Y | 产品描述 |
| extend_param | String | Y | Y | 扩展字段 |
| success_time | long | Y | Y | 成功时间 |
| create_time | long | Y | Y | 创建时间 |
| sign | String | Y | N | 签名串(用 app_secret 进行签名) |

**返回示例：**


```
{
  "code": 200,
  "message": "",
  "value": {
    "app_id": "100",
    "buy_amount": "100",
    "create_time": "1454036706",
    "cp_orderid": "10560",
    "deliver_status": "0",
    "extend_param": "test",
    "game_server_id": "0",
    "order_id": "1601291105060000024",
    "product_desc": "YuanBao",
    "product_name": "YuanBao",
    "product_per_price": "1.0",
    "sign": "d077092310a5507e944505655104443a",
    "status": "0",
    "success_time": "0",
    "total_price": "1.0",
    "uid": "118"
  }
```

## 2.5 附录

### 2.5.1 错误类型

| 状态码 | 状态描述 |
| --- | --- |
| 110003 | 请求参数错误 |
| 198001 | 游戏不存在 |
| 198002 | 签名错误 |
| 198004 | token 无效 |
| 120015 | 订单不存在 |

### 2.5.3 常见问题

**请求参数签名校验不通过：**

1. 检查是否使用了错误的app_secret,在接入时，常有错用app_key进行签名；
2. 在对参数签名时，检查参数是否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. 签名的参数和请求的参数要一致，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。