机锋 SDK 接入文档

更新日期：2013-12-16 版本号：3.3

目录

1. 版本履历

版本号	时间	更新内容
2.4	2011-12-01	修复 BUG：1.MIUI ROM 兼容性，某些情况下登录界面，无法输入密码。2.某些情况下登录会 FC
2.5	2012-03-07	取消最大支付限额，最小支付金额从 5 机锋券调整到 1 机锋券。 用户 20 分钟内可免登录支付。
2.6	2012-06-15	优化了支付流程，取消短信支付及支付类型。黄色部分为本次修改、更新的内容。
2.7	2012-10-25	增加单点登录，重构代码。
3.0	2012-03-06	增加“免注册登录”功能，增加对 Unity3D 程序调用的支持，重构优化代码，取消了自定义权限。
3.2	2013-09-10	增加了财付通充值渠道,解决了部分界面字体白色的问题。
3.3	2013-12-16	增加了银联充值渠道。

2. 术语与缩写解释

术语	定义	备注
支付 key（appkey）	是应用的唯一标识，每一个应用只能使用一个 appkey。	1．请登录开发者后台申请。 2．需审批后方可开通，时间为 1~2 个工作日。 3．未开通前不影响程序嵌套。
CPID	是开发者自定义的应用推广渠道标识，用于统计一个应用在不同渠道推广带来的用户数等数据。	1．非必须，可不填写 2．要求不超过 10 位，只能包含数字、字母及符号“.”。
密钥（KEY）	密匙（KEY）是由机锋网分配的 16 字节密钥，TEA 采用 16 轮迭代。	当 appkey 通过审核后，密钥可以在“开发者服务平台-机锋服务-申请支付 appkey”查看”。

注意：支付 key(appkey)及对应密匙（KEY）是调用机锋用户中心和订单查询 API 时必须使用的数据，请接入操作前提前申请完毕。

3. 使用方法

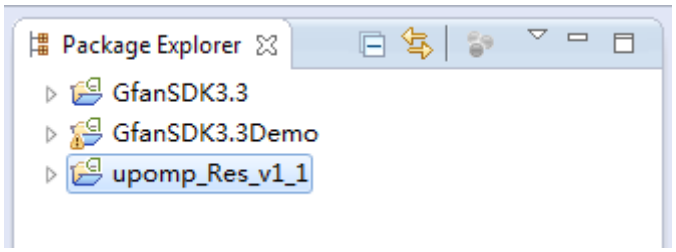
3.1 如何在项目中配置

第一步 导入以下工程：

机锋支付 sdk：GfanSDK3.3 工程

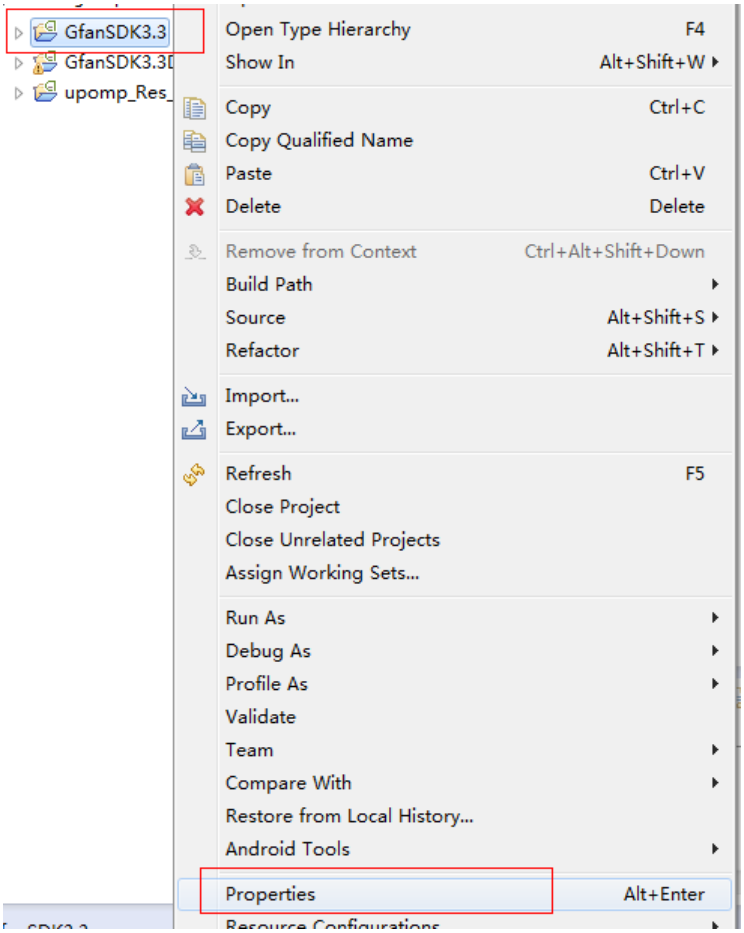
交通银行银联支付工程：upomp_Res_v1_1 工程

机锋支付 sdk3.3Demo：GfanSDK.3Demo 工程

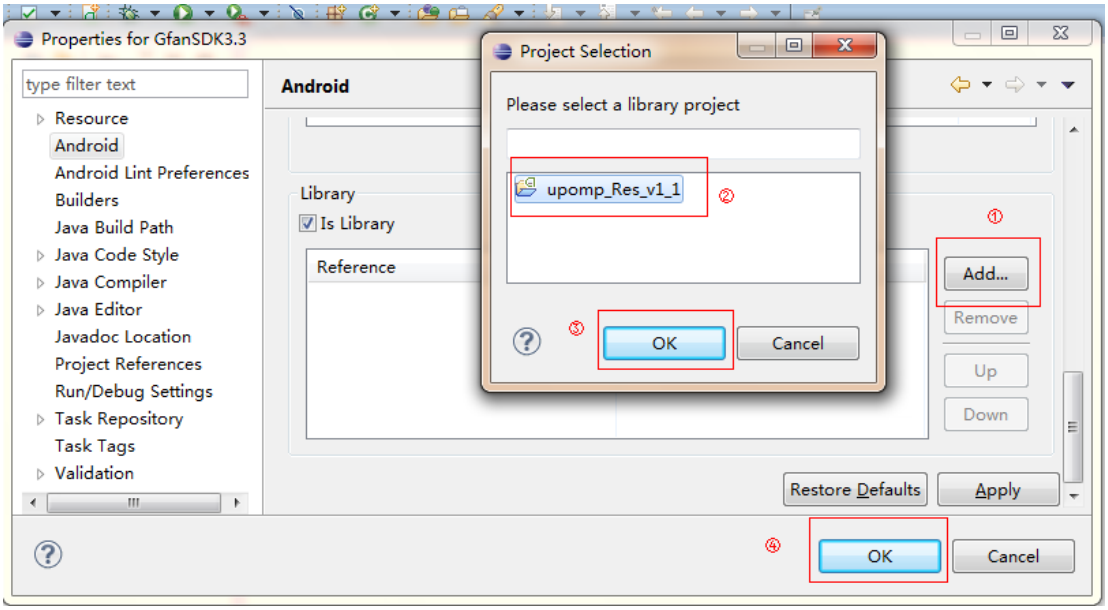


第二步 关联相关工程。

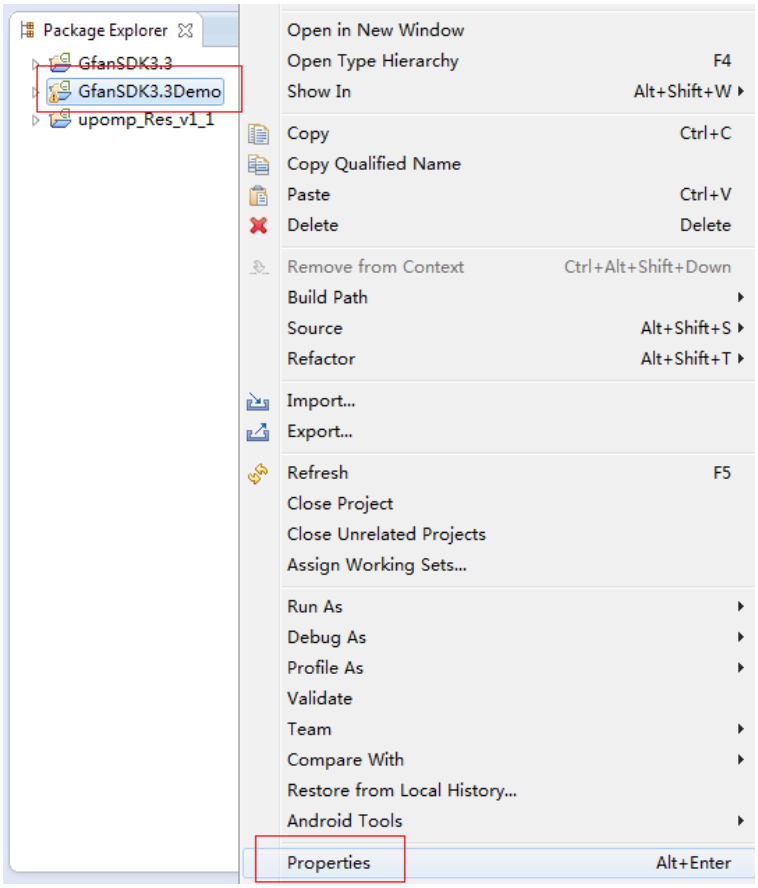
a) 在 Eclipse 中右键点击 GfanSDK3.3 项目后弹出菜单内选择“Properties”。



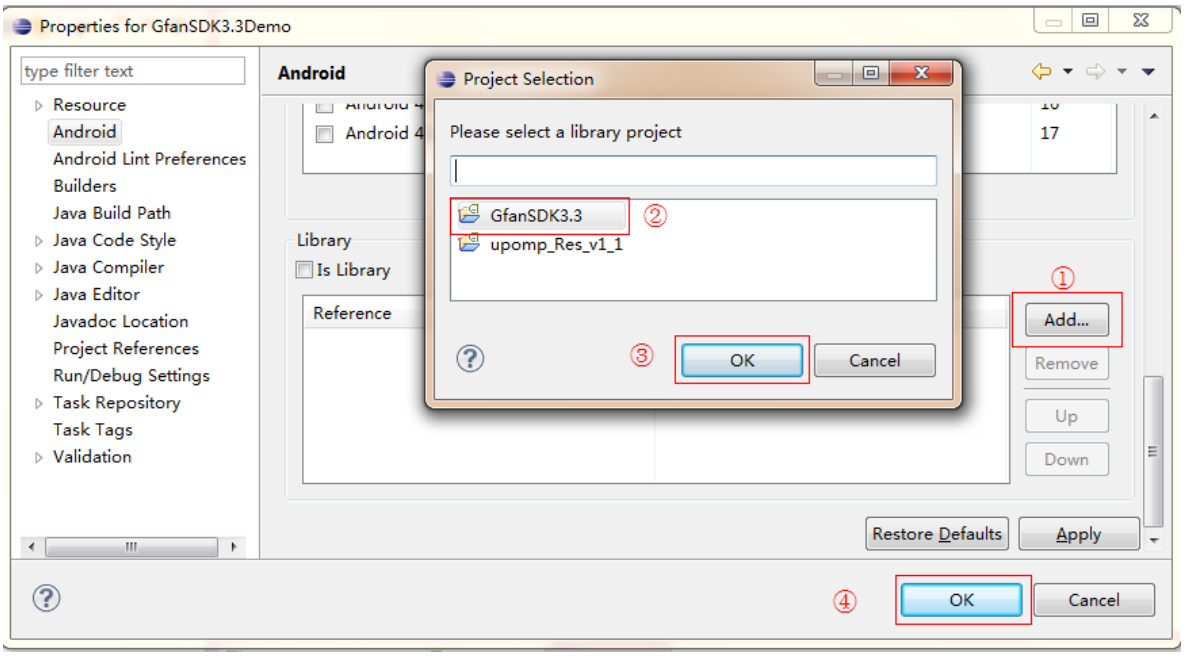
b) 选择 **Android** 选项卡，在 Library 中点击 **Add**，选择 **upomp_Res_v1_1** 工程，点击 ok 进行关联。如没有选项，请确认该工程是否勾选 " □Is Library" 选项。这样 GfanSDK3.3 工程就和银联的工程关联完毕了。



c) 在 Eclipse 中右键点击 GfanSDK3.3Demo 项目后弹出菜单内选择 "Properties"。



d) 选择 **Android** 选项卡，在 Library 中点击 **Add**，选择 **GfanSDK3.3** 工程，点击 ok 进行关联。如没有选项，请确认该工程是否勾选 " □Is Library" 选项。这样 GfanSDK3.3Demo 就和机锋支付 sdk 的工程关联完毕了。



第三步 配置 AndroidManifest.xml。

在项目根目录下打开 **AndroidManifest.xml**。

a) 配置 **AndroidManifest.xml** 中的权限，添加如下内容：

```
<!-- start for gfan sdk -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission
      android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<!-- end for gfan sdk -->
```

b) 配置 **AndroidManifest.xml** 中的常量，在 Application 节点内添加：

```
<meta-data android:name="gfan_pay_appkey"android:value="XXXXXXXX" />
<meta-data android:name="gfan_cpid"android:value="CPID" />
```

gfan_pay_appkey：支付 key(appkey)，登录开发者后台申请。

gfan_cpid :是开发者**自定义**的应用推广渠道标识，要求不超过 10 位，只能包含数字、字母及符号“.”。

c) 配置 **AndroidManifest.xml** 中 **targetSdk**

```
<uses-sdkandroid:targetSdkVersion="4" />
```

注意，此处可以不设置，但是可能在某些界面出现页面显示不正常，但不影响 SDK 的正常使用，亦可设置 4 以上的高版本。

d） 需要在工程的 **AndroidManifest.xml** 文件中引入这些组件的声明：

```
<!-- start for gfan sdk -->
        <activity
            android:name="com.mappn.sdk.uc.activity.LoginActivity"
            android:configChanges="orientation|keyboardHidden"
            android:theme="@style/Transparent" />
        <activity
            android:name="com.mappn.sdk.uc.activity.RegisterActivity"
            android:configChanges="orientation|keyboardHidden"
            android:theme="@style/Transparent" />
        <activity
            android:name="com.mappn.sdk.uc.activity.ChooseAccountActivity"
            android:configChanges="orientation|keyboardHidden"
            android:theme="@style/Transparent" />
        <activity
            android:name="com.mappn.sdk.pay.payment.PaymentsActivity"
            android:configChanges="orientation|keyboardHidden"
            android:theme="@style/Transparent" />
       <activity
            android:name="com.mappn.sdk.pay.chargement.ChargeActivity"
            android:configChanges="orientation|keyboardHidden"
            android:theme="@style/Transparent" />
```

```
      <activity
          android:name="com.mappn.sdk.pay.account.LoginActivity"
          android:configChanges="orientation|keyboardHidden"
          android:theme="@style/Transparent"
          android:windowSoftInputMode="adjustUnspecified" />
  <!-- 免注册登录 -->
       <activity
          android:name="com.mappn.sdk.uc.activity.OnekeyLoignActivity"
          android:configChanges="orientation|keyboardHidden"
          android:theme="@style/Transparent" />
  <!-- 完善用户信息 -->
       <activity
          android:name="com.mappn.sdk.uc.activity.ModfiyActivity"
          android:configChanges="orientation|keyboardHidden"
          android:theme="@style/Transparent" />
       <service android:name="com.mappn.sdk.pay.GfanPayService" />
  <!-- mo9 -->
       <activity
          android:name="com.mokredit.payment.MktPayment"
          android:configChanges="keyboardHidden|orientation"
          android:windowSoftInputMode="adjustResize" />
  <!-- start for net bank -->
       <activity
          android:name="com.unionpay.upomp.lthj.plugin.ui.SplashActivity"
          android:screenOrientation="portrait" >
          <intent-filter>
              <action
android:name="com.unionpay.upomp.lthj.android.plugin.init.test" />
              <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
       </activity>
       <activity

android:name="com.unionpay.upomp.lthj.plugin.ui.IndexActivityGroup"
          android:screenOrientation="portrait" >
           <intent-filter>
              <action
android:name="com.unionpay.upomp.lthj.android.plugin.index.test" />
              <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
       </activity>
```

```
<activity
            android:name="com.unionpay.upomp.lthj.plugin.ui.HomeActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.unionpay.upomp.lthj.plugin.ui.PayActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.unionpay.upomp.lthj.plugin.ui.AccountActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity

android:name="com.unionpay.upomp.lthj.plugin.ui.BankCardInfoActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity

android:name="com.unionpay.upomp.lthj.plugin.ui.SupportCardActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity
         android:name="com.unionpay.upomp.lthj.plugin.ui.UserProtocolActivity"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.unionpay.upomp.lthj.plugin.ui.AboutActivity"
            android:screenOrientation="portrait" >
        </activity>
        <!-- end for net bank -->
```

3.2 支付 SDK 调用

3.2.1 初始化 SDK

第一步 在应用启动的 Activity 内加入初始化代码：

```
GfanPay.getInstance(getApplicationContext()).init();
```

3.2.2 登录及注册调用(单点登录/免注册登录)

单点登录是在同一个设备内，多个应用之间可以共享机锋账号的功能。

如开发者希望使用自己的 UI 界面进行注册、登录，可参考“**机锋【支付 SDK】API**”说明文档，机锋同时提供以 API 方式的用户对接。

登录中的**免注册登录**功能，是“SDK 客户端”根据用户设备信息自动为用户生成一个机锋账号并且登录的过程，如更换手机后，将无法找回登录账号密码，开发者可以强制调用**完善账号**接口。

方法说明：

GfanUCenter.login(Activityactivity, GfanUCCallback gfanUCCallback)

说明：开始登录。

GfanUCenter.regist (Activityactivity, GfanUCCallback gfanUCCallback)

说明：开始注册。

GfanUCenter.logout(Activity activity);

说明：开始注销。

GfanUCenter.*modfiy* (Activityactivity, GfanUCCallback gfanUCCallback)

说明：开始完善账号。

GfanUCenter.isOneKey(Activity activity)

说明：检查用户是否为一键登录用户（游客账号）。

参数：

字段名	字段类型	字段含义
gfanChargeCallback	GfanUCCallback	登录/注册/完善账号回调函数

1) GfanUCCallback 回调接口说明：

```
public interface GfanUCCallback
```

方法：

- ```
  public abstract void onSuccess(User user, intloginType)
  ```

  说明：登录/注册成功时回调函数。

  参数：

字段名	字段类型	字段含义
user	User	用户信息
loginType	int	登录/注册/完善账号类型

- ```
  public abstract void onError(intloginType)
  ```

  说明：登录/注册失败时回调函数。

  参数：

字段名	字段类型	字段含义
loginType	int	登录/注册/完善账号类型

2) 登录/注册/完善账号类型常量：

常量名称	常量含义
GfanUCenter.RETURN_TYPE_REGIST	注册成功
GfanUCenter.RETURN_TYPE_LOGIN	登录成功
GfanUCenter.RETURN_TYPE_MODFIY	完善账号

3) 登录模板：

```
GfanUCenter.login(this, newGfanUCCallback() {
    @Override
    publicvoidonSuccess(User user, intloginType) {
        // 由登录页登录成功
        if (GfanUCenter.RETURN_TYPE_LOGIN == loginType) {
            // TODO登录成功处理
            // 由登录页注册成功
        } else {
            // TODO注册成功处理
        }
    }

    @Override
    publicvoidonError(intloginType) {
        // TODO失败处理
    }
});
```

4) 注册模板

```
GfanUCenter.regist(this, newGfanUCCallback() {
    @Override
    publicvoidonSuccess(User user, intloginType) {
        // 由注册页注册成功
    }

    @Override
    publicvoidonError(intloginType) {
        // TODO失败处理
    }
});
```

5) 注销模板

会在第二次开启时自动登录首次登录的账号，开发者可以调用注销接口，提供切换账号的功能。

```
GfanUCenter.logout(this);
```

6) **完善账号**模板（仅针对“**免注册登录**”创建的账号）

```
GfanUCenter.modfiy(this, new GfanUCCallback() {
            @Override
            public void onSuccess(User user, int returnType) {
                if (GfanUCenter.RETURN_TYPE_MODFIY == returnType) {
                    //TODO  完善账号成功
                }
            }

            @Override
            public void onError(int returnType) {
                if (GfanUCenter.RETURN_TYPE_MODFIY == returnType) {
                    //TODO  完善账号失败
                }
            }
});
```

7) 是否为一键登录用户模版（仅针对“免注册登录”创建的账号）

```
ToastUtil.showLong(getApplicationContext(),
(GfanUCenter.isOneKey(this)?"是":"不是")+"游客账号");
```

3.2.3 调用充值接口

开发者可以通过以下的方法来实现**现金充值成机锋券**的操作，操作成功后，返回充值用户的用户名及该用户的机锋券余额，开发者也可以不调用该接口，支付接口在机锋券余额不足时会自动跳转充值接口。

1) 充值方法说明：

GfanPay 类中公有方法：

```
public void charge(GfanChargeCallback gfanChargeCallback)
```

说明：开始一次充值操作。

参数：

字段名	字段类型	字段含义
gfanChargeCallback	GfanChargeCallback	充值回调函数

2) GfanChargeCallback 回调接口说明：

```
public interface GfanChargeCallback
```

方法：

- ```
  public abstract void onSuccess(User user)
  ```

  说明：充值成功时回调函数。

  参数：

字段名	字段类型	字段含义
user	User	用户信息

- ```
  public abstract void onError(User user)
  ```

  说明：充值失败时回调函数。

  注意：若用户未登录，则user为null，取值前要判断是否为空

  参数：

字段名	字段类型	字段含义
user	User	用户信息

3) 充值模板：

```
GfanPay.getInstance(getApplicationContext()).charge(
    newGfanChargeCallback() {
        @Override
        public void onSuccess(User user) {
            // TODO充值成功处理
        }
        @Override
        public void onError(User user) {
            // TODO充值失败处理
        }
    });
}
```

3.2.4 调用支付（消费）接口

1) 消费方法说明：

GfanPay 类中公有方法：

`pay(Order order, GfanPayCallback gfanPayCallback)`

说明：传入订单信息，开始一次消费。

参数：(字段内容参看 “附：类说明”)

字段名	字段类型	字段含义
order	Order	订单信息
gfanPayCallback	GfanPayCallback	消费回调函数

2) GfanPayCallback 回调接口说明：

`public interface GfanPayCallback`

方法：

- `public abstract void onSuccess(User user, Order order)`

  说明：消费成功时的回调函数。

参数（字段内容参看“附：类说明”）

字段名	字段类型	字段含义
user	User	用户信息
order	Order	订单信息

- `public abstract void onError(User user)`

  说明：消费失败时的回调函数。

  注意：若用户未登录，则user为null，取值前要判断是否为空

  参数：

字段名	字段类型	字段含义
user	User	用户信息

3) 消费模板：

```
Order order = new Order("订单名称", "订单描述",订单金额, "订单号");
GfanPay.getInstance(getApplicationContext()).pay(order,
    newGfanPayCallback() {
        @Override
        Public void onSuccess(User user, Order order) {
            // TODO消费成功处理
        }
        @Override
        Public void onError(User user) {
            // TODO消费失败处理
        }
    });
}
```

说明：机锋 SDK 提供“主动通知”、“订单查询”两种方式获取消费结果，具体见“4、获取消费结果”。

3.2.5 漏单查询接口调用

只有在有漏单存在的时候此方法才会返回数据，所谓漏单是指由于 Service 的问题导致服务器显示成功而客户端显示未成功。此时调用此方法可获取订单信息。

漏单查询说明：

GfanPay 公有方法：

```
Public void confirmPay (GfanConfirmOrderCallback gfanConfirmOrderCallback)
```

参数：

字段名	字段类型	字段含义
gfanConfirmOrderCallback	GfanConfirmOrderCallback	漏单查询回调

GfanConfirmOrderCallback 回调接口说明：

- `public abstract void onExist(Order order)`

说明：有漏单数据产生时回调函数

参数：

字段名	字段类型	字段含义
order	Order	订单信息

- `public abstract void onNotExist()`

说明：此接口为不存在漏单信息时所回调方法。

1）漏单查询模板：

```
GfanPay.getInstance(getApplicationContext()).confirmPay(new
GfanConfirmOrderCallback() {
            @Override
            public void onExist(Order order) {
                // TODO 存在漏单
            }
            @Override
            public void onNotExist() {
                // TODO 不存在漏单
            }
        });
```

3.2.6 运营数据分渠道统计

用户可以通过调用指定的事件，统计应用的启动、用户登录、用户注册、用户新建角色等数据。

查看方式“机锋开发者服务平台-机锋 SDK 收入-渠道时段数据/渠道累计数据”

1) 方法说明

GfanPay.*submitLogin*(Context context);

说明：在用户“登录游戏成功”后，调用此方法

GfanPay.*submitRegist*(Context context);

说明：在用户“注册游戏账号成功”后，调用此方法

GfanPay.*submitNewRole*(Context context);

说明：在用户“新建游戏角色成功”后，调用此方法

2) 登录示例

```
GfanPay.submitLogin(getApplicationContext());
```

3) 注册示例

```
GfanPay.submitRegist(getApplicationContext());
```

4) 新建角色示例

```
GfanPay.submitNewRole(getApplicationContext());
```

3.2.7 附：类说明

订单类

```
public class Order
```

构造方法：

- `Order(String payName, String payDesc, int money, String orderId)`

说明：Order 类构造函数，生成一个订单对象，orderId 由开发者传入。

注：每次传入的订单号必须唯一，不可重复。

推荐算法：MD5(username + appkey + payname +* timeMillis + ip)

参数：

字段名	字段类型	字段含义
payName	String	消费名称
payDesc	String	消费描述
money	int	消费机锋券数额，1 机锋券=0.1 元人民币
orderId	String	订单号，≤64 位

- `Order(String payName, String payDesc, int money)`

说明：Order 类构造函数，生成一个订单对象，orderId 由 SDK 自动生成。

参数：

字段名	字段类型	字段含义
payName	String	消费名称
payDesc	String	消费描述
money	int	消费金额

公有方法：

- `public String getPayName()`

  说明：返回传入的消费名称

- `public String getPayDesc()`

  说明：返回传入的消费描述

- `public intgetMoney()`

  说明：返回传入的消费金额

- `public String getOrderID()`

  说明：返回订单号

- `public String getNumber()`

  说明：返回消费成功后生成的流水号

用户类

`public class User`

公有方法：

- `public String getUserName()`

  说明：返回用户名

- `public long getUid()`

  说明：返回uid

- `public String getToken()`

  说明：返回登录token，用于向服务器验证用户身份是否合法

3.3 编程开发

1) 在混淆自己 APK 包的时候请不要将机锋 SDK 和银联 sdk 的 jar 包一起混淆，因为里面有些自定义 UI 控件，若被混淆后会因为无法找到相关类而抛异常。您可以在用 ant 构建混淆包的 build.xml 里面对应位置或者在 proguard.cfg 里加入以下代码：

```
-keep class **.R$* {*;}
-keep class com.mappn.sdk.** {*;}
-keep class com.alipay.** {*;}
-keep class com.mokredit.payment.MktLineLyt$*{<methods>;}
-keepclassmembers class com.mokredit.payment.MktLineLyt$*{*;}
```

若是仍然混淆出错，则加入以下代码：

```
-dontwarn **HoneycombMR2
-dontwarn **CompatICS
-dontwarn **Honeycomb
-dontwarn **CompatIcs*
-dontwarn **CompatFroyo
-dontwarn **CompatGingerbread
-dontwarn android.support.v4.**
-dontwarn **CompatHoneycomb
-dontwarn **CompatHoneycombMR2
-dontwarn **CompatCreatorHoneycombMR2
-keep class android.support.v4.** { *; }
-keep public class * extends android.support.v4.**
```

-keep public class * extends android.app.Fragment

-keepattributes InnerClasses

-dontwarn com.tendcloud.tenddata.*

-dontwarn com.tendcloud.tenddata.ly.*

-keep class com.tendcloud.tenddata.**{*;}

-keep class com.lthj.unipay.plugin.**{*;}

-keep class com.unionpay.upomp.lthj.**{*;}

2）在程序中多次声明**"gfan_pay_appkey"**，否则无法正常使用。

3）如需进行测试，可向机锋商务人员索取测试账号，帐号内有机锋券，可用于消费测试。建议开发者在测试支付 SDK 时将支付金额设置成 **1 个机锋券**，即 0.1 元，以减少测试成本。如果测试账号的机锋券余额不足，请联系**机锋商务人员**再添加。

4. 获取消费结果

详情请参考“**机锋【支付查询】API**”，有以下两种方式：

主动通知：

在申请支付 key(appkey)时，可以一并提交主动通知回调地址，当一笔订单**消费成功**后，机锋服务器会回调该地址，通知该订单已经成功消费。

到机锋服务器查询：

开发者可以根据获取到的 orderId 到机锋服务器查询此订单是否已经成功支付。

5. 查询收入及相关数据报表

开发者可使用自己的开发者帐号登录开发者后台，选择左侧导航栏“收入&结算”下的“机锋 SDK 收入”，通过此后台可以查询到以下数据：

用户支付的订单号（orderID）、CPID、时间、金额

分渠道（对应 CPID）的收入、启动、登录、注册、创建角色数据

具体分成比例及结算方式等，请联系机锋商务人员，并与机锋网签订正式合同。

6. 常见问题和解答

6.1 使用机锋 SDK，是否必须联网？

是，因为机锋 SDK 为联网支付模式，所以必须开通联网权限，并在支付时产生非常小的网络流量。

6.2 消费记录是否会随着用户删除应用而消失？

6.2.1 如果开发者在单机应用中使用机锋支付，因支付点记录在手机中，一旦用户删除应用，下次再安装同一应用，也必须重新支付才可使用，建议开发者在应用帮助中增加提示：用户从手机中删除本应用，则视同为放弃了已经付款购买的功能，再次安装并使用同一应用，仍需再次支付费用。

6.2.2 对于联网应用，开发者可将支付行为记录到自己的数据库中，以保证产品的支付记录不会被删除。

6.3 如果我的应用已经发布并使用了自己的帐号体系,是否可以升级后直接使用机锋帐号体系？

如果应用将原来的支付模式修改为机锋支付，升级没有问题。但如果过去的版本使用的是不同帐号体系的，不建议直接升级。因为这种升级会导致用户无法用老帐号登录新更新的应用。所以，目前建议是如修改了帐号体系，则用不同的包名来区分，这样也就不存在交叉更新的问题了。

6.4 用户在支付中遇到问题或支付纠纷如何处理？

因使用机锋 SDK 而出现的充值不成功等问题，由机锋网客服人员或者对应的商务联系人负责解决。（联系人请参见文档最后的联系人列表。）

6.5 为什么会出现“找不到 activity:activity not found exception”的情况？

没有将 activity 定义在 application 结点下，meta-data 也应一并放在 application 结点下。

6.6 为什么出现“该应用的支付 KEY 无效，不能成功支付”？

可能的原因有：

（1）appkey 未审核通过。

（2）meta-data 放在了 application 结点外。

6.7 为什么账户余额充足且联网正常，还会出现“支付不成功，请确定您的账户当中的余额充足并网络连接正常”的提示？

6.7.1 可能是因为在 PaymentInfo 的构造函数中添加了非法的 orderId，要求 orderId 是不大于 64 位的数字或字母组合而成的字符串，更不可含有“.”之类的字符。另外，要求 orderId 是不可重复的，如果重复也可能造成支付不成功。

6.7.1 还有可能是由于 PaymentInfo 的构造函数中添加了非法的支付点名称，要求支付点名称由数字或英文字母或中文组成，不可含有尖括号(“<”、“>”)之类的字符。

6.8 混淆后会导致无法支付。

最后请检查下是否将支付 SDK 进行了混淆，切勿将 SDK 混淆。

6.9 接入客户端过程中可能会遇到的问题。

6.9.1 导包错误：

导包出现找不到资源等错误，在确认关联无误的情况下，请查看 properties/Java Build Path/Order and Export 目录下的 Android private libraries 是否勾选上了，ADT 版本 22 之后如果不勾选将会报错。或查看导包进来的目录下是否有中文字符。

6.9.2 登录时报错误码 500，免注册登录报错误码 214：

查看 AndroidManifest.xml 文件中，支付 key（appkey）是否填写正确，填写成密钥会报这个错误。

6.9.3 不需要免注册登录这一功能：

可以在 sdk 的资源文件中，将免注册登录的 button 隐藏。

6.9.4 支付时报错误码 423：

订单号过长会导致这个错误，新版本订单号要求≤64 位。

6.9.5 支付前会闪一下屏幕：

支付前会有登录验证，因此会有登陆界面闪过。

6.9.6 支付时提示:该应用要求支付的机锋券超过了限制，不能成功支付，错误码：0：

支付金额小于等于 0 或者数字不合理的情况下会出现支付超过限制提示。

6.9.7 测试支付时，第一次支付成功，第二次支付的时候，会报错误码 220：

订单号重复会报错误码 220，为了对账的准确性，支付订单号要求不能重复使用。

开发者服务支持

机锋客服邮箱	support@gfan.com
游戏商务邮箱	Gamesales@gfan.com
机锋 sdk 及 api 技术服务支持 qq 群	151059121、245418897