**短信代码切换SDK接入开发规范**

**目录**

一．简介 3

二．SDK使用方法 3

Step 1. 集成SDK资源文件 3

Step 2. 修改AndroidManifest文件 3

Step 3. 加载动态库 4

Step 4 .初始化支付SDK 5

Step 5. 使用支付功能 5

Step6程序退出，销毁对象，避免内存泄露 6

### 一．简介

短信代码切换SDK是一套多渠道短信支付接入管理控制平台，目前集成的支付SDK包括游戏基地、移动MM、联动优势和MDO。

SDK实现了“一个应用，多种渠道”支付的功能，游戏开发者只需要调用一个接口即可完成多渠道支付，不需要关注SDK配置管理等细节。

### 二．SDK使用方法

请参考下面的Step 1到Step 6使用SDK：

##### Step 1. 集成SDK资源文件

1. 分别拷贝assets、libs、res资源至应用对应目录下；
2. 拷贝runtime目录至应用根目录下。

##### Step 2. 修改AndroidManifest文件

1. 在AndroidManifest.xml中添加用户权限：

<uses-permissionandroid:name=*"android.permission.INTERNET"*/>

<uses-permissionandroid:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>

<uses-permissionandroid:name=*"android.permission.ACCESS\_WIFI\_STATE"*/>

<uses-permissionandroid:name=*"android.permission.ACCESS\_COARSE\_LOCATION"*/>

<uses-permissionandroid:name=*"android.permission.ACCESS\_FINE\_LOCATION"*/>

<uses-permissionandroid:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"*/>

<uses-permissionandroid:name=*"android.permission.SEND\_SMS"*/>

<uses-permissionandroid:name=*"android.permission.READ\_PHONE\_STATE"*/>

<uses-permissionandroid:name=*"android.permission.READ\_CONTACTS"*/>

<uses-permissionandroid:name=*"android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"*/>

<uses-permissionandroid:name=*"android.permission.GET\_TASKS"*/>

1. 在AndroidManifest.xml注册service和activity：

<serviceandroid:name=*"com.ckmobilling.CkService"*

android:exported=*"true"*

android:process=*":remote"*>

<intent-filter>

<actionandroid:name=*"com.ckmobilling.ICkService"*/>

</intent-filter>

</service>

<activityandroid:name=*"com.ckmobilling.MdoPayActivity"*

android:configChanges=*"orientation|keyboardHidden"*

android:excludeFromRecents=*"true"*>

</activity>

<activityandroid:name=*"com.umpay.huafubao.ui.BillingActivity"*

android:configChanges=*"orientation|keyboardHidden"*

android:excludeFromRecents=*"true"*>

</activity>

<serviceandroid:name=*"com.umpay.huafubao.service.AppUpgradeService"*/>

##### Step 3. 加载动态库

1. 在应用的Application派生类的onCreate方法里调用如下语句：

**publicclass** App **extends** Application {

@Override

**publicvoid**onCreate() {

**super**.onCreate();

CkSdkApi.*getInstance*().loadLibrary(**this**);

}

}

2. 应用自定义Application后,同步修改AndroidManifest.xml文件.

<application

android:allowBackup=*"true"*

android:name=*".App" // 修改此处*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*>

… … …

</application>

##### Step 4 .初始化支付SDK

在应用的Activity派生类的onCreate方法里调用如下语句：

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

CkSdkApi.*getInstance*().initialize(**this**); //注意this 必须是个 Activity的上下文

… … …

}

##### Step 5. 使用支付功能

调用CkSdkApi类的doPayment函数进行支付操作。

**public void** doPayment(Context context, String ckPayCode, PaymentCallback callback);

参数说明：

context 所在Activity派生类上下文。该上下文必须和初始化上下文保持一致。

ckPayCode触控计费代码，服务器会根据具体支付sdk生成对应的支付代码或订单。

callback 支付结果反馈，提供开发者支付成功、失败和取消反馈信息。开发者需要new一个继承此接口类的对象来接收购买结果回调通知。

示例代码：

CkSdkApi.*getInstance*().doPayment(MainActivity.**this**, "0007", **new** PaymentCallback() {

**public void** paySuccess(PaymentResult result) {

// 支付成功

}

**public void** payFailed(PaymentResult result) {

// 支付失败

}

**public void** payCanceled(PaymentResult result) {

// 用户取消的支付

}

});

回调参数中的PaymentResult为结果描述类，结构如下：

{

public String payCode; // 道具编号，与doPayment方法中传入的触控PayCode一致，支付完成后，开发者如果需要再一次核对用户所购买内容，可以通过这项数据来检测。

}

在PaymentCallback函数中可以通过调用result.payCode获取payCode数据。

##### Step6程序退出，销毁对象，避免内存泄露

在退出过程中,添加如下代码对部分对象进行销毁.

@Override

protected void onDestroy() {

super.onDestroy();

//此处为添加代码

CkSdkApi.getInstance().onDestroy();

//添加结束

}