

联想网游 SDK 接入指导

V2.2

联想 ECS

2016 年 5 月

目 录

1. 手机网游联运接入流程	4
2. 联想网游 SDK 介绍	9
3. 集成联想网游 SDK	10
4. 集成联想网游 SDK 代码	16
5. 联想网游 SDK 详细说明	17
6. 混淆配置	23
7. 接入注意事项	28
8. 服务端验证 Lenovo ID Token(ST)	28
9. 服务端支付通知接口	32

修 改 历 史

版本	日期	备注	修改人
0.1	2016/3/10	最初版本	张伶俐
0.2	2016/3/29	增加 manifest 中 token 定义 (p15)	张伶俐
0.3	2016/5/26	1. 悬浮窗增加 V 币按钮 2. 解决禁止读取账户权限导致支付失败无提示问题 3. 增加倒数输错密码提醒 4. 增加可选接口:查询是否成年, 实名注册 5. 调整登录成功广告为图文, V 币兑换, 激活码领取三种模式 6. 调整退出广告为游戏推荐模式 7. 更新默认闪屏/登录成功广告图片	张伶俐

1. 手机网游联运接入流程

联想手机游戏（网游）SDK 是联想生态系统和云服务业务集团(ECS)推出的面向所有 Android 手机游戏，整合了联想帐号（联想 ID）统一认证、游戏应用内计费 and 支付等多平台支撑能力，接入简单快捷，游戏应用只需集成本 SDK 即可。

目前采用游戏联运的方式接入游戏，合作分为以下几个阶段：

- SDK 接入
- 游戏上线

1.1 SDK 接入

确定游戏联运合作后，需要签署相关游戏联运协议。同时启动游戏 SDK 对接。

1. 游戏 CP 需要注册联想开放平台开发者帐号（访问

<http://open.lenovo.com/developer/lenovoReg.jspx>），并通过开发者审核。

2. 在联想开放平台完成基本数据填写，申请应用 Open AppID，如下图。

- 在开发者管理后台，选择-创建应用-选择游戏（网游支付合作）



- 填写应用名称，包名，**注意包名在联想开放平台全局唯一，不可重复，保存后不可修改。**

完善信息

① 创建 ② 完善信息 ③ 上架

OPENAPPID: 1606220349117.app.ln

联运信息

应用包名

open.test.game.lenovo

应用名

测试

- 获取应用 Open AppID，如下图。（可在完善信息界面查看包名与 openappid，Open AppID 请按照本文档描述，配置到游戏应用的 AndroidManifest.xml 文件相应位置）

管理中心

1.创建应用 2.推广 3.数据统计 4.资料库 5.我的资料

搜索

待提交 在售中：4 测试中：3 + 创建应用

 **test**
版本号：
待提交

完善信息
删除

[计费设置](#) | [广告设置](#)

3. 点击计费设置，此时如果商户资料还未通过审核，会先进入商户资料填写页面（如下图）完成商户资料填写并完成商户审核。然后再进入联想开放平台，按屏幕引导，在移动应用内计费系统完善计费数据，完成游戏应用内计费点配置。

4. 商户资料完善

添加商户

邮箱:

类 型: ☐ 个人开发者 ☒ 企业开发者

* 商户名称:

* 收款人姓名:

* 开户银行:

* 银行卡卡号:

* 手机联系方式:

联系方式:

公司名称:

公司地址:

公司网址:

地域:

● 已添加应用列表

应用中心

应用列表

账户管理

基本信息

交易管理

交易流水

移动应用内计费系统

您好, vpay01@163.com [退出登录](#)

应用列表

应用名称:

应用名称	应用代码	状态	商品名称	商品编码
10.22测试创建=修改	1410222000978.app.in	待完善		
vpaytest2q1...	1410141080579.app.in	正常		
时空猎人示例g1111	1410110147899.app.in	待完善		
刀塔传奇示例	1410090774122.app.in	正常	黄石	276
			绿石	277
			青石	278
			蓝石	279
			宝石	280
			晶石	281
			金刚石	294
vpaytest1	1410101286360.app.in	待完善		

● 应用信息详情

应用中心

应用列表

账户管理

基本信息

交易管理

交易流水

应用详情 神魔示例

切换应用

应用代码: 1410090365246.app.in

应用名称: 神魔示例 (不可修改,请前往乐商店修改)

支付密钥: MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBANiMnFkBpx9
ZOU5i6a/0RiVlanSO
nRroG6KtcCHzLFbSrfAJRi66StbJ2bzU1ZbxjdZBr1LuaJHZmnSxhGgioBmIm5gti
7C-380

(请双击选中框内全部内容进行复制)

支付方式: + 添加支付方式

出现顺序	支付方式	折扣率	操作
1	V币	100%	
2	支付宝	100%	-
3	财付通	100%	-
4	充值卡	100%	-
5	游戏点卡	100%	-
6	银行卡	100%	-

应用描述: 测试

(不可修改,请前往乐商店修改)

交易结果通知: ☒ 同步 ☐ 不同步

接收地址: http://uss.test.lenovomm.cn/cp/cborder

通知接口测试

● 添加计费商品

商品列表 + 增加商品

商品信息	价格	操作
青玉	1.0 元/次 按次计费	-
墨玉	3.0 元/次 按次计费	-

● 查看商品编码 (waresid)

应用中心

应用列表

账户管理

基本信息

交易管理

交易流水

应用列表

应用名称:

查询

应用名称	应用代码	状态	商品名称	商品编码
抽奖示例	1410090365246.app.in	正常	青玉	263
			墨玉	264
			紫玉	265
			红玉	266
			黑玉	267
			和田玉	275
			绝世美玉	291

- 计费数据配置需特别注意事项，见下图红色框标识

交易结果通知: ☒ 同步 ☐ 不同步

接收地址:

建议点击“通知接口测试”：系统会发送模拟交易结果数据到接收地址，用以验证商户的通知接口是否可用

网游在创建应用信息时请务必打开此开关并填写交易信息同步地址。如不确定可在联系技术人员后取得。若不打开则交易信息不予同步。

- 完成游戏计费信息配置后，下载 SDK，并按本文档描述和 Sample 的指引，完成 SDK 接入和游戏的自测

- 在 SDK 接入和自测过程中，可随时登录联想开放平台，通过管理中心 > 待提交 > 计费设置，查看 Open AppID 和支付密钥 appkey 以及进行计费点配置管理

管理中心

1.创建应用



2.推广



3.数据统计

4.资料库

5.我的资料

搜索



消息

待提交



在售中：4



测试中：3

+ 创建应用



test

待提交

版本号：

完善信息

删除

计费设置 | 广告设置

6. 将集成 SDK 的游戏安装包提交给联想测试验证

联想联运游戏 SDK 接入测试联系人

	姓 名	QQ	邮 箱
测试联系人	王崇	1455809322	wangchong3@lenovo.com
版本发布/问题反馈	边玉龙	2426658946	bianyl3@lenovo.com
开发技术支持	胡雪婵	642062148	huxc4@lenovo.com

7. 验证通过，在联想开放平台开发者管理后台自助上传应用，并提交申请发布

管理中心

1.创建应用

2.推广

3.数据统计

4.资料库

5.我的资料

搜索

消息

待提交

在售中：4

测试中：3

+ 创建应用



test

待提交

版本号：

完善信息

删除

计费设置 | 广告设置

1.2 游戏上线

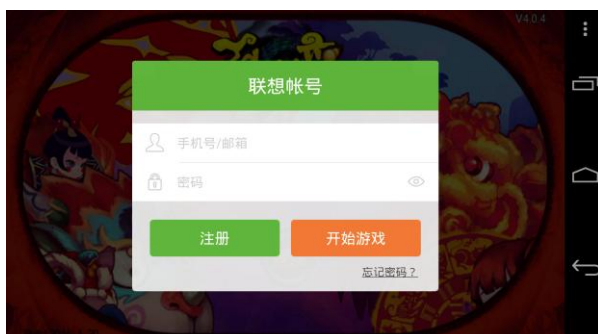
游戏审核通过上线推广。

2. 联想网游 SDK 介绍

联想网游 SDK 是联想为联运的网游提供的接入 SDK。联想网游 SDK 的功能包括：

1. 帮助用户进行自动登录或手动登录
2. 登录完成进入游戏后，显示联想网游工具栏（注：网游工具栏只有在对应的游戏具有攻略、礼包、论坛板块 3 项中至少 1 项的时候，才会出现。否则工具栏自动隐藏）
3. 需要支付时，启动联想收银台进行支付

下图是联想网游 SDK 的运行过程中的重要步骤。



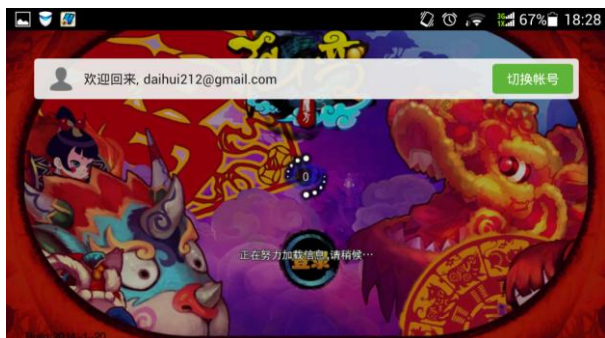
手动登录



登录进行中



获取 ST 进行中



登录成功



显示游戏工具条



打开联想收银台

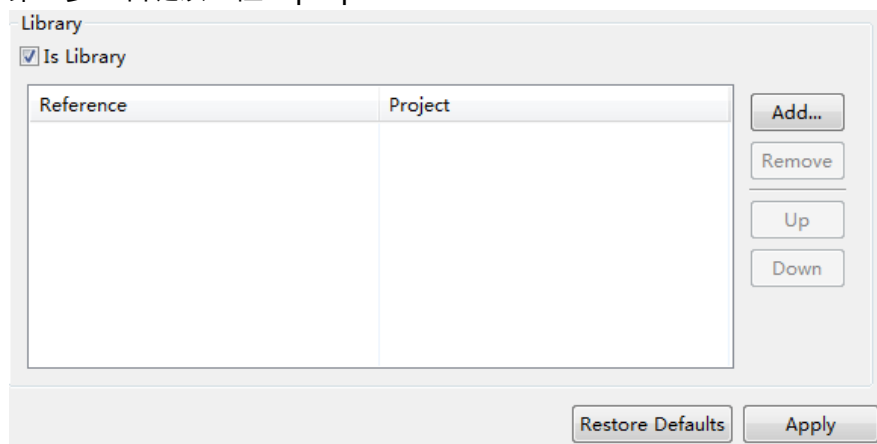
3. 集成联想网游 SDK SDK 压缩包组成

- |---DOC // 联想网游 SDK 文档存放目录, 包含《联想游戏 SDK 快速指导(应用)-vx.x》
- |---Demo_SRC //联想网游 SDK 示例 demo 应用源码目录
- |---SDK //联想网游 SDK 资源工程存放目录
- |---Demo_Sample //联想网游 SDK 示例 APK 以及游戏服务端计费功能相关示例代码存放目录
 - |--- 计费服务端示例代码 //游戏服务端计费功能相关示例代码
 - |--- Demo.apk // 联想网游 SDK 示例 APK
- |--- 游戏图标角标 //联运游戏桌面图标角标 Logo 放置说明和角标资源文件

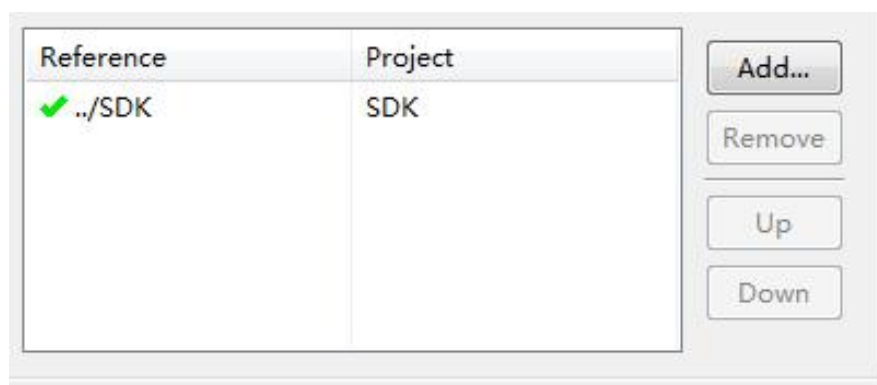
3.2 集成资源工程

第一步：将“SDK”目录导入到 eclipse 工程中

第二步：右键该工程->properties->Android



第三步：右键自己的工程->properties->Android，点击 add 添加即可



3.3 配置游戏 AndroidManifest.xml 文件

对 游戏 AndroidManifest.xml 配置以联想提供的 gamesdk_sample 下的 AndroidManifest.xml 为准，以下供参考！

3.3.1 添加 meta-data, activity, receiver 和 service 到游戏

AndroidManifest.xml 中

```
<!--Lenovo game app id -->
    <meta-data
        android:name="Lenovo.open.appid"
        android:value="在此处填入申请的Lenovo open AppID" />
    <meta-data
        android:name="Lenovo.gamesdk.new"
        android:value="2.6.3" />

<!-- 游戏接入时请直接复制此项内容-->
```

```

        <meta-data android:name="lenovo:applicationToken"
android:value="WQMF9HG8J5WZ"/>

        <meta-data android:name="lenovo:channel" android:value=" 在此处填入申请的Lenovo
open AppID "/>

<!--Lenovo game 登录 -->
    <!-- 配置联想闪屏欢迎页WelcomeActivity，必须配置android:screenOrientation属性来
指定闪屏界面为横屏或竖屏，默认横屏 -->
    <!--
        <activity
            android:name="com.Lenovo.Lsf.gamesdk.ui.WelcomeActivity"
            android:configChanges="orientation|keyboardHidden"
            android:screenOrientation="landscape"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
            android:windowSoftInputMode="stateHidden" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>-->

    <!-- 登录成功欢迎条界面，默认横屏，通过screenOrientation可以进行修改 -->
    <activity
        android:name="com.Lenovo.Lsf.Lenovoid.ui.KeyLoginActivity"
        android:configChanges="orientation|keyboardHidden"
        android:screenOrientation="landscape"
        android:theme="@android:style/Theme.Translucent"
        android:windowSoftInputMode="stateHidden" >
    </activity>
    <!-- 登录成功欢迎条界面结束 -->

    <!-- 登录过程界面，默认横屏，通过screenOrientation可以进行修改，注意必须 10 个
activity同时修改,最好指定全部横屏或全部竖屏，否则在部分机器（z1）上会导致工具条重叠 -->
    <activity
        android:name="com.Lenovo.Lsf.Lenovoid.ui.PsLoginActivity"
        android:configChanges="orientation|keyboardHidden"
        android:screenOrientation="landscape"
        android:label="@string/com_lenovo_lsf_ui_name"
        android:launchMode="singleTask"
        android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog"
        android:windowSoftInputMode="stateHidden" >

        <intent-filter>

```

```

        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.PsLoginCommonActivity"
    android:configChanges="orientation|keyboard|keyboardHidden"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog"
    android:launchMode="singleTask"
    android:windowSoftInputMode="stateVisible" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.StartLoginingGameActivity"
    android:configChanges="keyboardHidden|orientation|screenSize|navigation"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.StartGameErrorActivity"
    android:configChanges="keyboardHidden|orientation|screenSize|navigation"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.FindPasswordActivity"
    android:configChanges="orientation|keyboardHidden"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.RegistByPhoneActivity"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.RegistByPhoneConfirmActivity"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_Lsf_Translucent_NoTitle_Dialog" >
</activity>
<activity
    android:name="com.Lenovo.Lsf.Lenovoid.ui.FindPasswordConfirmActivity"
    android:configChanges="orientation|keyboardHidden"

```

```

        android:screenOrientation="landscape"
        android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog" >
    </activity>
    <activity
        android:name="com.Lenovo.Lsf.Lenovoid.ui.WebViewActivity"
        android:configChanges="orientation|keyboardHidden"
        android:excludeFromRecents="true"
        android:label=""
        android:screenOrientation="landscape "
        android:theme="@style/com_lenovo_lsf_ui" >
    </activity>
    <activity
        android:name="com.lenovo.pop.ui.FlexibleActivity"

android:configChanges="keyboardHidden|orientation|screenSize|navigation"
        android:screenOrientation="landscape"
        android:launchMode="singleTop"
        android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog" >
    </activity>        <activity
        android:name="com.Lenovo.pop.ui.QuiteActivity"

android:configChanges="keyboardHidden|orientation|screenSize|navigation"
        android:screenOrientation="landscape"
        android:launchMode="singleTop"
        android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog" >
    </activity>
    <activity
        android:name="com.lenovo.lsf.lenovoid.ui.QuitActivity"

android:configChanges="keyboardHidden|orientation|screenSize|navigation"
        android:screenOrientation="landscape"
        android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog" >
    </activity>
        <activity android:name="com.lenovo.lsf.lenovoid.ui.RegistRealAuthActivity"
android:configchanges="keyboardHidden|orientation"
android:screenorientation="landscape"
android:theme="@style/com_lenovo_lsf_Translucent_NoTitle_Dialog">
    </activity>
    <!-- 登录过程界面结束 -->

    <service
        android:name="com.Lenovo.Lsf.Lenovoid.toolbar.AppCheckService" >
    </service>

    <receiver        android:name="com.lenovo.lsf.gamesdk.receiver.GameSdkReceiver"

```

```

android:permission="com.lenovo.lsf.device.permission.MESSAGE">
    <intent-filter>
        <action
android:name="com.lenovo.lsf.gamesdk.receiver.GameSdkReceiver"></action>
        <action android:name="在此处填入申请的 lenovo open AppID"></action>
        <category android:name="在此处填入 Package Name" />
    </intent-filter>
</receiver>

    <receiver
android:name="com.lenovo.lsf.gamesdk.receiver.GameSdkAndroidLReceiver">
        <intent-filter>
            <action android:name="com.lenovo.lsf.device.ANDROID_L_MSG"></action>
            <category android:name="在此处填入 Package Name" />
        </intent-filter>
    </receiver>

    <receiver
        android:name="com.lenovo.lsf.push.receiver.PushReceiver"
android:process=":PushService">
        <intent-filter >
            <action android:name="com.lenovo.lsf.intent.REGISTER" />
            <action android:name="com.lenovo.lsf.intent.UNREGISTER" />
            <action android:name="com.lenovo.lsf.intent.LOG_CONTROL" />
            <action android:name="com.lenovo.lsf.intent.PUSH_TYPE_CONTROL" />
            <action android:name="com.lenovo.lsf.intent.PUSH_TEST_MESSAGE" />
            <action android:name="com.lenovo.lsf.intent.SWITCH_ON_SERVICE" />
            <action android:name="com.lenovo.lsf.intent.SWITCH_OFF_SERVICE" />
        </intent-filter>
    </receiver>

    <service
        android:name="com.lenovo.lsf.push.service.PushService"
android:process=":PushService"
        android:exported="true">
    </service>

    <activity
        android:name="com.lenovo.lsf.push.ui.DisplayActivity"
android:process=":DisplayActivity"
        android:theme="@android:style/Theme.Translucent"
android:screenOrientation="portrait">
    </activity>

    <service
        android:name="com.lenovo.lsf.push.ui.DisplayService"
android:process=":PushService"
        android:exported="true">
    </service>

<!--Lenovo game 支付 -->
<!-- 收银台界面，默认横屏，通过screenOrientation可以进行修改-->
<activity
    android:name="com.lenovo.pay.api.PayManagerActivity"

android:configChanges="keyboardHidden|orientation|screenSize|navigation|locale|lay
outDirection"
    android:launchMode="singleTop"
    android:screenOrientation="landscape"
    android:theme="@style/com_lenovo_pay_theme_dialog" />

```



```

        <activity
            android:name="com.Lenovo.Lsf.Lenovoid.ui.minewebview.HomeHtmlActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:launchMode="standard"
            android:windowSoftInputMode="stateHidden|adjustPan"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
            android:screenOrientation="landscape">
                <intent-filter>
                    <action android:name="android.intent.action.VIEW" /><action
                        android:name="com.Lenovo.Lsf.Lenovoid.action.ACTION_START_WEB_VIEW"/><category
                            android:name="android.intent.category.DEFAULT" />
                </intent-filter>
            </activity>
<!-- 收银台界面结束-->
<!-- 如 无 法 注 册    "keyboardHidden|orientation|screenSize" ,    请 修 改 为
"keyboardHidden|orientation"-->
<!-- 易联 -->

```

注：以上全部组件需要添加在 <application>内部。

3.3.2增加 Permission

```

<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="com.lenovo.lsf.device.permission.MESSAGE" />
<uses-permission android:name="com.lenovo.lsf.device.permission.RECEIVE" />
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION" />
<uses-permission android:name="android.permission.RESTART_PACKAGES" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.EXPAND_STATUS_BAR" />

```

4. 集成联想网游 SDK 代码

通常情况下，推荐开发者使用如下的快速方式集成代码。快速接入只需要调用以下 **3** 个主要 API，每个 API 的详细说明见本文 5.3 部分。第一步：调用 SDK 初始化：

LenovoGameApi.doInit()

游戏应用初始启动时，调用 LenovoGameApi.doInit()接口进行游戏 sdk 的初始化。

4.2 第二步：调用自动登录: LenovoGameApi.doAutoLogin()

在原主入口 Activity 中调用快速登录接口 LenovoGameApi.doAutoLogin()，进行自动登录操作。（游戏启动时，联想游戏欢迎页会效果持续 1s，之后会进入到游戏原初始页面。）

4.3 第三步：调用支付: LenovoGameApi.doPay()

游戏中需要支付时，调用联想网游 SDK 的 LenovoGameApi.doPay()接口进行支付。

4.4 第四步：调用实名制: LenovoGameApi.doRegistRealAuth()

游戏中需要实名认证时，调用联想网游 SDK 的 LenovoGameApi.doRegistRealAuth()接口进行认证。

4.5 第五步：调用是否成年: LenovoGameApi.doCheckRealAuth()

游戏中需要查询是否实名认证及是否成年时，调用联想网游 SDK 的 LenovoGameApi.doCheckRealAuth()接口进行查询。

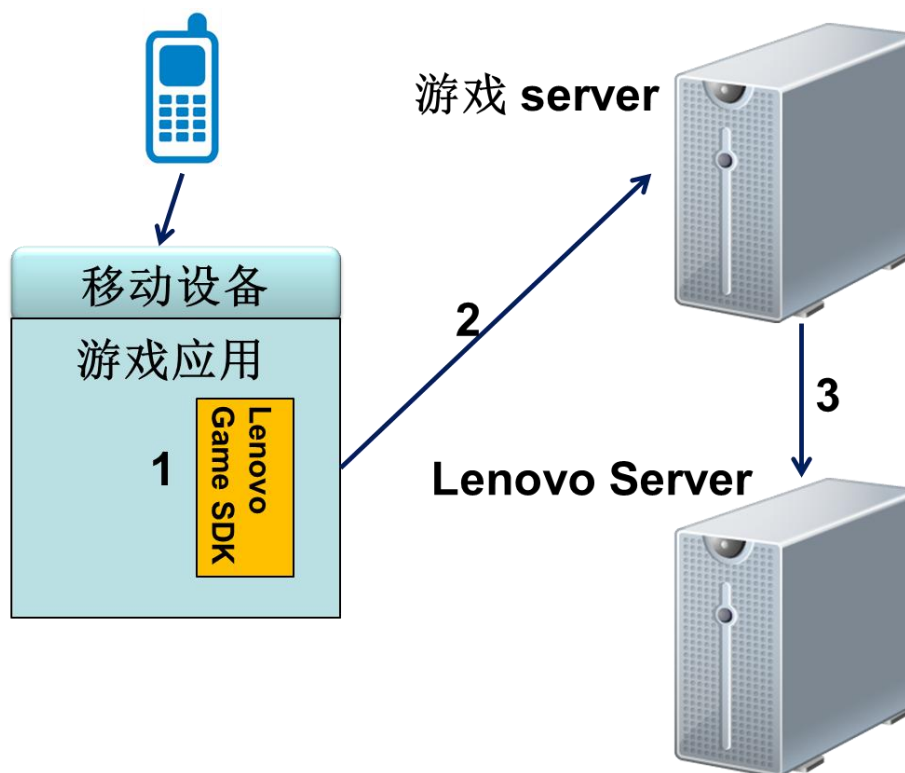
4.6 退出接口：调用退出：LenovoGameApi.doQuit()

游戏中需要调用联想的退出接口 LenovoGameApi.doQuit 实现退出操作，并且出现联想的广告位

4.7 示例代码

以上接口的示例代码在联想提供的 Demo_SRC 中。具体路径为：
Demo_SRC\src\com\lenovo\id\pay\sample*。

5. 联想网游 SDK 详细说明 Lenovo ID 身份认证流程

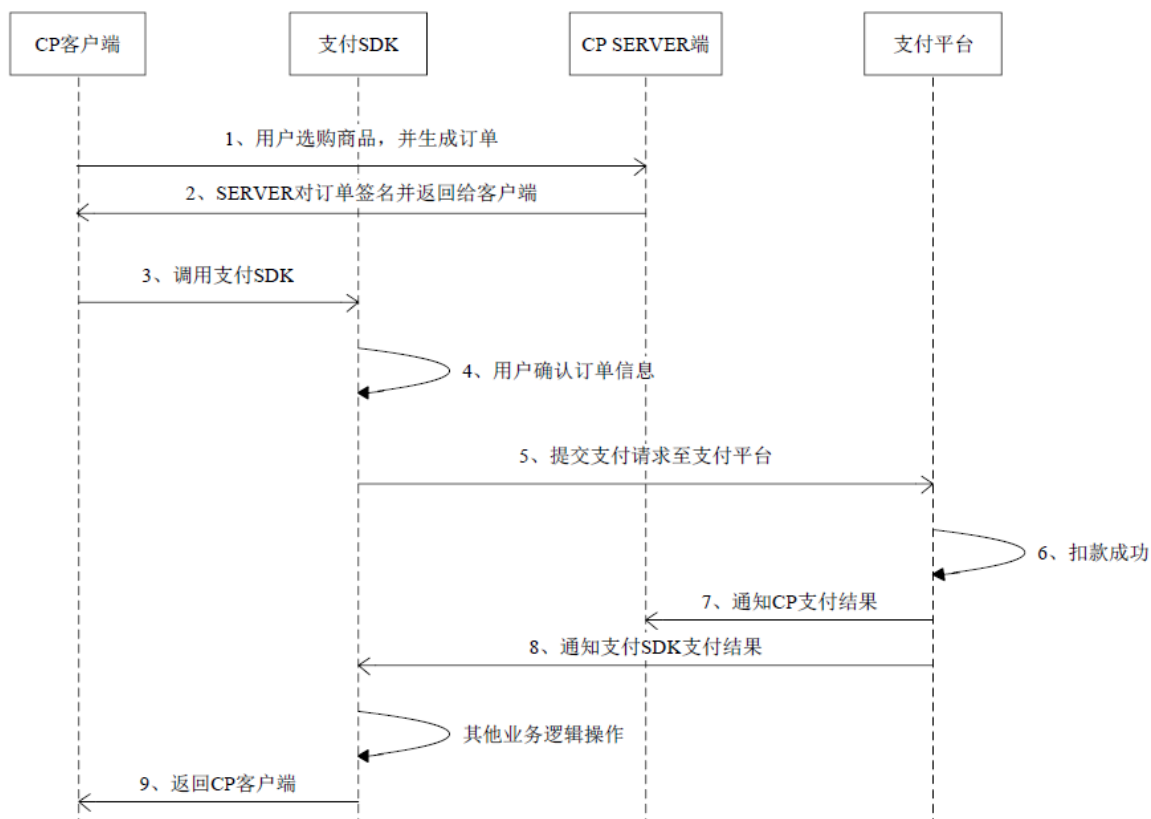


- 1) 集成联想网游 SDK，登录取得 ST (token)
- 2) 将获取到的 token 传递到游戏服务器
- 3) 通过 Lenovo ID Web API 验证 ST 的有效性，同时获取用户的唯一 ID

5.2 联想支付流程

- 联想支付平台目前提供的购买模式为同步购买。
- 在调用接口后，应程序需要同步等待支付结果的通知。
- 在玩家使用联想快捷支付支付成功后，支付平台会通知应用程序客户端。
- 如果应用有服务端对接，支付结果也会通知到游戏服务端的地址。
- 支付结果通知的详细介绍请参照 5.3.5 支付和结果校验 API 部分的说明

具体支付流程如下：



5.3 API 详细说明

5.3.1 SDK 初始化 API (必接)

API : `LenovoGameApi.doInit(Activity activity,String appid)`

功能 : 进行 gamesdk 使用前的初始化工作。**注意** : 此接口必须主线程调用。**如果 manifest 中 sdktarget >=23 的话 请在初始化之前 调用原生接口 requestPermissions 申请读写 sd 卡权限，否则会无法显示广告。强烈建议 sdktarget 小于 23。**

参数 :

activity : 注意这里一定要传 Activity 实例的 context,否则在 ICS 以下版本的 android 系统上可能会出现类型转换错误 ;

appid : 用户申请的 open AppID ;

调用示例:

```

private void initView() {
    // 加载游戏资源
    setContentView(R.layout.load_resource);
    //SDK初始化
    LenovoGameApi.doInit(GameLauncherActivity.this,Config.appid);
    .....
}
  
```

5.3.2 自动登录 API (必接)

API : `LenovoGameApi.doAutoLogin(Activity activity, final IAuthResult callback)`

功能 : 执行用户登录, 获取 ST 并通过 `IAuthResult` 返回给应用。

参数 :

`activity` : 注意这里一定要传 `Activity` 实例的 `context`, 否则可能会出现类型转换错误 ;

`callback` : 完成请求后的返回值调用对象

`IAuthResult`: 登录结果会通过 `IAuthResult` 类返回给游戏。游戏可以通过 `override onFinished()` 函数来取得登录结果数据

注意 : 此接口必须主线程调用, 接口里面已经启动线程, `IAuthResult` 必须设置。不要在回调函数中执行 UI 操作, 如果需要操作 UI, 请发消息到主线程处理。

调用示例:

```
private void getTokenByQuickLogin() {
    LenovoGameApi.doAutoLogin(this, new IAuthResult() {
        @Override
        // 登录回调, 处理登录结果
        public void onFinished(boolean ret, String data) {
            if (ret) {
                // 登录成功
            } else {
                // 登录失败(失败原因开启飞行模式、网络不通等)
            }
        }
    });
}
```

5.3.3 帐号登录回调 API

API : `void onFinished(boolean ret, String data)`

功能 : 登录接口、获取 ST 接口中的异步调用回调接口函数, 认证结束会调用该接口函数。

参数 :

`ret` 返回认证状态 : `true` 成功, `false` 失败 ;

`data` 成功时为 ST 数据 ;
 失败时 `null`

调用示例: 见 `LenovoGameApi.doAutoLogin()`

5.3.4 查询是否实名及成年 API (可选)

API : `LenovoGameApi.doCheckRealAuth (Context context, String st, String realmID, final IAuthResult callback)`

功能 : 查询是否实名及成年并通过 `IAuthResult` 返回给应用。

参数：

context: context

st: 用户的 st 信息

realmID:openappid

callback：完成请求后的返回值调用对象

IAuthResult:查询结果会通过 IAuthResult 类返回给游戏。游戏可以通过 override onFinishied()

函数来取得查询结果数据

调用示例：

```
LenovoGameApi.doCheckRealAuth(GameLauncherActivity.this, Ipsust, new IAuthResult() {
    @Override
    public void onFinishied(boolean ret, String data) {
        if(ret){//已经实名注册
            String[] a = data.split("-");
            Log.i("demo", "doCheckRealAuth a:[" +a[0]+"," +a[1]+"]");

            if(RealAuthConstants.REAL_AUTH_OVER_EIGHTEEN.equals(a[1])){
                Toast.makeText(GameLauncherActivity.this, "已实名认证，已成年，
出生日期："+a[0], Toast.LENGTH_SHORT).show();
            }else
            if(RealAuthConstants.REAL_AUTH_UNDER_EIGHTEEN.equals(a[1])){
                Toast.makeText(GameLauncherActivity.this, "已实名认证，未成年，
出生日期："+a[0], Toast.LENGTH_SHORT).show();
            }
            }else{
                Toast.makeText(GameLauncherActivity.this, "尚未实名认证",
Toast.LENGTH_SHORT).show();
            }

        }
    }
});
```

5.3.5 实名认证(可选)

API：LenovoGameApi.doRegistRealAuth (Context context, String st, String realmID, Boolean force, final IAuthResult callback)

功能：查询是否实名及成年并通过 IAuthResult 返回给应用。

参数：

context: context

st: 用户的 st 信息

force: 是否强制注册，此项为 true 时，弹出的实名认证对话框不可取消，必须实名认证成功才能认证成功。

callback：完成请求后的返回值调用对象

IAuthResult:查询结果会通过 IAuthResult 类返回给游戏。游戏可以通过 override onFinishied()

函数来取得查询结果数据

调用示例：

```
LenovoGameApi.doRegistRealAuth(GameLauncherActivity.this, Ipsust, false/*是否强制注册*/, new
IAuthResult() {
```

```

@Override
public void onFinish(boolean ret, String data) {
    if(ret){//实名认证成功
        String[] a = data.split("-");
        Log.i("demo", "doRegistRealAuth a:[" + a[0] + "," + a[1] + "," + a[2] + "]);

        if(RealAuthConstants.REAL_AUTH_SUCCESS_SURE.equals(a[0])){
            //有效验证
        }else
        if(RealAuthConstants.REAL_AUTH_SUCCESS_NOT_SURE.equals(a[0])){
            //快速验证 如果这种结果建议隔段时间再次查询doCheckRealAuth
        }
        if(RealAuthConstants.REAL_AUTH_OVER_EIGHTEEN.equals(a[2])){
            Toast.makeText(GameLauncherActivity.this, "实名认证成功, 已成年,
出生日期: " + a[1], Toast.LENGTH_SHORT).show();
        }else
        if(RealAuthConstants.REAL_AUTH_UNDER_EIGHTEEN.equals(a[2])){
            Toast.makeText(GameLauncherActivity.this, "实名认证成功, 未成年,
出生日期: " + a[1], Toast.LENGTH_SHORT).show();
        }
        }else{//实名认证失败
            if(RealAuthConstants.REAL_AUTH_FAIL_CANCEL.equals(data)){
                Toast.makeText(GameLauncherActivity.this, "实名认证失败:用户取消", Toast.LENGTH_SHORT).show();
            }else if(RealAuthConstants.REAL_AUTH_FAIL_ERROR.equals(data)){
                Toast.makeText(GameLauncherActivity.this, "实名认证失败:系统异常", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
});

```

5.3.6 获取 ST API (不常用)

API : `LenovoGameApi .getStData(Context context, IAuthResult callback)`

功能 : 一般情况下, 登录成功后, ST 会由回调接口返回, App 不需要单独调用此接口。如果游戏在其他时间需要使用 ST, 可以调用此接口再次获取。

参数 :

context : 注意这里一定要传 Activity 实例的 context, 否则可能会出现类型转换错误;

callback : 完成请求后的返回值调用对象

IAuthResult: 登录结果会通过 IAuthResult 类返回给游戏。游戏可以通过 override onFinish() 函数来取得登录结果数据。

注意 : 此接口必须主线程调用, 接口里面已经启动线程, IAuthResult 必须设置。不要在回调函数中执行 UI 操作, 如果需要操作 UI, 请发消息到主线程处理。

调用示例:

```

LenovoGameApi. getStData (GameLauncherActivity.this, new LenovoGameApi.IAuthResult()
{

```

```

@Override
public void onFinish(boolean ret, String data) {

    if (ret) {
        //登录并且获取ST成功
    } else {
        //获取ST失败
    }
}
});

```

5.3.7 支付和结果校验 API (必接)

API :

LenovoGameApi .doPay(Activity activity, String appkey, GamePayRequest request, IPayResult callback)

参数说明：设置参数的时候请注意参数字段的类型

参数名称	类型	说明	备注
appkey	String	应用支付密钥	接入时从联想获取
appid	String(20)	Open appid, 应用代码, 长度为 20 位的字符串	接入时从联想获取, 本字段不能为空
waresid	int	商品编码	接入时商户自建
notifyurl	String	交易结果同步回调地址	此字段客户端暂不使用, 请在CP自服务系统中配置回调地址
Exorderno	String	外部订单号, 长度小于 50 字节的字符串, 本字段不能为空, 且字段中不能有"&"或者"="字符。	外部订单号作为区分订单的标志, 同时作为在线支付成功后, 应用对支付结果签名的校验字段, 关系到支付安全, 请务必定义
price	int	开放价格策略填真实兑换的金额(单位为分)。其他策略填 0	单位为分
cpprivateinfo	String	商户私有信息。最大长度 128。本字段不能为空, 且字段中不能有"&"或者"="字符。	可选字段, 商户私有信息在做交易结果同步的时候回传给开发者

调用示例代码：

```

GamePayRequest payRequest = new GamePayRequest();
// 请填写商品自己的参数
payRequest.addParam("notifyurl", ""); //notifyurl当前版本暂时不用, 传空string
payRequest.addParam("appid", Config.appid);

```



```

        payRequest.addParam("waresid", waresid);
        payRequest.addParam("exorderno", "sample" + System.currentTimeMillis());
        payRequest.addParam("price", price);
        payRequest.addParam("cppprivateinfo", "123456");
        LenovoGameApi.doPay(GoodsListActivity.this, Config.appkey, payRequest, new
IPayResult() {
            @Override
            public void onPayResult(int resultCode, String signValue,
                String resultInfo) { // resultInfo = 应用编号&商品编号&外部订单号
                if (LenovoGameApi.PAY_SUCCESS == resultCode) {
                    Toast.makeText(GoodsListActivity.this,
                        "sample:支付成功", Toast.LENGTH_SHORT)
                        .show();
                } else if (LenovoGameApi.PAY_CANCEL == resultCode) {
                    Toast.makeText(GoodsListActivity.this, "sample:取消支付",
                        Toast.LENGTH_SHORT).show();
                    // 取消支付处理，默认采用finish(), 请根据需要修改
                    Log.e(Config.TAG, "return cancel");
                } else {
                    Toast.makeText(GoodsListActivity.this, "sample:支付失败",
                        Toast.LENGTH_SHORT).show();
                    // 计费失败处理，默认采用finish(), 请根据需要修改
                    Log.e(Config.TAG, "return Error");
                }
            }
        });

```

支付结果通知说明：

支付成功后，支付平台通过两种方式告知游戏支付结果

1) 客户端支付结果通知：

在判断支付的结果是SDKApi.PAY_SUCCESS之后，调用游戏服务端来验证支付结果是否正确，只有在这两个都验证通过情况下才能算支付成功

2) 服务端支付结果通知：

您可以通过客户端参数notifyurl来设置支付结果通知地址，用户支付成功后，我们会将支付的相关信息同步到您设置的地址。另外您也需要在联想商户服务后台配置一个缺省的地址。缺省地址的设置方式，登录联想移动应用内计费平台（<http://paydev.lenovomm.com/index/index.shtml>），选择应用后，修改应用信息，设置通知地址。

应用信息 [\[保存\]](#)


应用名称: 热血斗地主

应用编号: 10000100000033100001

支付私钥: NjdENThFMkQ2M0ExODhFMjlCQjJGMzQ2REVGNzQxNjc4NEI1QjY3Mk1URXlPRGt4T0RneU5ESXhNVGt4T1RFeU1Ua3JNVGcyTkRjeU1UVTRPRF
UyTlRZeE5USTNNVfV3TWpZeE5EQXd0akUyTURVNU9ETTFN=1V4


[查看老版本密钥 \(注3.2.35之前的sdk, 含3.2.35\)](#)

☒ 接收支付结果通知

 如果勾选了接收支付结果通知, 则支付结果请以本通知为准而非客户端的API返回结果。

接收地址:

[通知接口测试](#)

 建议点击“通知接口测试”: 系统会发送模拟交易结果数据到接收地址, 用以验证商户的通知接口是否可用。

6. 混淆配置

如果您需要混淆 apk, 要在工程中 proguard.cfg 文件内添加如下的混淆参数, 避免联想网游 SDK 服务提供的类被混淆。

```
-dontwarn android.net.http.**
-keep class android.net.http.**{ *;}
-dontskipnonpubliclibraryclasses
-dontoptimize
-dontusemixedcaseclassnames
-dontskipnonpubliclibraryclasses
-dontpreverify
-keepattributes SourceFile,LineNumberTable
-verbose
-optimizations !code/simplification/arithmetic,!field/*,!class/merging/*

-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.app.backup.BackupAgentHelper
```

```
-keep public class * extends android.preference.Preference
-keep public class * extends android.support.v4.view.ViewPager$OnPageChangeListener
-keep public class * extends android.support.v4.view.PagerAdapter

-keep class org.apache.commons.*
-dontwarn org.apache.commons.**

-dontwarn com.lenovo.pay.sign.**
-keep class com.ut.*

-keep class android.content.pm.IPackageInstallObserver {
public <fields>;
public <methods>;
}

-keep class android.content.pm.IPackageInstallObserver.Stub {
public <fields>;
public <methods>;
}

-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet);
}
```

```

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet, int);
}

-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keep class * implements android.os.Parcelable {
    public static final android.os.Parcelable$Creator *;
}

-keep class com.lenovo.lsf.lenovoid.ui.WebViewActivity$LenovoID {
public <fields>;
public <methods>;
}

-keep class com.lenovo.lsf.lenovoid.ui.WebViewActivity$JsProcessClass {
public <fields>;
public <methods>;
}

-keep class com.lenovo.lsf.gamesdk.LenovoGameApi {
public <fields>;
public <methods>;
}

-keep class com.lenovo.lsf.gamesdk.LenovoGameApi.IQuitCallback {
public <fields>;

```

```

public <methods>;
}

-keep class com.lenovo.lsf.gamesdk.LenovoGameApi.GamePayRequest {
public <fields>;
public <methods>;
}

-keep class com.lenovo.lsf.gamesdk.LenovoGameApi.IPayResult {
public <fields>;
public <methods>;
}

-keep class com.lenovo.pay.**{ *;}

-keep class com.lenovo.lsf.lenovoid.ui.WebViewActivity

-dontwarn android.support.v4.**
-keep class android.support.v4.view.** { *; }
-keep interface android.support.v4.view.** { *; }
-dontwarn android.support.app.**
-keep class android.support.v4.app.** { *; }
-keep interface android.support.v4.app.** { *; }
-keep class android.support.v4.content.Loader {}
-keep class com.lenovo.lsf.push.** { *; }
-dontwarn com.lenovo.lsf.push.**
-keep interface com.lenovo.lsf.push.** { *; }
-keep class com.lenovo.lsf.lenovoid.advertise.** { *; }
-dontwarn com.lenovo.lsf.lenovoid.advertise.**
-keep interface com.lenovo.lsf.lenovoid.advertise.** { *; }

-keep class com.lenovo.pay.** { *; }
-dontwarn com.lenovo.pay.**
-keep interface com.lenovo.pay.** { *; }
#不混淆 org.apache.http.legacy.jar
-dontwarn android.net.compatibility.**
-dontwarn android.net.http.**
-dontwarn com.android.internal.http.multipart.**
-dontwarn org.apache.commons.**
-dontwarn org.apache.http.**
-keep class android.net.compatibility.**{*;}
-keep class android.net.http.**{*;}
-keep class com.android.internal.http.multipart.**{*;}
-keep class org.apache.commons.**{*;}
-keep class org.apache.http.**{*;}
-keep class org.apache.http.entity.mime.** { *; }
-keep class cz.msebera.android.httpclient.** { *; }
-keep class com.loopj.android.http.** { *; }

-dontwarn org.apache.http.impl.client.**

```

```

-keep class org.apache.http.impl.client.**{*;}
-keepclasseswithmembers                                     class
org.apache.http.impl.client.DefaultHttpRequestRetryHandler {
    public <init>();
    public <init>(int, boolean);
    public private protected <fields>;
    public private protected <methods>;
}
-keepclasseswithmembers class com.loopj.android.http.AsyncHttpClient {
    public private protected <fields>;
    public private protected <methods>;
    *;
}
-keepclasseswithmembers class com.loopj.android.http.AsyncHttpResponseHandler {
    public private protected <fields>;
    public private protected <methods>;
    *;
}
-keepclasseswithmembers class com.loopj.android.http.PersistentCookieStore {
    public private protected <fields>;
    public private protected <methods>;
    *;
}
-keepclasseswithmembers class com.loopj.android.http.RequestParams {
    public private protected <fields>;
    public private protected <methods>;
    *;
}
-keep class de.greenrobot.** {
    <fields>;
    <methods>;
}

-keep class com.lenovo.lps.reaper.sdk.db.** {
    <fields>;
    <methods>;
}
-keep class **AnalyticsTracker {
    <methods>;
}
-keep class com.lenovo.pop.api.** { *; }
-dontwarn com.lenovo.pop.api.**
-keep interface com.lenovo.pop.**{*;}
-keep class com.lenovo.pop.user.**{*;}
-dontwarn com.lenovo.pop.user.**
-keep class com.lenovo.pop.ui.**{*;}
-dontwarn com.lenovo.pop.ui.**
-keep class com.lenovo.pop.utility.**{*;}
-dontwarn com.lenovo.pop.utility.**

```

```

-keep interface com.lenovo.pop.api.OnFinishListener {
    public <fields>;
    public <methods>;
    *;
}
-keep public class com.lenovo.lsf.lenovoid.ui.miniwebview.** { *; }
-keepclasseswithmembers                                     class
com.lenovo.lsf.lenovoid.ui.miniwebview.HomeJsInterface {
    public private protected <fields>;
    public private protected <methods>;
}

```

```
*;  
}
```

7. 接入注意事项

1) 接入关键检查点

序号	接入关键点	备注
1.	配置和使用正确的 key 值	要在配置中正确使用 open AppID 信息,在代码中正确使用支付的 open AppID、appkey 和 waresid。
2	调用初始化接口	调用 SDK 初始化接口,初始化 SDK 运行环境
3	调用自动登录接口	代码中调用快速登录接口完成用户登录,获得 token。
4	服务器验证 Token	把获取的登录 token 上传到自己的服务器,然后通过后台服务器之间的接口获取用户的 UID 来唯一的标识当前用户。
5	支付	用户点击购买时调用支付接口完成支付。
6	其他检查	导出的 apk 请检查相关资源是否已打包进来 (so 文件、res 目录等等)

2) 独立包名

需要给联想单独的游戏的包名(PN),请在包名后缀上加入.lenovo 字样

8. 服务端验证 Lenovo ID Token(ST)

1) 接口描述

◇ 接口编号	API-63
◇ 访问 URL	http://passport.lenovo.com/interserver/authen/1.2/getaccountid
◇ 通讯方式	HTTP GET
◇ 参数:	lpsust: LPS user account ticket, 用来标志用户身份的一个 ticket, 这个值请传入获取到的 ST (Token)
	realm: 服务安全域标识, 参数值为联想颁发的 open AppID。
◇ 描述:	根据客户端获取的 lpsust 以及获取该 lpsust 的 realm 来获取用户信息, 具体信息参见对应的接口文档。 如果可以获取该用户信息, 证明 lpsust 合法, 并且是由该 realm 对应生成, 如果不能获取该用户信息, 则表明该 lpsust 不合法, 或者 realm 不正确。

--	--

2) http 示例

http://passport.lenovo.com/interserver/authen/1.2/getaccountid?lpsust=ZAgAAAAAAGE9MTAwMDM1NTA4MDMmYj0yJmM9NCZkPTEwJmU9RTZGM0EzMTY5RjAwQTM2QzE4MzNERDM4QzhCQkU0QzkxJmg9MTM3MjkxMDg2NDI3NSZpPTQzMjAwJm89MDAwMDAwMDAwMDAwMDAwJnA9aW1laSZxPTEwMTExMSZlc2VybmFtZT0xMzg5MDUzNTg4N6z979s5fL06DibrT5d7D6s=&realm=appstore.lps.lenovo.com

3) 用户信息 Schema

以下是使用XSD(xml schema definition)描述的接口返回用户信息的xml结构。实际返回的XML信息见后面的示例。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="IdentityInfo" nillable="false">
    <xs:annotation>
      <xs:documentation>用户信息</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:all>
        <xs:element name="AccountID" type="xs:long">
          <xs:annotation>
            <xs:documentation>对于用户帐号，该字段为用户ID。对于PID帐号，
该字段为PID值。</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Username" type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>用户名（可选项）</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="DeviceID" type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>登录所用设备ID（可选项）
</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



```

        <xs:element name="verified" minOccurs="0">
            <xs:annotation>
                <xs:documentation>帐户是否已激活。0:未激活, 1: 已激活。
            </xs:documentation>
            </xs:annotation>
        </xs:element>

<xs:element name="Thirdname" type="xs:string" minOccurs="0">
    <xs:annotation>
        <xs:documentation>3rd party IDP' s name. see section
        "Common Data Format" </xs:documentation>
    </xs:annotation>
</xs:element>

</xs:all>
</xs:complexType>
</xs:element>
</xs:schema>

```

实际返回的用户信息 XML 示例:

```

<?xml version="1.0"
encoding="UTF-8"?><IdentityInfo><AccountID>10003550803</AccountID><Username
>13810535887</Username><DeviceID>123</DeviceID><verified>1</verified></Identit
yInfo>

```

4) 错误信息 Schema

以下是使用 XSD(xml schema definition)描述的接口返回错误信息的 xml 格式结构。实际返回的 XML 信息见后面的示例。

```

<?xml version=" 1.0" encoding=" UTF-8" ?>
<xs:schema xmlns:xs=" http://www.w3.org/2001/XMLSchema"
elementFormDefault=" qualified" attributeFormDefault=" unqualified" >
<xs:element name=" Error" >
    <xs:annotation>
        <xs:documentation>错误信息</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:all>
            <xs:element name=" Code" type=" xs:string" >

```

```

        <xs:annotation>
            <xs:documentation>错误码</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name=" Timestamp" type=" xs:dateTime" />
    <xs:element name=" Message" minOccurs=" 0" >
        <xs:annotation>
            <xs:documentation>错误信息</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base=" xs:string" >
                <xs:maxLength value=" 512" />
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name=" Detail" nillable=" true" minOccurs=" 0" >
        <xs:annotation>
            <xs:documentation>错误详细信息</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base=" xs:string" >
                <xs:maxLength value=" 1024" />
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name=" Source" minOccurs=" 0" >
        <xs:annotation>
            <xs:documentation>错误源</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base=" xs:string" >
                <xs:maxLength value=" 256" />
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name=" URL" type=" xs:anyURI" minOccurs=" 0" >
        <xs:annotation>
            <xs:documentation>错误描述相关URL</xs:documentation>
        </xs:annotation>

```

```

        </xs:element>
    </xs:all>
</xs:complexType>
</xs:element>
</xs:schema>

```

返回错误信息XML示例：

```

<?xml version="1.0" encoding="UTF-8"?>
<Error>
<Code>USS-0121</Code>
<Timestamp>2014-07-01T17:51:49+08:00</Timestamp>
</Error>

```

5) 错误码

错误码(Code)	描述
USS-0100	无效的用户名，需要检查用户名格式是否正确。
USS-0101	口令错误。
USS-0103	无此用户，请检查用户名是否正确
USS-0104	用户名已存在，不允许重复注册
USS-0105	帐号必须激活后才能登录
USS-0108	该用户已激活，请勿重复操作。
USS-0110	无效的 IMEI，SN 或 MAC
USS-0111	帐号已被 disable
USS-0113	口令类型错误
USS-0121	无效的 realm
USS-0122	此服务不支持该 realm
USS-0126	Ticket 值解析失败。
USS-0135	无效的请求数据
USS-0151	账号已锁定
USS-0160	需要使用验证码（在申请帐号时，如果服务端检测到异常行为，会返回该错误给客户端，客户端需要使用图形验证码）
USS-0170	密码格式错误。（密码的限制规则是： 4 ~ 20 位字符，包括英文大小写字母、英文数字、减号和下划线）
USS-0181	校验码错误或失效，针对手机账号对账号校验时可能产生
USS-0190	（针对短信注册，服务端不支持该运营商的号码。） 尚未开通，请尝试其他方式注册。
USS-0540	无效的 Ipsust
USS-0542	未提供 Ipsust 信息
USS-0202	用户登录已失效

USS-0x0000	后台快捷登录使用次数达到 10 次
USS-0x0001	后台快捷登录失败
USS-0x0002	后台快捷登录检测到未安装联想通行证

9. 服务端支付通知接口 本接口用于交易完成之后，联想支付平台主动向商户服务端发起交易结果同步，结果将通知到您在联想商户服务后台设置的“交易结果通知” URL 地址。

1) 接口参数

接口均采用 http 协议，POST 方法。POST 参数为 transdata、sign。transdata 为本次支付的具体业务参数，数据格式为 json 格式；sign 为 transdata 的签名数据。具体呈现方式为 transdata=xxxx&sign=yyyy，其中 yyyy 就是对 xxxx 的签名数据。

2) 应答与签名

商户收到平台数据后，需要使用自服务平台提供的支付密钥 appkey 来验证签名。如果验证签名失败，返回字符串 FAILURE，通过则给支付平台返回字符串 SUCCESS 应答。

3) 交易结果通知的重发

平台在没有收到商户应答的情况下，会定时重发。但是重发一定次数后，将不再进行重发。

4) 验签方式

本方法用于支付结果通知的验证签名。收到平台的通知后，解析出 transdata 和 sign 后，请使用 validSign 方法验证签名是否通过。如果不通过，请勿进行其它操作，避免损失。验证签名的方法见服务端接入示例代码。方法依赖的 jar 包 lenovo_pay_sign-1.0.0.jar 包含在联想网游 SDK 发布包中。

验证签名的方法函数原型:

```
/**
 * 密钥验签
 * @param transdata 待加密数据
 * @param sign      同步签名
 * @param appkey    CP支付密钥
 * @return
 */
public static Boolean validSign(String transdata,String sign,String appkey)
```

调用方法:

```
CpTransSyncSignValid.validSign(transdata,sign, appkey);
```

同步数据示例

同步数据具体呈现方式 (http 包体数据):

```
transdata={"exorderno":"1","transid":"2","appid":"3","waresid":31,"feetype":4,"money":5,"count":6,"result":0,"transtype":0,"transtime":"2012-12-12 11:10","cpprivate":"7","paytype":1}&sign=d91cbc584316b9d99919921a9
```

同步数据参数详情

解参数名称	参数含义	数据类型	参数长度	参数说明
exorderno	外部订单号	String	Max(50)	商户生成的订单号
transid	交易流水号	String	Max(32)	计费支付平台的交易流水号
appid	游戏 id	String	Max(20)	Open AppID
waresid	商品编码	integer	Max(8)	商品编号
feetype	计费方式	integer	Max(3)	计费方式 :0、按次 ;1、开放价格 ;
money	交易金额	integer	Max(10)	本次交易的金额，单位：分
count	购买数量	integer	Max(10)	本次购买的商品数量
result	交易结果	integer	Max(2)	交易结果：0-交易成功；1-交易失败
transtype	交易类型	integer	Max(2)	交易类型：0-交易；1-冲正
transtime	交易时间	String	Max(20)	交易时间格式：yyyy-mm-dd hh24:mi:ss
cpprivate	商户私有信息	String	Max(128)	商户私有信息
paytype	支付方式	Integer	Max(2)	支付方式(该字段值后续可能会增加) 0-话费支付 1-充值卡 2-游戏点卡 3-银行卡 401-支付宝 402-财付通 5-联想币 6-联想一键支付