JAVA游戏客户端接入

一、概述

1. 此文档为Android原生游戏接入文档，使用第三方引擎的游戏开发者请参考相应接入文档。
2. 接入中涉及的产品代码及测试渠道代码等接入信息需通过开发者后台获取。文档中涉及到的Demo可以从SDK下载中下载。
3. 本文档对应的SDK版本为V1.5.0及以上.
   注意：请勿在代码中使用LJMainActivity字眼

二、接入准备

2.1、工程导入

将Demo项目中libs目录下jar包复制到游戏工程libs目录下； 将Demo项目中assets目录下的文件复制到游戏项目的assets目录下；

2.2、配置AndroidManifest.xml文件

在<application>标签内加入如下权限声明：

```
  <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.GET_TASKS" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" /
>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAG
E" />
  <uses-permission android:name="android.permission.BLUETOOTH" />
```

在开发者后台添加完产品后，在产品信息中会得到**产品代码**和**测试渠道代码**，将这两个参数配置到<application>标签内：

```
  <meta-data android:name="XMGAME_PRODUCT_CODE" android:value="产品代码"
/>
  <meta-data android:name="XMGAME_CHANNEL_CODE" android:value="测试渠道代
码" />
```

<application>标签的属性**android:name**必须配置为**com.xinmei365.game.proxy.XMApplication**或其子类：

```
<application
    android:name="com.xinmei365.game.proxy.XMApplication"
    android:label="@string/app_name"
    android:icon="@drawable/icon">
```

在Application节点下增加登录Activity声明，否则会影响本地登录调试：

```
<activity
      android:name="com.xinmei365.game.proxy.XMSDKLoginActivity" />
```

2.3、设置包名

SDK接入完成后，上传的游戏母包需要使用特定的包名，包名规范为：com.ljapps.+产品code，如游戏产品code为p646，则包名可设置为com.ljapps.p646（该包名仅用于我方接口检测，并不是最终渠道包名，各渠道包名可在后台自定义配置），设置方法为修改AndroidManifest.xml文件中manifest标签下的package属性的值，如下所示：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ljapps.p646"
    android:versionCode="1"
    android:versionName="1.0" >
```

三、接口接入

> *使用SDK必须接入的接口：闪屏接口、初始化接口、生命周期接口、用户接口、支付接口、退出接口、用户信息扩展接口*

3.1、闪屏接口

定义一个Activity，继承 **com.xinmei365.game.proxy.XMSplashActivity**：

```
public class MySplashActivity extends XMSplashActivity {
    public int getBackgroundColor() {
        //当闪屏PNG图片无法铺满部分机型的屏幕时，设置与闪屏颜色配合的背景色会给用户更好的体验
        return Color.WHITE;
    }
    @Override
    public void onSplashStop() {
        //闪屏结束后，启动游戏的Activity
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        this.finish();
    }
}
```

并在AndroidManifest.xml文件中将该类设置为游戏启动类：

```
<activity
    android:name="包名+.MySplashActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:screenOrientation="portrait" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

> *接入该接口后，开发者可以在游戏后台灵活配置闪屏内容、数量、次序等，不配置闪屏图片则不出现闪屏,该接口不会影响程序原有闪屏（母包中闪屏示例图无需处理，会被平台自动删除；该接口不会影响到程序原有闪屏）。*

• 3.2、初始化接口

初始化方法，在游戏的第一个Activity(非闪屏Activity)中 onCreate() 方法中调用，并处理回调：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    GameProxy.getInstance().init(this, new XMInitCallback() {
        @Override
        public void onInitSuccess() {
            //初始化成功之后才可调用其他接口
        }
        @Override
        public void onInitFailure(String msg) {
            //初始化失败
        }
    });
}
```

• 3.3、生命周期接口

在游戏各个Activity（除闪屏Activity外）生命周期中调用SDK生命周期接口：

```
public void onCreate() {
    super.onCreate();
    GameProxy.getInstance().onCreate(activity);
}
public void onStop() {
```

```
        super.onStop();
        GameProxy.getInstance().onStop(activity);
    }
    public void onDestroy() {
        super.onDestroy();
        GameProxy.getInstance().onDestroy(activity);
    }
    public void onResume() {
        super.onResume();
        GameProxy.getInstance().onResume(activity);
    }
    public void onPause() {
        super.onPause();
        GameProxy.getInstance().onPause(activity);
    }
    public void onStart() {
        super.onStart();
        GameProxy.getInstance().onStart(activity);
    }
    public void onRestart() {
        super.onRestart();
        GameProxy.getInstance().onRestart(activity);
    }
    protected void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
        GameProxy.getInstance().onNewIntent(intent);
    }
    protected void onActivityResult(int requestCode, int resultCode,Intent
 data) {
        super.onActivityResult(requestCode, resultCode, data);
        GameProxy.getInstance().onActivityResult(activity,requestCode,resu
ltCode, data);
    }
```

- ## 3.4、用户接口
- ### 3.4.1、设置用户监听器

在用户登录登出相关的Activity的onCreate()方法中调用（由于该接口包含登录及登出回调，所以必须在调用登录登出接口前调用）：

```
    GameProxy.getInstance().setUserListener(activity, new XMUserListene
r() {
        @Override
        public void onLoginFailed(String reason, Object customParams) {
        }
```

```
        @Override
        public void onLoginSuccess(XMUser user, Object params) {
        }
        @Override
        public void onLogout(Object params) {
        }
    }
```

接口说明：

a）public void onLoginSuccess(XMUser user, Object params) 方法在调用SDK的登录方法并登录成功后被调用，参数说明：参数 params为调用登录接口时传入的自定义参数，此处原样返回；参数 XMUser user的数据结构如下：

```
    public class XMUser {
        private String channelCode;  // 渠道编码，用来区分用户来自哪个渠道，非空
        private String username;  //用户昵称，可能为空
        private String userID;    // 我方给出的不冲突的用户ID，非空
        private String token;    // 令牌，非空
        private String productCode;    // 产品代码，非空
        private String channelUserId;    // 渠道用户id，渠道原始用户ID，非空
        private String channelLabel;    //渠道标识，非空
    }
```

为了保证登录信息的有效性与合法性，请务必进行登录验证操作（否则无法提交到开发者后台）即：游戏客户端将得到的XMUser信息上传到游戏服务器，游戏服务器与我方服务器通信进行登录信息验证，游戏服务器将验证结果通知游戏客户端，游戏客户端根据验证结果允许玩家进入游戏或重新登录(参考服务端接入文档)

b）public void onLoginFailed(String reason,Object params) 方法在调用SDK的登录方法并登录失败后被调用，登录失败可能的原因有：

1）用户名或密码输入错误

2）AndroidManifest文件中配置的信息（渠道代码、产品代码）错误

3）开发者后台填入的渠道参数错误

4）渠道服务异常

登录失败时，请仔细排查出错原因

c）public void onLogout(Object params) 在如下情况会被调用:

1）点击游戏内的登出按钮，调用SDK的登出方法后

2）点击渠道用户中心界面的注销按钮

3）点击渠道用户中心界面的切换帐号按钮

游戏中没有调用logout接口但收到渠道的登出回调时同样会调用该回调，并不是只有调用了logout接口后才会收到该回调。请确保在游戏中任何时候，当该

> 方法被调用时游戏均可正确登出，回到登录页面。

- ## 3.4.2、登录接口

```
    GameProxy.getInstance().login(Activity activity,Object customParam
s);
```

> **参数说明：**
>
> activity ：当前Activity，请勿传入Application Context
> customParams ：用户自定义参数，在onLoginSuccess及onLoginFailed回调中原样返回
>
> **接口说明：**
>
> **请在初始化成功后，调用此接口。**

- ## 3.4.3、登出接口

```
    GameProxy.getInstance().logout(Activity activity,Object customPara
ms);
```

> **参数说明：**
>
> activity 当前Activity，请勿传入Application Context
> customParams自定义参数，在onLogout(String params)回调中原样返回
>
> **接口说明：**
>
> 登出操作用于用户切换帐号等事件处理，有如下不同处理方式：
> a）若游戏中存在登出或者切换帐号的按钮，则可在点击按钮时进行登出接口调用，在登出回调中进行重新登录等操作
> b）若游戏中不存在登出或者切换帐号的按钮时，建议修改游戏添加登出或切换帐号按钮，若实在无法添加，可在退出游戏前调用登出接口，这种情况下会存在部分渠道会审核不通过的情况，需游戏方与渠道去进行沟通。

- ## 3.5、支付接口

支付收到onSuccess回调后，请到游戏服务器查询订单状态再进行道具发放，请勿直接做支付成功的UI提示。

```
    public void pay(Activity context,XMPayParams params,PayCallBack pa
yCallBack)
```

> **参数说明：**
>
> context 上下文Activity
> params 支付参数
> payCallBack 支付回调接口

示例代码：

```
        XMPayParams params = new XMPayParams();
        params.setAmount(600);//支付金额,单位人民币分
        params.setItemName("水晶");//商品名称
        params.setCount(60);//购买数量
        params.setCustomParam("test pay");//自定义参数
        params.setCallbackUrl("http://testpay");//支付结果通知地址，即游戏服务器地址,交易结束后，我方会向该地址发送通知，通知交易的金额，customParams等信息
        params.setChargePointName("60水晶");//计费点名称，用于有计费点的渠道，没有可传空
        GameProxy.getInstance().pay(MainActivity.this, params, new PayCallBack() {
                @Override
                public void onSuccess(String sucessInfo) {
                    // 此处回调仅代表用户已发起支付操作，不代表是否充值成功，具体充值是否到账需以服务器间通知为准；
                }
                @Override
                public void onFail(String failInfo) {
                    // 此处回调代表用户已放弃支付，无需向服务器查询充值状态
                }
            });
```

接口说明：

支付回调提供了onSuccess及onFail两种通知，但充值是否到账还需以服务器间通知为准，不可将此作为判断充值是否成功的标志。

public void onSuccess(String successInfo)，该回调仅代表用户已发起支付操作，不代表是否充值成功，具体充值是否到账需以服务器间通知为准；在此回调中游戏方可开始向游戏服务器发起请求，查看订单是否已支付成功，若支付成功则发送道具。

public void onFail(String failInfo)，该回调代表用户已放弃支付，无需向服务器查询充值状态。

XMPayParams请使用com.xinmei365.game.proxy.pay路径下的。

• 3.6、退出接口

接入退出接口之前，请务必阅读《退出接口说明》，否则容易导致接错，漏接等。

• 3.6.1、退出

```
        GameProxy.getInstance().exit(Activity activity,  LJExitCallback callback);
```

> ***参数说明：***
>
> *activity 当前Activity callback 退出回调*
>
> ***示例代码：***
>
> ```
>      GameProxy.getInstance().exit(MainActivity.this, new LJExitCallback() {
>          @Override
>          public void onGameExit() {
>          }
>          @Override
>          public void onChannelExit() {
>          }
>      });
> ```
>
> ***接口说明：***
>
> *在游戏退出前调用退出接口，会有如下不同处理：*
> *a）渠道存在退出界面，如91、360等，游戏方只需在onChannelExit()回调中进行退出游戏逻辑即可，无需再弹退出界面*
> *b）渠道不存在退出界面，如百度移动游戏等，游戏方需在onGameExit()回调中弹出自己的游戏退出确认界面，然后再进行退出逻辑，否则会出现渠道审核不通过情况*

- ## 3.6.2、资源回收接口

在游戏退出前，退出接口中调用

```
    GameProxy.getInstance().applicationDestroy(context);
```

- ## 3.7、用户信息扩展接口

```
GameProxy.getInstance().setExtData(Context context, String ext);
```

> ***参数说明：***
>
> *context 上下文Activity，不要使用getApplication()*
> *ext 上传数据（json格式）*
> *上传数据组成：*
> *_id 当前情景，目前支持 enterServer，levelUp，createRole*
> *roleId 当前登录的玩家角色ID，必须为数字，若无，可传入userId*
> *roleName 当前登录的玩家角色名，不能为空，不能为null，若无，传入'游戏名称+username'，如'三国风吹来的鱼"*
> *roleLevel 当前登录的玩家角色等级，必须为数字，且不能为0，若无，传入1*
> *zoneId 当前登录的游戏区服ID，必须为数字，且不能为0，若无，传入1*
> *zoneName 当前登录的游戏区服名称，不能为空，不能为null，若无，传入游戏名称+'1区'，如'刀塔传奇1区"*

balance 当前用户游戏币余额，必须为数字，若无，传入0
vip 当前用户VIP等级，必须为数字，若无，传入1
partyName 当前用户所属帮派，不能为空，不能为null，若无，传入'无帮派"
extra 扩展信息，某些特殊渠道用户信息入口，默认请随便传值，不能为null

示例代码：

```
Map<String,String> datas = new HashMap<String,String>();
datas.put("_id", "enterServer");
datas.put("roleId", "13524696");
datas.put("roleName", "方木");
datas.put("roleLevel", "24");
datas.put("zoneId", "1");
datas.put("zoneName", "墨土1区");
datas.put("balance", "88");
datas.put("vip", "2");
datas.put("partyName", "无尽天涯");
datas.put("extra","extra");
JSONObject obj = new JSONObject(datas);
GameProxy.getInstance().setExtData(this, obj.toString());
```

接口说明：

上传数据接口需在三处调用，分别为进入服务器、玩家创建用户角色、玩家升级，有如下三点需注意：

a）游戏中该接口必须调用三次，传入不同的_id，以区分不同的上传数据；

b）若游戏中无对应接口功能，如游戏中无需创建角色，则可根据自身情况在合适位置进行调用，如登录成功后；

c）在上传游戏数据时，若存在无对应字段值的情况，可传入默认值，具体参见数据组成部分；

3.8、其他接口

使用channelLabel区分渠道：具体参见技术接入文档《如何区分渠道》。

四、混淆

若需要混淆java代码,请勿将SDK代码混淆，可在proguard中进行如下配置： -keep class com.xinmei365.game.proxy.**{*;} -keep class com.alipay.android.app.** {*;} -keep class com.xinmei365.game.proxy.alipay.** {*;}

五、FAQ