易接网游SDK中间件接入使用说明

上海雪鲤鱼计算机科技有限公司

版本	修改日期	作者	修改内容
1.0.0	2014/6/30	刘莹	初始化版本
1.0.1	2014/7/29	李跃年	优化
1.0.2	2014/12/01	刘莹	优化
1.0.3	2015/7/09	田红、吕雪勤、万少锋	增加 LUA 接口； 增加初始化回调的函数； 增加扩展接口； 修改 AndroidManifest.xml 文件的内容； 增加避免模糊的内容
1.0.4	2016/3/07	田红、吕雪勤	增加 setData3 个必接接口

目录

1 引言

1.1 术语及缩略词

appId：游戏的唯一标识，用于区分不同游戏的唯一标准。在易接开发者中心游戏管理模块中创建新游戏获取。

channelId：支付渠道标识，此 id 可区分渠道，在易接后台有相应的渠道对照表。

1.2 适用对象及范围

适用于策划人员、系统设计人员、开发工程师和测试工程师。

2 SDK 概述

2.1 总体描述

易接 Android 网游中间件 SDK 为游戏开发者提供统一调用接口。

易接网游中间件 SDK 接入总体流程图：

易接网游 SDK 中间件接入标准流程

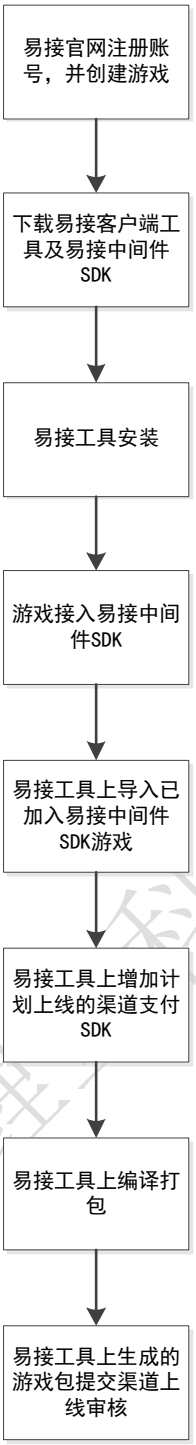


图 2-1

2.1.1. 登陆流程图

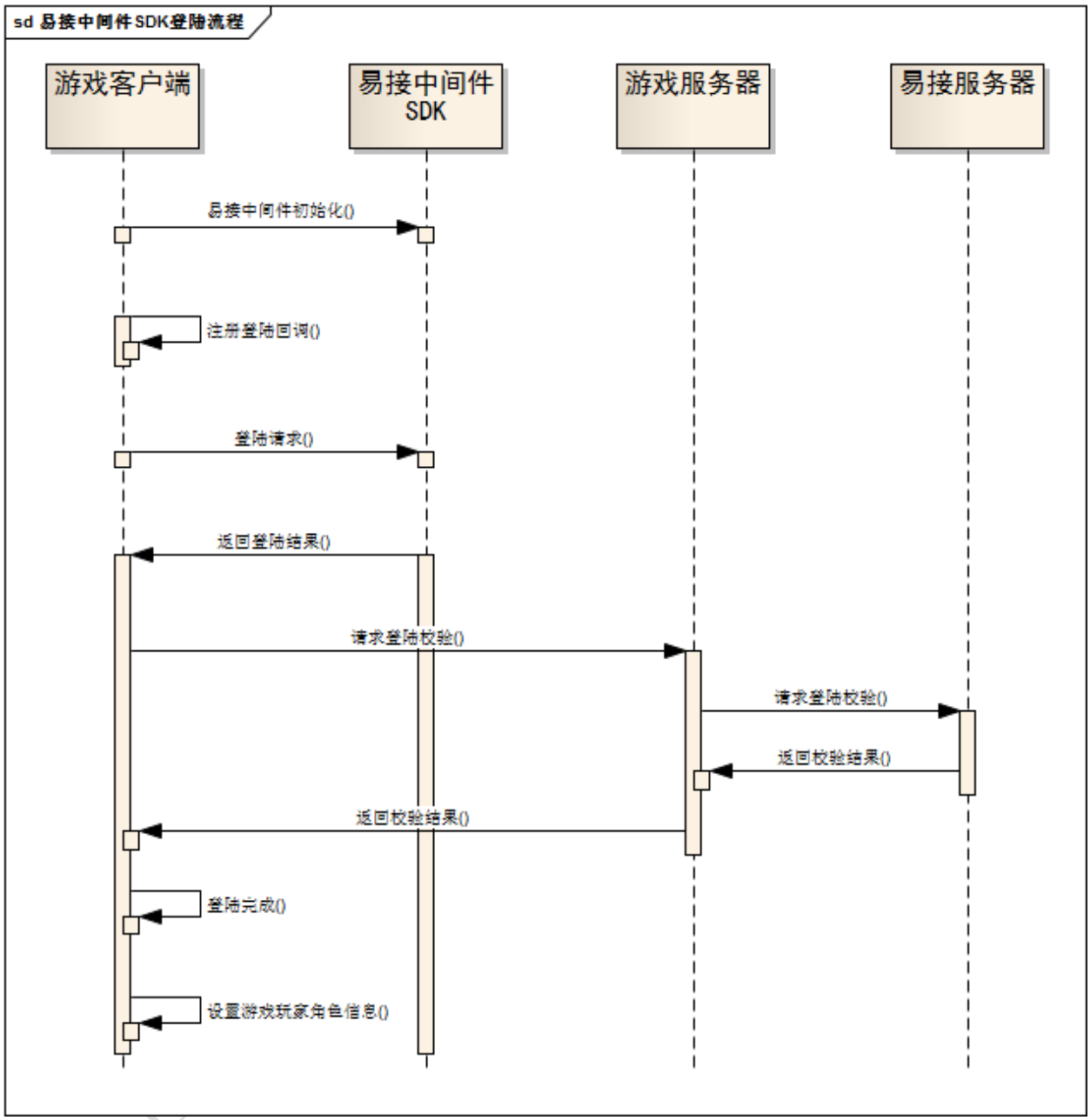


图 2-2

2.1.2 支付流程图

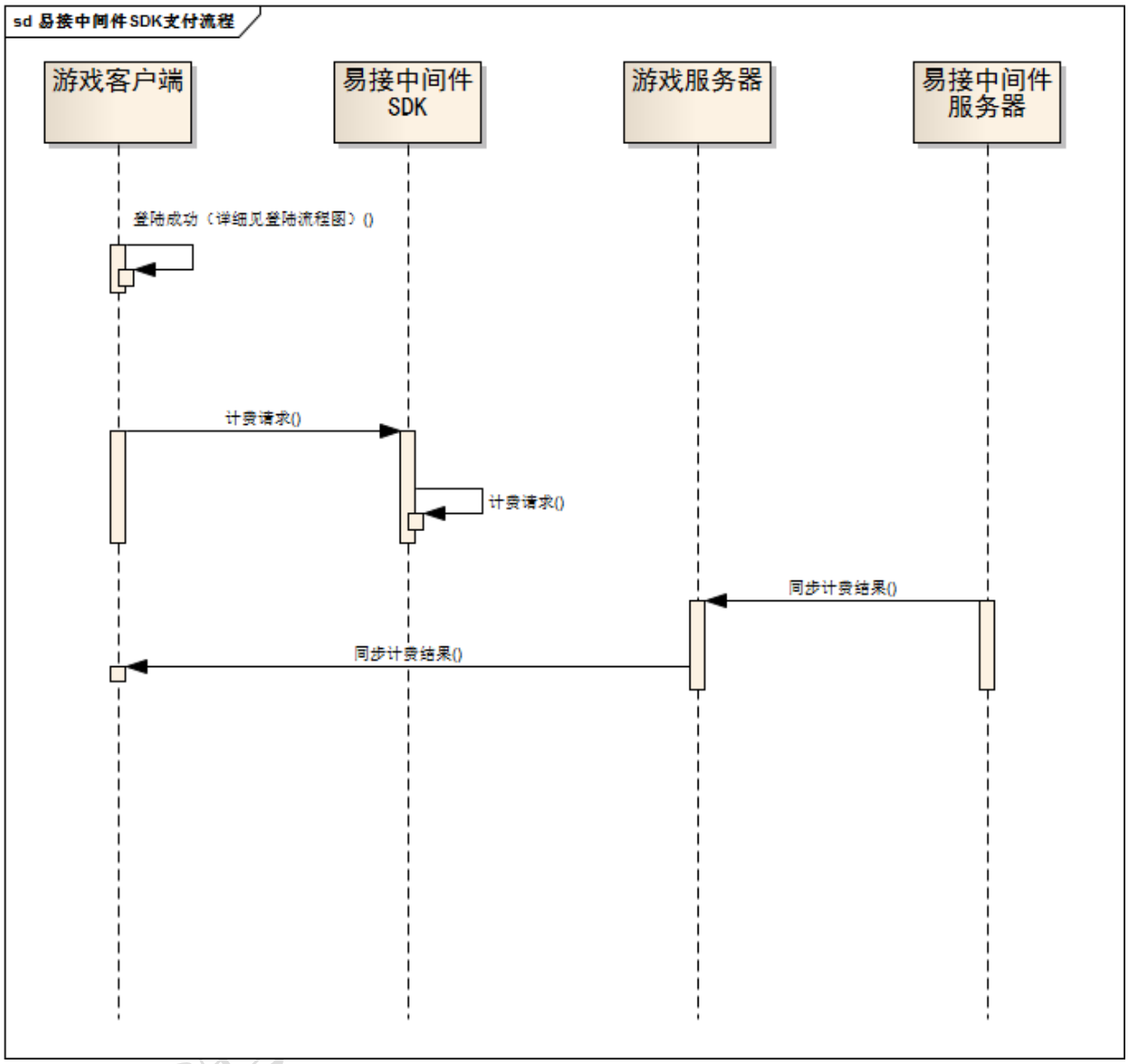


图 2-3

2.2 JAVA 游戏接口

2.2.1 SDK 使用准备

SDK 开发包包括以下几个文件与目录：

SDK 开发资源包：SDK 目录中包含 SDK 的资源文件，请复制 SDK 目录中的

所有目录与文件，并拷贝至各自的游戏工程中，如下：

2.2.1.1 拷贝资源

将\sdk\java \assets 目录下的文件拷贝到游戏对应目录下。

将\sdk\java \lib 目录下的文件拷贝到游戏对应目录下。

2.2.1.2 添加 SDK 提供的 lib 引用

将\sdk\java \libs 目录下的文件拷贝到游戏对应目录下.并做如下图关联。

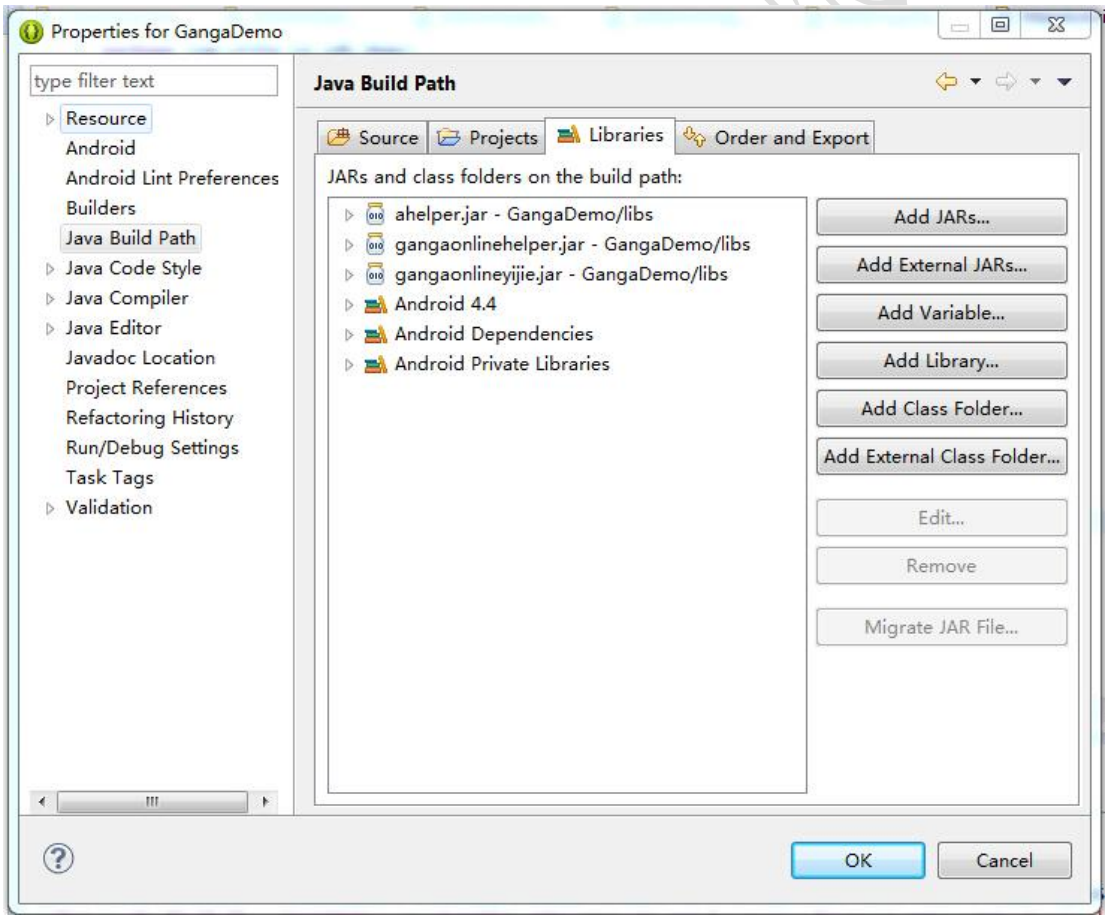


图 2-1

2.2.1.3 添加 SDK 提供的 Activity 引用

在游戏 AndroidManifest.xml 文件的 application 中添加由易接提供如下代码，可参考“网游\sdk\ AndroidManifest.xml "。

```
<service
     android:name="com.snowfish.a.a.s.ABGSvc"
     android:enabled="true"
      android:process="com.snowfish.a.a.bg">
<intent-filter>
     <action android:name="com.snowfish.a.a.s.ABGSvc"/>
     <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
</service>
<meta-data
     android:name="com.snowfish.customer"
     android:value="SNOWFISH">
</meta-data>
<meta-data
     android:name="com.snowfish.channel"
     android:value="SNOWFISH">
</meta-data>
<meta-data
     android:name="com.snowfish.sdk.version"
     android:value="2">
</meta-data>
<meta-data
     android:name="com.snowfish.appid"
     android:value="KEY">
</meta-data>
<meta-data
     android:name="com.snowfish.channelid"
     android:value="{4ff036a1-3254eafe}">
</meta-data>
```

标红处的 *KEY* 值需要替换成在易接用户中心申请获取的 APPID

格式如下：{12345678-12345678}。

2.2.1.4 修改 Application

1）若无自定义Applicatiion，则修改AndroidManifest.xml的Application如下：

```
<application android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2）若开发者自定义的Application。则自定义Application需要继承 *com.snowfish.cn.ganga.helper.SFOnlineApplication* ，AndroidManifest.xml修改如下:

```
<application android:name="自定义 Application"/>
```

2.2.1.5 添加 SDK 需要的 use-permision

在游戏的 AndroidManifest.xml 中添加 use-permision 如下，

可参考“网游\sdk\ AndroidManifest.xml "

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

2.2.1.6 增加闪屏 Activity

继承 com.snowfish.cn.ganga.helper.SFOnlineSplashActivity 并将该 Activity 设置为程序启动时的 Activity。

```
public class MySplashActivity extends SFOnlineSplashActivity {
    public int getBackgroundColor() {
    // 返回闪屏的背景颜色
        return Color.WHITE;
      }
```

```
    @Override
    public void onSplashStop() {
    // 闪屏结束进入游戏
            Intent intent = newIntent(this, MainActivity.class);
            startActivity(intent);
            this.finish();
        }
}
```

AndroidManifest.xml 中关于该 Activity 的声明，要声明为启动 Activity

```
<activity
        android:name="com.snowfish.cn.ganga.helper.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>
```

2.2.1.7 SDK 避免模糊，防止计费时发生异常。

两种方式建议使用第二种方式：

1. proguard-project.txt 文件中增加如下代码：

```
-keep class com.snowfish.** { *; }
-dontwarn  com.unity3d.**
-keep class com.unity3d.**{*; }
```

2. android\sdk\tools\proguard\proguard-android.txt 文件中增加：

```
-keep class com.snowfish.** { *; }
-dontwarn  com.unity3d.**
-keep class com.unity3d.**{*; }
```

2.2.2 接口说明

2.2.2.1 SDK 初始化接口（必选接口）

如下方法在游戏主 Activity 中调用。

易接网游 SDK 中间件接入标准流程

1. **初始化函数，以下 2 个接口任选一个接入**

（1）public static void onCreate(Activity activity)

该方法用于需要在游戏主 Activity 中的 onCreate 中调用，

注意：只需调用一次

调用用例:

```
SFOnlineHelper.onCreate(this);
```

（2）public static void onCreate(Activity activity,final SFOnlineInit Listener initListener)

该方法用于需要在游戏主 Activity 中的 onCreate 中调用，带有初始化完成的回调函数

调用用例:

```
SFOnlineHelper.onCreate(this,new SFOnlineInitListener() {
        @Override
    public void onResponse(String tag, String value) {
        if(tag.equalsIgnoreCase("success")){
           //初始化成功的回调
        }else if(tag.equalsIgnoreCase("fail")){
          //初始化失败的回调，value：如果SDK返回了失败的原因，会给value赋值
        }
        }});
```

2. **public static void onStop(Activity activity)**

该方法在游戏 Activity 中的 onStop 中调用

调用用例:

```
SFOnlineHelper.onStop(this);
```

3. **public static void onDestroy(Activity activity)**

该方法在游戏 Activity 中的 onDestroy 中调用

调用用例:

```
SFOnlineHelper.onDestroy(this);
```

4. **public static void onResume(Activity activity)**

该方法在游戏 Activity 中的 onResume 中调用

调用用例:

```
SFOnlineHelper.onResume(this);
```

5. **public static void onPause (Activity activity)**

该方法在游戏 Activity 中的 onPause 中调用

调用用例: `SFOnlineHelper.onPause (this);`

6. `public static void onRestart (Activity activity)`

该方法在游戏 Activity 中的 onRestart 中调用

调用用例: `SFOnlineHelper.onRestart (this);`

2.2.2.2 用户登陆登出接口（必选接口）

1. `public static void login(Activity context, Object customParams)`

该方法用于登录，调用用例：

```
SFOnlineHelper.login(activity, "Login");
```

2. `public static void logout(Activity context, Object customParams)`

该方法用于登出，调用用例：

```
SFOnlineHelper.logout(activity, "LoginOut");
```

3. `public static void setLoginListener(Activity context,SFOnlineLoginListener listener)`

该方法为登录的监听函数需要在调用 login 函数之前调用，调用用例：

```
SFOnlineHelper.setLoginListener(activity, new SFOnlineLoginListener() {
        @Override
        public void onLoginSuccess(SFOnlineUser user, Object customParams) {
            //登陆成功回调
        }

        @Override
        public void onLoginFailed(String reason, Object customParams) {
            //登陆失败回调
        }

        @Override
        public void onLogout(Object customParams) {
          //登出回调
        }


    });
```

onLoginSuccess登陆成功回调回来的SFOnlineUser参数描述如下，

用于服务器登录校验:

参数	获取方式	类型	注释
appId	user.getProductCode()	String	易接平台创建的游戏 ID，appId
channelId	user.getChannelId()	String	易接平台标示的渠道 SDK ID
userId	user.getChannelUserId()	String	渠道 SDK 标示的用户 ID
token	user.getToken()	String	渠道 SDK 登录完成后的 Session ID。 特别提醒：部分渠道此参数会包含特殊值如‘+’，空格之类的，如直接使用 URL 参数传输到游戏服务器请求校验，请使用 URLEncoder 编码

2.2.2.3 登陆验证接口（必选接口）

由于有些 SDK 要求必须做登录验证，为接入规范，必须接入登录验证，只有登录验证成功才算真正的登录成功。

调用用例：参考 demo 中的

```
public static void LoginCheck (final SFOnlineUser user)
```

2.2.2.4 设置角色基本数据（必选接口）

1. `public static void setRoleData(Context context, String roleId , String roleName, String roleLevel, String zoneId, String zoneName)`

部分渠道如 UC 渠道，要对游戏人物数据进行统计，而且为接入规范，所以为必选接口

调用时间：在游戏登录验证成功后,进入游戏界面时**上报角色信息 时**调用

参数描述:

参数名称	类型	注释
context	Context	上下文 Activity
roleId	String	角色唯一标识

roleName	String	角色名
roleLevel	String	角色的等级
zoneId	String	角色所在区域唯一标识
zoneName	String	角色所在区域名称

2.2.2.5 退出接口（必选接口）

```
public static void exit(
Activity context, SFOnlineExitListener  listener)
```

当游戏退出前必须调用该方法,进行清理工作。如果游戏直接退出,而不调用该方法,可能会出现未知错误,导致程序崩溃，一般游戏在按返回键退出时调用此接口。

调用用例：

```
SFOnlineHelper.exit(context, new SFOnlineExitListener() {
            /*  onSDKExit
             *  @description  当SDK有退出方法及界面，回调该函数
             *  @param bool   SDK是否退出标志位
             */
            @Override
            public void onSDKExit(boolean bool) {
                if (bool){
                    //SDK已经退出，此处可以调用游戏的退出函数
                    }
            }
            /*  onNoExiterProvide
             *  @description  SDK没有退出方法及界面，回调该函数，可在此使用游戏退出界面
             */
            @Override
            public void onNoExiterProvide() {
            }
        });
```

2.2.2.6 定额计费接口（必选接口）

1. **public static void pay (Context context, int unitPrice, String unitName, int count, String callBackInfo, String callBackUrl, SFOnlinePayResultListener payResultListener)**

   该接口用于定额支付的接口函数。

   参数描述:

参数名称	类型	注释
context	Context	上下文 Activity
unitPrice	int	游戏道具价格，单位为人民币分
itemName	String	虚拟货币名称
count	int	用户选择购买道具界面的默认道具数量。（总价为 count*unitPrice）
callBackInfo	String	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	String	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息
payResultListener	SFOnlinePayResultListener	支付回调接口

调用用例:

```
SFOnlineHelper.pay(MainActivity.this, 100, "金币", 1, "购买金币",
        "http://192.168.0.224:8980/omsdk-cp/user/paylog/sync",
        new SFOnlinePayResultListener() {
                    @Override
                    public void onSuccess(String remain) {

                    }
                    @Override
                    public void onFailed(String remain) {
```

```
            }
        });
```

说明：网游的支付结果请以服务器端的同步结果为准，因为某些 SDK 客户端没有支付结果的回调。

2.2.2.7 非定额计费接口（可选接口）

```
1. public static void charge(Context context, String itemName,  int
   unitPrice,int count,String callBackInfo, String
   callBackUrl,SFOnlinePayResultListenerpay ResultListener)
```

该接口用于用户触发计费进行付费行为的入口函数。合作伙伴可以在需要计费的地方调用此接口进行计费。该接口用于非定额计费。

参数描述:

参数名称	类型	注释
context	Context	上下文 Activity
itemName	String	虚拟货币名称
unitPrice	int	游戏道具价格，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量.(总价为 count*unitPrice)
callBackInfo	String	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	String	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息
payResultListener	SFOnlinePayResultListener	支付回调接口

调用用例：

```
SFOnlineHelper.charge(MainActivity.this,"金币", 100, 1, "购买金币",
          "http://192.168.0.224:8980/omsdk-cp/user/paylog/sync",
       new SFOnlinePayResultListener() {
                        @Override
                        publicvoidonSuccess(String remain) {
                        }
                        @Override
                        publicvoidonFailed(String remain) {
                        }
                    });
```

2.2.2.8 扩展接口 1（必选接口）

```
public static void setData(Context context, String key,Object value)
```

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要上报角色信息，为接入规范，所以为必选接口。

接口：SFOnlineHelper.setData(Context context,String key, Object value);

参数描述：

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下

```
JSONObject roleInfo = new JSONObject();
roleInfo.put("roleId", "1");                //当前登录的玩家角色 ID，必须为数字
roleInfo.put("roleName", "猎人");    //当前登录的玩家角色名，不能为空，不能为 null
roleInfo.put("roleLevel", "100");     //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
roleInfo.put("zoneId", "1");              //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
roleInfo.put("zoneName", "阿狸一区");//当前登录的游戏区服名称，不能为空，不能为 null
roleInfo.put("balance", "0");     //用户游戏币余额，必须为数字，若无，传入 0
roleInfo.put("vip", "1");                   //当前用户 VIP 等级，必须为数字，若无，传入 1
roleInfo.put("partyName", "无帮派");//当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”
```

@key 键值定义如下：

key	接口描述	调用示例
createrole	创建新角色时调用	SFOnlineHelper.setData(context,"createrole",roleInfo.toString());
levelup	玩家升级角色时调用	SFOnlineHelper.setData(context,"levelup",roleInfo.toString());

enterServer	选择服务器进入时调用	SFOnlineHelper.setData(context,"enterServer",roleInfo.toString());

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个渠道的“参数填写帮助”

2.2.2.9 扩展接口 2（可选接口）

```
public static String extend(Activity activity,String data,Map<String, Object> callback)
```

扩展接口，有些 SDK， 要求必须接入统计接口或者其它特殊的接口，并且有返回值或者回调的函数，用户可以使用此接口调用，具体可以参考易接工具上的 SDK 的参数填写帮助。

参数说明如下：

```
/**
     * @param activity                上下文Activity
     * @param data                    需要传入的数据
     * @param ,Map<String, Object>    回调函数的HashMap,如果没有可以传null
                                      String:回调函数的索引：
                                      callback1、callback2、callback3........以此类推
     *                                Object：SFExpandListener类型：回调函数的定义
*/
```

调用用例：

```
HashMap callback = new HashMap<String, SFExpandListener>();
        SFExpandListener lis1 = new SFExpandListener() {
            @Override
            public void onResponse(String tag, String value) {

            }
        };
         callback.put("callback1", lis1);
       //  callback.put("callback2", lis2);
        // callback.put("callback3", lis3);

    String r = SFOnlineHelper.extend((Activity) mContext, "isOneKey", callback);
```

2.3 Unity3D 游戏接口

2.3.1 开发前准备

1．将 SDK\game\Unity3D 文件 Copy 到 Assets\Plugins\Android\中；

2．AndroidManifest 的文件：

（1） 在游戏 AndroidManifest.xml 文件的 application 中添加由易接提供如下代码，可参考“网游\sdk\ AndroidManifest.xml "

```
<service
      android:name="com.snowfish.a.a.s.ABGSvc"
      android:enabled="true"
      android:process="com.snowfish.a.a.bg">
<intent-filter>
      <action android:name="com.snowfish.a.a.s.ABGSvc"/>
      <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
</service>
<meta-data
      android:name="com.snowfish.customer"
      android:value="SNOWFISH">
</meta-data>
<meta-data
      android:name="com.snowfish.channel"
      android:value="SNOWFISH">
</meta-data>
<meta-data
      android:name="com.snowfish.sdk.version"
      android:value="2">
</meta-data>
<meta-data
      android:name="com.snowfish.appid"
      android:value="KEY">
</meta-data>
<meta-data
      android:name="com.snowfish.channelid"
      android:value="{4ff036a1-3254eafe}">
</meta-data>
```

标红处的KEY值需要替换成在易接用户中心申请获取的APPID

格式如下：{12345678-12345678}。

（2） 修改 Application

1） 若无自定义Applicatiion，则修改AndroidManifest.xml的Application如下：

```
<application
android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2） 若开发者自定义的Application。则自定义Application需要继承 *com.snowfish.cn.ganga.helper.SFOnlineApplication* ，AndroidManifest.xml修改如下:

```
<application android:name="自定义Application"/>
```

（3） 在游戏的 AndroidManifest.xml 中添加 use-permision 如下，可参考“网游\sdk\ AndroidManifest.xml "

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

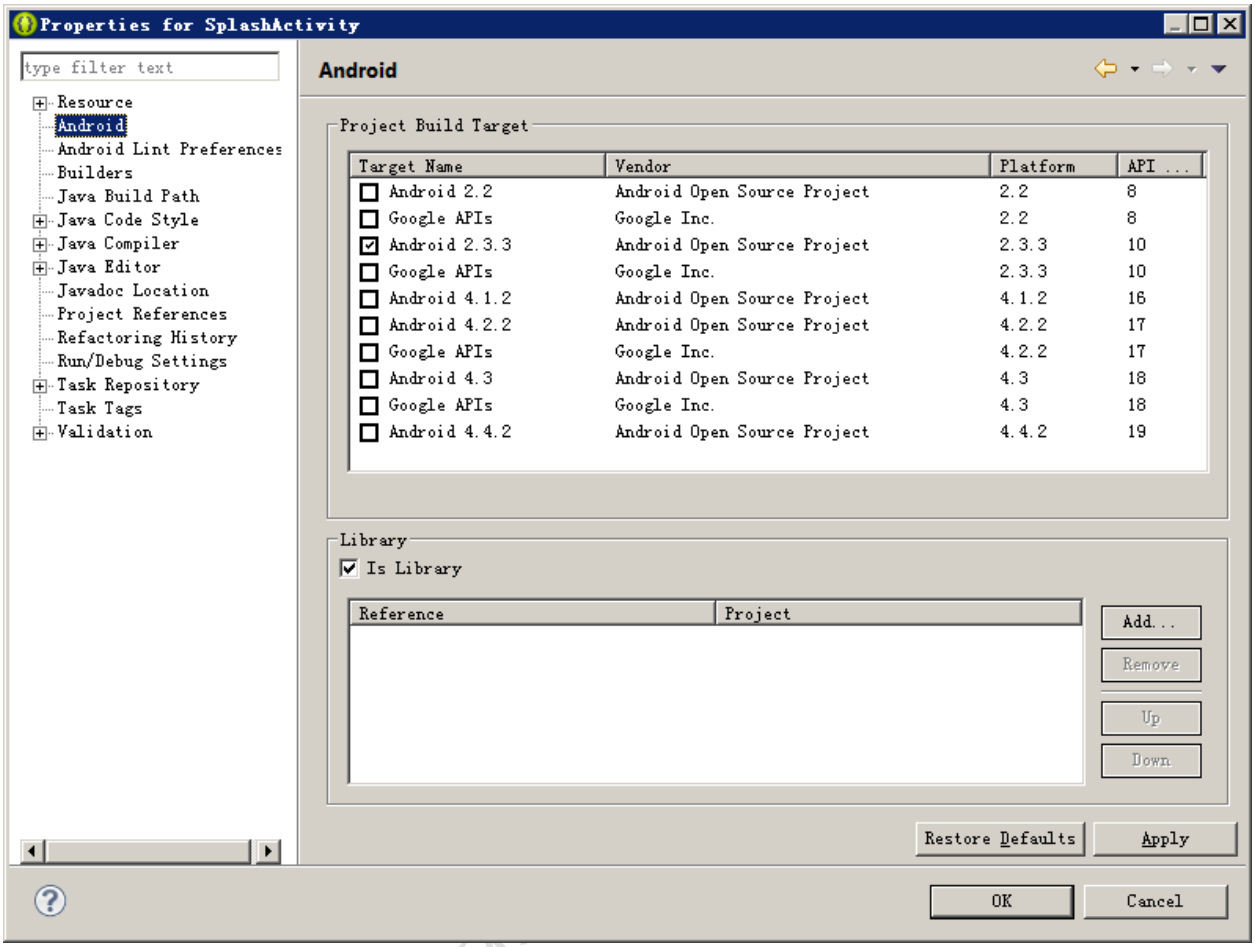
3. 添加启动闪屏 Activitiy 和 Unity3d 游戏入口 Activity

（1） Eclipse 中建立一个与 unity3d 工程同包名工程，引入 gangaonlineyijie.jar 和 gangaonlinehelper.jar 库；

（2） 新建一个闪屏 activity 例如 SplashActivity，继承 com.snowfish.cn.ganga.helper.SFOnlineSplashActivity;（同 2.2.1.6 部分）；

（3） 新建一个游戏入口 activity 例如 MainActivity 继承 Unity3D 的 UnityPlayerActivity 类；

（4） 将工程 properties 的 android 选项中 Is Library 选中，如下图



（5） 将工程 bin 目录下生成的 jar 拷贝至 unity 工程 Assets\Plugins\Android\bin 文件夹下；

（6） 将 Assets\Plugins\Android\AndroidManifest.xml 中的主入口 Activity 配置为 SplashActivity。例如：

```
<activityandroid:name=".SplashActivity">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

（7） 在 Assets\Plugins\Android\AndroidManifest.xml 中声明游戏的入口 MainActivity，如下：

```
<activity
        android:name=".MainActivity"
        android:configChanges="orientation|navigation|screenSize
```

```
                                    |keyboard|keyboardHidden"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
            <meta-data
                android:name="unityplayer.UnityActivity"
                android:value="true" />
            <meta-data
                android:name="unityplayer.ForwardNativeEventsToDalvik"
                android:value="true" >
            </meta-data>
        </activity>
```

（8） 游戏的入口 Activity（例如上述 *MainActivity*）请添加到 Assets\Plugins\Android\res\values\strings.xml 中的 *zy_class_name* 字符串中，使得闪屏完成后即启动游戏,例如：

```
<string
    name="zy_class_name"> com.yijie.cn.sdk.demo.MainActivity
</string>
```

2.3.2 接口说明

2.3.2.1 SDK 初始化及 Activity 生命周期接口（必选接口）

本节方法在游戏入口 Activity 中调用，例如上述 *MainActivity*。

1． 初始化函数，以下2个接口任选一个接入

（1）SFOnlineHelper.*onCreate*(Activity arg0)

该方法在游戏*Activity*中的*onCreate*中调用；调用用例

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    SFOnlineHelper.onCreate(this);
}
```

注意：只需调用一次

（2）public static void onCreate(Activity **activity,**final SFOnlineInitListener **initListener)**

该方法用于需要在游戏主 Activity 中的 onCreate 中调用，带有初始化完成

的回调函数

调用用例:

```
SFOnlineHelper.onCreate(this,new SFOnlineInitListener() {
        @Override
    public void onResponse(String tag, String value) {
        if(tag.equalsIgnoreCase("success")){
           //初始化成功的回调
        }else if(tag.equalsIgnoreCase("fail")){
          //初始化失败的回调，value：如果SDK返回了失败的原因，会给value赋值
        }
        }});
```

2. **SFOnlineHelper.*onStop*(Activity arg0)**

该方法在游戏*Activity*中的*onStop*中调用，调用用例:

```
@Override
protected void onStop() {
    super.onStop();
    SFOnlineHelper.onStop(this);
}
```

3. **SFOnlineHelper.*onDestroy*(Activity arg0);**

该方法在游戏*Activity*中的*onDestroy*中调用，调用用例:

```
@Override
protected void onDestroy() {
    super.onDestroy();
    SFOnlineHelper.onDestroy(this);
 }
```

4. **SFOnlineHelper.*onResume*(Activity arg0);**

该方法在游戏*Activity*中的*onResume*中调用，如果出现Unity3D界面刷新异常可以增加延迟调用，调用用例:

```
@Override
protected void onResume() {
    super.onResume();
    handel.postDelayed(new Runnable() {
        public void run() {
            SFOnlineHelper.onResume(MainActivity.this);
        }
    }, 1000);
 }
```

5. **SFOnlineHelper.*onPause*(Activity arg0);**

该方法在游戏*Activity*中的*onPause*中调用，调用用例：

```
@Override
protected void onPause() {
   super.onPause();
   SFOnlineHelper.onPause(this);
}
```

6. SFOnlineHelper.*onRestart*(Activity arg0);

该方法在游戏*Activity*中的*onRestart*中调用，调用用例：

```
@Override
protected void onRestart() {
   super.onRestart();
   SFOnlineHelper.onRestart(this);
}
```

2.3.2.2 用户登陆登出接口（必选接口）

以下小节所有接口，请在 Unity3D 脚本中使用。

```
1. [DllImport("gangaOnlineUnityHelper")]
privatestatic extern void login(IntPtr context, string customParams);
```

该方法用于登录，调用用例：

```
login(curActivity.GetRawObject(),"Login");
```

```
2. [DllImport("gangaOnlineUnityHelper")]
private static extern void logout(IntPtr context, string customParams);
```

该方法用于登出，调用用例：

```
logout(curActivity.GetRawObject(),"LoginOut");
```

```
3. [DllImport("gangaOnlineUnityHelper")]
private static extern void exit(IntPtr context, string gameObject,
string listener);
```

该方法用于系统全局退出，listener为退出监听函数，调用用例：

```
exit(curActivity.GetRawObject(),"Main Camera"," ExitResult");
```

```
4. [DllImport("gangaOnlineUnityHelper")]
private static extern void setLoginListener (IntPtr context,
```

string gameObject, string listener);

该接口方法用于设置登陆监听;

参数描述:

参数名称	类型	注释
context	IntPtr	currentActivity
gameObject	string	游戏场景中的对象，SDK 内部完成计费逻辑后，并把计费结果通过 Unity 内部 API(com.unity3d.player.UnityPlayer.UnitySendMessage(String gameObject,StringruntimeScriptMethod,Stringargs)通知到 Unity，故游戏开发者需要指定一个游戏对象和该对象的运行脚本，用于侦听 SDK 的计费结果
listener	string	登陆监听函数，隶属于 gameObject 对象的运行时脚本的方法名称，该方法会在收到通知后触发

调用用例

```
setLoginListener (curActivity.GetRawObject(),"Main Camera", "LoginResult");
```

2.3.2.3 定额计费接口（必选接口）

```
1. [DllImport("gangaOnlineUnityHelper")]
private static extern void pay(IntPtr context, string gameObject,
intunitPrice, string unitName, int count, string callBackInfo, string callBac
kUrl, string payResultListener);
```

该接口用于定额支付的接口函数。合作伙伴在需要使用定额支付的时候使用该接口。

参数描述:

参数名称	类型	注释
context	IntPtr	上下文 Activity
unitName	string	虚拟货币名称
unitPrice	int	当前 1 个虚拟货币需要支付金额，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量。（总价

		count*unitPrice）
callBackInfo	string	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	string	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息
gameObject	string	游戏场景中的对象，SDK 内部完成计费逻辑后，并把计费结果通过 Unity 内部 API(com.unity3d.player.UnityPlayer.UnitySendMessage(String gameObject,StringruntimeScriptMethod,Stringargs)通知到 Unity，故游戏开发者需要指定一个游戏对象和该对象的运行脚本，用于侦听 SDK 的计费结果。
payResultListener	string	支付监听函数，隶属于 gameObject 对象的运行时脚本的方法名称，该方法会在收到通知后触发

调用用例：

```
pay(curActivity.GetRawObject(), "Main Camera", 100,  "100金币", 1, "购买金币",
"http://192.168.0.224:8980/omsdk-cp/user/paylog/sync", "PayResult");
```

2.3.2.4 非定额计费接口（可选接口）

```
1. [DllImport("gangaOnlineUnityHelper")]
private static extern void charge(IntPtr context, string gameObject, string
                itemName, intunitPrice, int count, string callBackInfo,
                string callBackUrl, string payResultListener);
```

该接口用于用户触发计费进行付费行为的入口函数。合作伙伴可以在需要计

费的地方调用此接口进行计费。该接口用于非定额计费。

参数描述：

参数名称	类型	注释
context	IntPtr	上下文 Activity
itemName	string	虚拟货币名称
unitPrice	int	游戏道具单位价格，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量.(count*unitPrice)
callBackInfo	string	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	string	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息
gameObject	string	游戏场景中的对象，SDK 内部完成计费逻辑后，并把计费结果通过 Unity 内部 API(com.unity3d.player.UnityPlayer.UnitySendMessage(String gameObject,StringruntimeScriptMethod,Stringargs)通知到 Unity，故游戏开发者需要指定一个游戏对象和该对象的运行脚本，用于侦听 SDK 的计费结果。
payResultListener	string	支付监听函数，隶属于 gameObject 对象的运行时脚本的方法名称，该方法会在收到通知后触发

调用用例：

```
charge(curActivity.GetRawObject(), "Main Camera", "200金币", 200, 1, "购买金币", "http://192.168.0.224:8980/omsdk-cp/user/paylog/sync", "PayResult");
```

2.3.2.5 设置角色基本数据（必选接口）

```
1. [DllImport("gangaOnlineUnityHelper")]
private static extern void setRoleData(IntPtr context, string roleId,
stringroleName, string roleLevel, string zoneId, string zoneName);
```

部分渠道如UC渠道，要对游戏人物数据进行统计，而且为接入规范，

调用时间：在游戏登录验证成功后

参数描述：

参数名称	类型	注释
context	IntPtr	上下文 Activity
roleId	string	角色唯一标识
roleName	string	角色名
roleLevel	string	角色等级
zoneId	string	区域唯一标识
zoneName	string	区域名称

2.3.2.6 扩展接口 1（必选接口）

```
1.[DllImport("gangaOnlineUnityHelper")]
  private static extern void setData (IntPtr context, string key, string value);
```

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要上报角色信息，为接入规范，所以为必选接口。

参数描述：

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下

```
SFJSONObject    roleInfo = new SFJSONObject ();
roleInfo.put("roleId", "1");              //当前登录的玩家角色 ID，必须为数字
roleInfo.put("roleName", "猎人");   //当前登录的玩家角色名，不能为空，不能为 null
roleInfo.put("roleLevel", "100");     //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
roleInfo.put("zoneId", "1");              //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
```

```
roleInfo.put("zoneName", "阿狸一区");//当前登录的游戏区服名称，不能为空，不能为 null
roleInfo.put("balance", "0");    //用户游戏币余额，必须为数字，若无，传入 0
roleInfo.put("vip", "1");                //当前用户 VIP 等级，必须为数字，若无，传入 1
roleInfo.put("partyName", "无帮派");//当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”
```

@key 键值定义如下：

key	接口描述	调用示例
createrole	创建新角色时调用	setData(curActivity.GetRawObject(),"createrole",roleInfo.toString());
levelup	玩家升级角色时调用	setData(curActivity.GetRawObject (),"levelup",roleInfo.toString());
enterServer	选择服务器进入时调用	setData(curActivity.GetRawObject(),"enterServer",roleInfo.toString());

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个渠道的“参数填写帮助”

2.3.2.7 扩展接口 2（可选接口）

```
1. [DllImport("gangaOnlineUnityHelper")]
   private static extern void extend (IntPtr context, string data,string gameObject, string listener);
```

扩展接口，有些 SDK， 要求必须接入统计接口或者其它特殊的接口，并且有返回值或者回调的函数，用户可以使用此接口调用，具体可以参考易接工具上的 SDK 的参数填写帮助。调用用例：

```
SFJSONObject temp =new SFJSONObject();
// callbackCount:扩展接口回调函数的数量，没有可以传“0”
temp.put("callbackcount","1");
temp.put("callback1","callback1");
string ss = temp.toInlineString();
SFJSONObject temp1 =new SFJSONObject();
temp1.put("callbackmap",ss);
// extendcallback:扩展接口返回信息，如果不需要返回值可以不加。
temp1.put("extendcallback","extendCallback");
string ll = temp1.toString();
extend (curActivity.GetRawObject(),data, "Main Camera", ll);
```

详细代码请参考Unity3DDemo。

2.3.2.8 回调消息定义

回调采用 Json 方式传递参数，定义如下：

1．登陆回调：

回调返回的Json键值定义对应表：

Key	值类型	注释
"result"	string	登出
"customParams"	string	自定义参数
"userinfo"	SFJSONObject	登陆账户信息

键值定义对应表 2.3.2.7.1

键值"result"，Value值定义如下表：

名称	值	注释
LoginResult.LOGOUT	"0"	登出
LoginResult.LOGIN_SUCCESS	"1"	登入成功
LoginResult.LOGIN_FAILED	"2"	登入失败

键值"result"值定义对应表 2.3.2.7.2

键值" userinfo "， Value值定义如下表：

参数	键值	类型	注释
id	"id"	long	易接内部 userid，该值可能为 0，请不要以此参数作为判定。
channelId	"channelid"	String	易接平台标示的渠道 SDK ID，
ChannelUserId	"channeluserid"	String	渠道 SDK 标示的用户 ID。
UserName	"username"		渠道 SDK 的用户名称。
Token	"token"	String	渠道 SDK 登录完成后的 Session ID。特别提醒：部分渠道此参数会包含特殊值如‘+’，空格之类的，如直接使用 URL 参数传输到游戏服务器请求校验，请使用 URLEncoder 编码。
ProductCode	"productcode"	string	易接平台创建的游戏 ID，appId

键值" userinfo "值定义对应表 2.3.2.7.2

调用用例：

```
// 登陆监听函数
void LoginResult (string result)
{

    Debug.Log ("------------loginResult=" + result);
    SFJSONObject sfjson = new SFJSONObject (result);
    string type = (string)sfjson.get ("result");
    string customParams = (string)sfjson.get ("customParams");
    if (APaymentHelper.LoginResult.LOGOUT == type) {
        str = "login result = logout" + customParams;
        user = null;
        if (!isDebug) {
            bLogined = false;
        }
        using (AndroidJavaClass unityPlayer = new AndroidJavaClass ("com.unity3d.player.UnityPlayer")) {
            using (AndroidJavaObject curActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity")) {
                if (bLogined) {
                    return;
                }
                login (curActivity.GetRawObject (), "Login");

            }
        }
    } else if (APaymentHelper.LoginResult.LOGIN_SUCCESS == type) {
        SFJSONObject userinfo = (SFJSONObject)sfjson.get ("userinfo");
        if (userinfo != null) {
            long id = long.Parse ((string)userinfo.get ("id"));
            string channelId = (string)userinfo.get ("channelid");
            string ChannelUserId = (string)userinfo.get ("channeluserid");
            string UserName = (string)userinfo.get ("username");
            string Token = (string)userinfo.get ("token");
            string ProductCode = (string)userinfo.get ("productcode");
            user = new SFOnlineUser (id, channelId, ChannelUserId,
                                                    UserName, Token, ProductCode);
            Debug.Log ("## id:" + id + " channelId:" + channelId + " ChannelUserId:" + ChannelUserId
                + " UserName:" + UserName + " Token:" + Token + " ProductCode:" + ProductCode);
        }

        str = "login result = login success" + customParams;
        LoginCheck ();
    } else if (APaymentHelper.LoginResult.LOGIN_FAILED == type) {
        str = "login result = login failed" + customParams;
    }
}
```

详细代码请参考Unity3DDemo。

2. 支付回调：

名称	值	注释
PayResult. PAY_SUCCESS	"0"	支付成功
PayResult. PAY_FAILURE	"1"	支付失败
PayResult. PAY_ORDER_NO	"2"	返回支付订单号

使用方法参见 unity3DDemo。

3. exit回调

名称	值	注释
ExitResult.SDKEXIT	"0"	Exit
ExitResult.SDKEXIT_NO_PROVIDE	"1"	SDKEXIT_NO_PROVIDE

使用方法参见unity3DDemo。

4. extend回调

回调返回的Json键值定义对应表：

Key	值类型	注释
"tag"	string	“success”成功，”fail” 失败
“value”	string	失败带回的信息。
"customParams"	string	自定义参数

使用方法参见unity3DDemo。

2.4 Flash air 游戏接口

2.4.1 开发前准备

1. 将 sdk\flashAir\assets 目录下的文件 Copy 到 assets 中；
2. 把 sdk\flashAir\libs\ganga.android.ane 拷贝到游戏项目的 libs 目录下；
3. 在项目属性的 Flex Build Path - Native Extension 下，添加对 ganga.android.ane 的引用；
4. 在项目属性的 Flex Build Packaging - Google Android - Native Extensions 下，在 Package 列为 ganga.android.ane 打上勾；
5. 在游戏项目的 xxx-app.xml 里添加 SDK 提供的 Activity 引用，可参考“网游\sdk\ AndroidManifest.xml "

```
<service
        android:name="com.snowfish.a.a.s.ABGSvc"
        android:enabled="true"
        android:process="com.snowfish.a.a.bg">
    <intent-filter>
        <action android:name="com.snowfish.a.a.s.ABGSvc"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</service>
<meta-data
        android:name="com.snowfish.customer"
        android:value="SNOWFISH">
</meta-data>
<meta-data
        android:name="com.snowfish.channel"
```

```
        android:value="SNOWFISH">
</meta-data>
<meta-data
        android:name="com.snowfish.sdk.version"
        android:value="2">
</meta-data>
<meta-data
        android:name="com.snowfish.appid"
        android:value="KEY">
</meta-data>
<meta-data
        android:name="com.snowfish.channelid"
        android:value="{4ff036a1-3254eafe}">
</meta-data>
```

标红处的 *KEY* 值需要替换成在易接用户中心申请获取的 APPID

如：{12345678-12345678}。

6. 修改 Application

1）若无自定义Applicatiion，则修改AndroidManifest.xml的Application如下：

```
<application
android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2）若开发者自定义的Application。则自定义Application需要继承 *com.snowfish.cn.ganga.helper.SFOnlineApplication* ，AndroidManifest.xml修改如下:

```
<application android:name="自定义 Application"/>
```

7. 在游戏的 AndroidManifest.xml 中添加 use-permision 如下，

可参考“网游\sdk\ AndroidManifest.xml "

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

```
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

8. 在游戏项目的 xxx-app.xml 里添加

```
<extensions>
<extensionID>
      cn.ganga.ane.android
</extensionID>
</extensions>
```

2.4.2 接口说明

2.4.2.1 SDK 初始化接口 （必选接口）

以下 2 个接口任选一个接入，注意：只需调用一次

1、`public function initSDK():void`

该方法用于SDK的初始化，调用用例：

```
SFOnlineHelper.getInstance().initSDK();
SFOnlineHelper.getInstance().onResume();
NativeApplication.nativeApplication.addEventListener(Event.EXITING, onDestroy);
NativeApplication.nativeApplication.addEventListener(Event.ACTIVATE,onActivate);
NativeApplication.nativeApplication.addEventListener(Event.DEACTIVATE,onDeActivat
e );

    SFOnlineHelper.getInstance().addEventListener(Constants.EVENT_TYPE_GANGA_CALL
BACK,onCallbackEvent);
    if(stage)//AIR内置对象
    { // 添加切换到后台的监听
        stage.addEventListener(ThrottleEvent.THROTTLE,pauseHandler);
     //添加对back键的处理
        stage.addEventListener(KeyboardEvent.KEY_DOWN, onKey);
    }
```

onCallbackEvent函数是对登陆、支付的回调的处理，具体用法参考demo。

2、`public function initSDKExtend():void`

该方法用于 SDK 的初始化，带有初始化完成的回调函数

调用用例：

```
SFOnlineHelper.getInstance().initSDKExtend();
SFOnlineHelper.getInstance().onResume();
NativeApplication.nativeApplication.addEventListener(Event.EXITING, onDestroy);
NativeApplication.nativeApplication.addEventListener(Event.ACTIVATE,onActivate);
NativeApplication.nativeApplication.addEventListener(Event.DEACTIVATE,onDeActivat
e );

    SFOnlineHelper.getInstance().addEventListener(Constants.EVENT_TYPE_GANGA_CALL
BACK,onCallbackEvent);
    if(stage)//AIR内置对象
    { // 添加切换到后台的监听
        stage.addEventListener(ThrottleEvent.THROTTLE,pauseHandler);
    //添加对back键的处理
        stage.addEventListener(KeyboardEvent.KEY_DOWN, onKey);
    }
```

onCallbackEvent函数是对初始化、登陆、支付的回调的处理，具体用法参考demo。

2.4.2.2 用户登陆接口（必选接口）

```
public function login(type:String):void
```

该方法用于登录，调用用例：

```
SFOnlineHelper.getInstance().login("Login");
```

2.4.2.3 用户登陆验证（必选接口）

由于有些 SDK 要求必须做登录验证，为接入规范，必须接入登录验证，只有登录验证成功才算真正的登录成功。

参考demo中的

```
protected function btnLoginCheck_clickHandler(event:MouseEvent):void
```

2.4.2.4 用户登出接口（可选接口）

```
public function logout():void
```

该方法为登出函数，调用用例：

```
SFOnlineHelper.getInstance().logout();
```

2.4.2.5 退出接口 （必选接口）

```
public function exitSDK():void
```

当游戏退出前必须调用该方法,进行清理工作。如果游戏直接退出,而不调用该方法,可能会出现未知错误,导致程序崩溃，一般游戏在按返回键退出时调用此接口。

调用用例：

```
SFOnlineHelper.getInstance().exitSDK();
```

2.4.2.6 设置角色基本数据（必选接口）

```
public function setRoleData(roleId:    String,
                            roleName:  String,
                            roleLevel: String,
                            zoneId:    String,
                            zoneName:  String):void
```

部分渠道如 UC 渠道，要对游戏人物数据进行统计，而且为接入规范，所以为必选接口

调用时间：在游戏登录验证成功后

参数描述：

参数名称	类型	注释
roleId	String	角色唯一标识
roleName	String	角色名
roleLevel	String	角色等级
zoneId	String	区域唯一标识

zoneName	String	区域名称

调用用例：

```
SFOnlineHelper.getInstance().setRoleData("1", "猎人", "100", "1", "阿狸一区");
```

2.4.2.7 定额计费接口（必选接口）

```
public function pay(unitPrice: int,
                    unitName: String,
                    count: int,
                    callBackInfo: String,
                    callBackUrl: String):void
```

该接口用于定额支付的接口函数。合作伙伴在需要使用定额支付的时候使用该接口。

参数描述：

参数名称	类型	注释
itemName	String	虚拟货币名称
unitPrice	int	游戏道具单位价格，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量（总价为 count*unitPrice）
callBackInfo	String	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	String	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息

调用用例：

```
SFOnlineHelper.getInstance().pay (100, "金币", 1, "购买金币",
"http://192.168.0.224:8980/omsdk-cp/user/paylog/sync");
```

说明：网游的支付结果请以服务器端的同步结果为准，因为某些 SDK 客户端没有支付结果的回调。

2.4.2.8 非定额计费接口（可选接口）

```
public function charge(itemName:String,
                       unitPrice: int,
                       count:int,
                       callBackInfo:String,callBackUrl:String):void
```

该接口用于用户触发计费进行付费行为的入口函数，该接口用于非定额计费。

参数描述：

参数名称	类型	注释
itemName	String	虚拟货币名称
unitPrice	int	游戏道具单位价格，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量（总价为 count*unitPrice）
callBackInfo	String	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	String	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息

调用用例：

```
 SFOnlineHelper.getInstance().charge("金币", 100, 2, "购买金币",
"http://192.168.0.224:8980/omsdk-cp/user/paylog/sync"）;
```

2.4.2.9 扩展接口 1（必选接口）

```
public function setData(key: String, value:Object):void
```

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要上报角色信息，为接入规范，所以为必选接口。

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下：

roleId :当前登录的玩家角色 ID，必须为数字
roleName :当前登录的玩家角色名，不能为空，不能为 null
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
balance :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 :当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”

@key 键值定义和调用实例如下：

```
var value : Object = JSON.stringify({"roleId":"1","roleName":"猎人
","roleLevel":"100","zoneId":"1","zoneName":"阿狸一区
","balance":"0","vip":"1","partyName":"无帮派"},null,null);
```

key	接口描述	调用示例
createrole	创建新角色时调用	SFOnlineHelper.getInstance(). setData("createrole" , value);
levelup	玩家升级角色时调用	SFOnlineHelper.getInstance(). setData("levelup" , value);
enterServer	选择服务器进入时调用	SFOnlineHelper.getInstance(). setData("enterServer" , value);

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个渠道的“参数填写帮助”

2.4.2.10 扩展接口 2（可选接口）

```
public function extend(data:String,callbackCount:String):void
```

扩展接口，有些 SDK， 要求必须接入统计接口或者其它特殊的接口，并且有

返回值或者回调的函数，用户可以使用此接口调用，具体可以参考易接工具上的 SDK 的参数填写帮助。

参数说明：
```
/** 扩展接口
 * data:需要传入的数据
 * callbackCount:扩展接口回调函数的数量，没有可以传“0”
 */
```

调用用例：

```
SFOnlineHelper.getInstance().extend ("isOneKey","3");
```

2.4.2.11 获取 Meta 信息接口（可选接口）

```
public function getMetaData (key: String):void
```

该接口用于获取android中Meta信息接口，结果在onCallbackEvent中获取

调用用例：

```
SFOnlineHelper.getInstance().getMetaData("com.snowfish.appid");
```

2.4.2.12 回调消息处理（必选接口）

参考 Demo 中的

```
public function onCallbackEvent(event:CallbackEvent):void
```

返回类型： var callbackType:String = event.callbackType;

返回数据： var data:Object = event.data;

2.4.2.13.1 初始化回调

名称	注释
Constants. CALLBACKTYPE_InitResponse	初始化结果返回

2.4.2.13.2 登录回调

名称	注释

Constants. CALLBACKTYPE_LoginSuccess	登录成功（登录成功后做登录校验，只有登录校验成功才算真正的登录成功）
Constants. CALLBACKTYPE_LoginFailed	登录失败
Constants. CALLBACKTYPE_Logout	登出

2.4.2.13.3 登录验证结果返回

名称	注释
Constants. CALLBACKTYPE_ExecuteHttpGet	登录验证结果返回

调用以下函数后会返回登录验证的结果

```
SFOnlineHelper.getInstance().executeHttpGet(url);
```

2.4.2.13.4 支付回调

名称	注释
Constants. CALLBACKTYPE_PaySuccess	支付成功
Constants. CALLBACKTYPE_PayFailed	支付失败
Constants. CALLBACKTYPE_PayonOderNo	返回支付订单号

2.4.2.13.5 exit 回调

名称	注释
Constants. CALLBACKTYE_Exit	Exit

退出类型的说明：

```
if(exitType == "onSDKExitSuccess"){
    //当SDK有退出方法及界面，退出成功，此处可以调用游戏的退出函数

 }else if(exitType == "onSDKExitFail") {
    //当SDK有退出方法及界面，退出失败调用该函数

 }else if(exitType == "onNoExiterProvide"){
    //SDK没有退出方法及界面，回调该函数，可在此使用游戏退出界面
 }
```

```
            }
        }
```

2.4.2.13.6 扩展接口 2 的返回值

名称	注释
Constants. CALLBACKTYPE_Expand_Return	extend 返回结果

2.4.2.13.7 扩展接口 2 的回调处理

名称	注释
Constants. CALLBACKTYPE_Expand_Calback	extend 的回调处理

2.4.2.13.8 获取 Meta 信息回调

名称	注释
Constants.CALLBACKTYPE_GetMetaData	GetMetaData 返回结果

2.5 C++游戏接口

2.5.1 开发前准备

2.5.1.1 拷贝资源

SDK 开发资源包：SDK 目录中包含 SDK 的资源文件，请复制 SDK 目录中的所有目录与文件，并拷贝至各自的游戏工程中，如下：

a)将\sdk\ cocos \assets 目录下的文件拷贝到游戏对应目录下(例如 cocos 有些版本将资源放在 Resources);

b)将\sdk\ cocos \ libs 目录下的文件拷贝到游戏对应目录下;

c)将\sdk\cocos\libs\armabi\libgangaOnlineUnityHelper.so 拷贝到游戏项目的 libs\armabi 目录和 jni 目录下；

d)将\sdk\cocos\Classes\SFGameNativeInterface.hpp 文件中拷贝到游戏工程中 C++头文件存放位置，如 Cocos2dx 项目放入 Classes 目录，并且在使用接口文件中加入 include 引用：`#include "SFGameNativeInterface.hpp"`。

2.5.1.2 添加 SDK 提供的 lib 引用

将 game\libs 目录下的文件拷贝到游戏对应目录下.并做如下图关联。

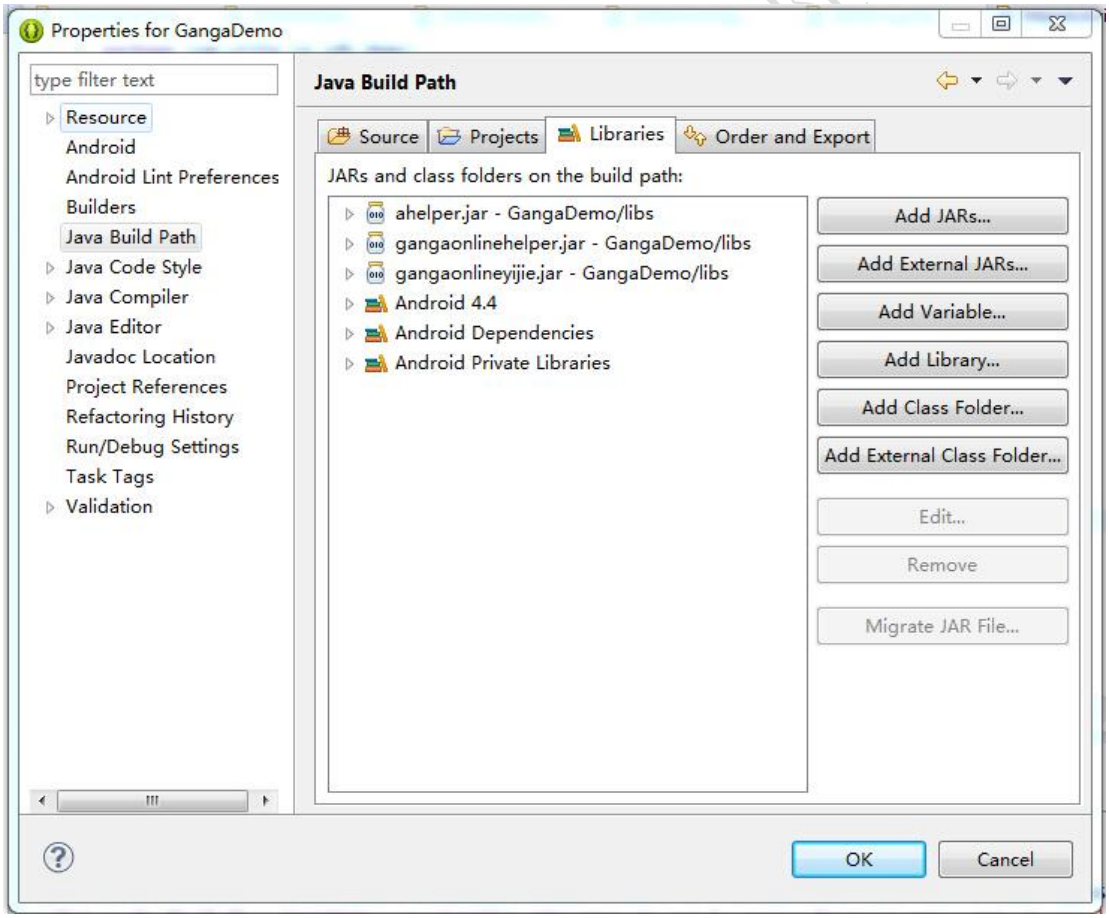


图 2-1

2.5.1.3 添加 SDK 提供的 Activity 引用

在游戏 AndroidManifest.xml 文件的 application 中添加由易接提供如下代码，

可参考“网游\sdk\ AndroidManifest.xml "

```
<service
        android:name="com.snowfish.a.a.s.ABGSvc"
        android:enabled="true"
        android:process="com.snowfish.a.a.bg">
<intent-filter>
        <action android:name="com.snowfish.a.a.s.ABGSvc"/>
        <category android:name="android.intent.category.DEFAULT"/>
</intent-filter>
</service>
<meta-data
        android:name="com.snowfish.customer"
        android:value="SNOWFISH">
</meta-data>
<meta-data
        android:name="com.snowfish.channel"
        android:value="SNOWFISH">
</meta-data>
<meta-data
        android:name="com.snowfish.sdk.version"
        android:value="2">
</meta-data>
<meta-data
        android:name="com.snowfish.appid"
        android:value="KEY">
</meta-data>
<meta-data
        android:name="com.snowfish.channelid"
        android:value="{4ff036a1-3254eafe}">
</meta-data>
```

标红处的 *KEY* 值是在易接用户中心申请获取的 APPID

格式如下：{12345678-12345678}。

2.5.1.4 修改 Application

1）若无自定义Applicatiion，则修改AndroidManifest.xml的Application如下：

```
<application
android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2）若开发者自定义的Application。则自定义Application需要继承 *com.snowfish.cn.ganga.helper.SFOnlineApplication* ，AndroidManifest.xml修改如下:

```
<application android:name="自定义 Application"/>
```

2.5.1.5 添加 SDK 需要的 use-permision

在游戏的 AndroidManifest.xml 中添加 use-permision 如下，

可参考“网游\sdk\ AndroidManifest.xml "

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

2.5.1.6 增加闪屏 Activity

继承 com.snowfish.cn.ganga.helper.SFOnlineSplashActivity 并将该 Activity 设置为程序启动时的 Activity。

```
public class MySplashActivity extends SFOnlineSplashActivity {
    public int getBackgroundColor() {
    // 返回闪屏的背景颜色
        returnColor.WHITE;
      }
    @Override
    public void onSplashStop() {
```

```
    // 闪屏结束进入游戏
            Intent intent = newIntent(this, MainActivity.class);
            startActivity(intent);
            this.finish();
        }
}
```

AndroidManifest.xml 中关于该 Activity 的声明，要声明为启动 Activity

```
<activity
        android:name="com.snowfish.cn.ganga.helper.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>
```

2.5.1.7 SDK 避免模糊，防止计费时发生异常。

两种方式建议使用第二种方式：

1、proguard-project.txt 文件中增加如下代码：

```
-keep class com.snowfish.** { *; }
-dontwarn  com.unity3d.**
-keep class com.unity3d.**{*; }
```

2、android\sdk\tools\proguard\proguard-android.txt 文件中增加：

```
-keep class com.snowfish.** { *; }
-dontwarn  com.unity3d.**
-keep class com.unity3d.**{*; }
```

2.5.1.8 Android.mk 文件修改

修改 android 工程的 jni/Android.mk 文件，添加如下代码(黄色突出代码):

```
LOCAL_PATH := $(call my-dir)
//yijie SDK add -s
include $(CLEAR_VARS)
LOCAL_MODULE := gangaOnlineUnityHelper
LOCAL_SRC_FILES := libgangaOnlineUnityHelper.so
```

```
include $(PREBUILT_SHARED_LIBRARY)
//yijie SDK add -e

include $(CLEAR_VARS)
LOCAL_MODULE := cocos2dcpp_shared

....

LOCAL_WHOLE_STATIC_LIBRARIES := cocos2dx_static
LOCAL_WHOLE_STATIC_LIBRARIES += cocosdenshion_static
//yijie SDK add -s
LOCAL_SHARED_LIBRARIES += gangaOnlineUnityHelper
//yijie SDK add -e
```

2.5.2 接口说明

2.5.2.1 SDK 初始化接口 （必选接口）

一、C++初始化接口

游戏主 Activity 中加载库文件，如需要填在其他库文件，请将此库文件放在第一个加载:

```
static {
    System.loadLibrary("gangaOnlineUnityHelper");
}
```

添加 jni 初始化接口，在游戏主 Activity 的 onCreate 中调用

以下 2 个接口任选一个接入，注意：只需调用一次

(1)初始化无回调方法

```
SFNativeAdapter.init(AppActivity.this, new SFActionCallback() {
        @Override
        public void callback(Runnable run) {
            runOnGLThread(run);
        }
    });
```

（2）初始化带有回调函数（在游戏里注册初始化回调）

```
SFNativeAdapter.init_listener(AppActivity.this, new SFActionCallback() {
        @Override
        public void callback(Runnable run) {
```

```
            runOnGLThread(run);
        }
    });
```

注： cocos2.0 版本中没有 runOnGLThread 此方法，需要自行添加。参考如下：

```
public void runOnGLThread(final Runnable pRunnable) {
 this.mGLSurfaceView.queueEvent(pRunnable);// mGLSurfaceView基类为GLSurfaceView
 }
```

二、游戏主 Activity 初始化接口（必选接口）

如下方法在游戏主 Activity 中调用。

1、public static void onCreate(Activity activity)

该方法用于需要在游戏主 Activity 中的 onCreate 中调用，只需调用一次

注：如果调用初始化带有回调的方法，此方法就不需要调用

调用用例： `SFOnlineHelper.onCreate(this);`

2、public static void onStop(Activity activity)

该方法在游戏 Activity 中的 onStop 中调用

调用用例： `SFOnlineHelper.onStop(this);`

3、public static void onDestroy(Activity activity)

该方法在游戏 Activity 中的 onDestroy 中调用

调用用例： `SFOnlineHelper.onDestroy(this);`

4、public static void onResume(Activity activity)

该方法在游戏 Activity 中的 onResume 中调用

调用用例： `SFOnlineHelper.onResume(this);`

5、public static void onPause (Activity activity)

该方法在游戏 Activity 中的 onPause 中调用

调用用例： `SFOnlineHelper.onPause (this);`

6、public static void onRestart (Activity activity)

该方法在游戏 Activity 中的 onRestart 中调用

调用用例： `SFOnlineHelper.onRestart (this);`

2.5.2.2 用户登陆接口 （必选接口）

```
static void SFGameNativeInterface::login(const char* params);
```

该方法用于登陆，调用用例：

```
SFGameNativeInterface::setLoginCallback(&loginCallback);//设置回调函数
SFGameNativeInterface::login("login");
```

2.5.2.3 用户登出接口（可选接口）

```
static void SFGameNativeInterface::logout(const char* params);
```

该方法用于登出，调用用例：

```
SFGameNativeInterface::setLoginCallback(&loginCallback);//设置回调函数
SFGameNativeInterface::logout("logout");
```

2.5.2.4 退出接口 （必选接口）

```
static void SFGameNativeInterface::onExit();
```

该方法用于退出，调用用例：

```
SFGameNativeInterface::setExitCallback(&ExitCallback);//设置回调函数
SFGameNativeInterface::onExit();
```

2.5.2.5 非定额计费接口（可选接口）

```
static void SFGameNativeInterface::charge(const char* itemName, int unitPrice, int count, const char* callBackInfo, const char* callBackUrl);
```

该接口用于用户触发计费进行付费行为的入口函数。合作伙伴可以在需要计费的地方调用此接口进行计费。该接口用于非定额计费。

参数描述：

参数名称	类型	注释
itemName	const char*	虚拟货币名称
unitPrice	int	游戏道具价格，单位为人民币分
count	int	用户选择购买道具界面的默认道具数量.(count*unitPrice)

callBackInfo	const char*	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）
callBackUrl	const char*	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息

调用用例：

```
SFGameNativeInterface::setPayResultCallback(&payCallback);//设置回调
SFGameNativeInterface::charge("200金币", 2, 1, "购买金币",
      “http://192.168.0.224:8980/omsdk-cp/user/paylog/sync”);
```

2.5.2.6 定额计费接口 （必选接口）

```
static void SFGameNativeInterface::pay(int unitPrice, const char*  unitName,
       int count, const char*  callBackInfo, const char* callBackUrl);
```

该接口用于定额支付的接口函数。合作伙伴在需要使用定额支付的时候使用该接口。

参数描述：

参数名称	类型	注释
unitPrice	int	游戏道具价格，单位为人民币分
unitName	const char*	虚拟货币名称
count	int	用户选择购买道具界面的默认道具数量。（总价 count*unitPrice）
callBackInfo	const char*	由游戏开发者定义传入的字符串，会与支付结果一同发送给游戏服务器，游戏服务器可通过该字段判断交易的详细内容（金额角色等）

callBackUrl	const char*	将支付结果通知给游戏服务器时的通知地址 url，交易结束后，系统会向该 url 发送 http 请求，通知交易的结果金额 callbackInfo 等信息

调用用例：

```
SFGameNativeInterface::setPayResultCallback(&payCallback);
SFGameNativeInterface::pay(1, "100 金币", 1, "购买金币",
    “http://192.168.0.224:8980/omsdk-cp/user/paylog/sync”);
```

2.5.2.7 设置角色基本数据（必选接口）

```
static void SFGameNativeInterface::setRoleData(const char* roleId, const char*
roleName, const char* roleLevel, const char* zoneId, const char* zoneName);
```

部分渠道如UC渠道，要对游戏人物数据进行统计，而且为接入规范，

调用时间：在游戏登录验证成功后

参数描述：

参数名称	类型	注释
roleId	const char*	角色唯一标识
roleName	const char*	角色名
roleLevel	const char*	角色等级
zoneId	const char*	区域唯一标识
zoneName	const char*	区域名称

2.5.2.8 扩展接口 1（必选接口）

```
static void SFGameNativeInterface::setData(const char* key, const char* value);
```

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要

上报角色信息，为接入规范，所以为必选接口。

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下：

roleId :当前登录的玩家角色 ID，必须为数字
roleName :当前登录的玩家角色名，不能为空，不能为 null
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
balance :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 :当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”

@key 键值定义和调用实例如下：

```
rapidjson::Document writedoc;
writedoc.SetObject();
rapidjson::Document::AllocatorType& allocator = writedoc.GetAllocator();
rapidjson::Value root(rapidjson::kObjectType);

// json object 格式添加 “名称/值” 对
root.AddMember("roleId", 1, allocator);
root.AddMember("roleName", "猎人", allocator);
root.AddMember("roleLevel", "100", allocator);
root.AddMember("zoneId", "1", allocator);
root.AddMember("zoneName", "阿狸一区", allocator);
root.AddMember("balance", "0", allocator);
root.AddMember("vip", "1", allocator);
root.AddMember("partyName", "无帮派", allocator);

rapidjson::StringBuffer buffer;
rapidjson::Writer<rapidjson::StringBuffer> writer(buffer);
root.Accept(writer);
const char* value = buffer.GetString();
```

key	接口描述	调用示例
createrole	创建新角色时调用	SFGameNativeInterface::setData ("createrole" , value);
levelup	玩家升级角色时调用	SFGameNativeInterface::setData ("levelup" , value);
enterServer	选择服务器进入时调用	SFGameNativeInterface::setData ("enterServer" , value);

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个

渠道的“参数填写帮助”

2.5.2.9 扩展接口 2（可选接口）

```
const static char* SFGameNativeInterface::extend(const char* data, int count);
```

扩展接口，有些 SDK， 要求必须接入统计接口或者其它特殊的接口，并且有返回值或者回调的函数，用户可以使用此接口调用，具体可以参考易接工具上的 SDK 的参数填写帮助。

参数说明：

```
/** 扩展接口
 * data:需要传入的数据
 * count:扩展接口回调函数的数量，没有可以传“0”
 */
```

调用示例：

```
SFGameNativeInterface::extend ("data ",2);
```

2.5.2.10 初始化回调类

```
class SFNativeOnlineInitCallback
{
public:
    SFNativeOnlineInitCallback() {};
    virtual ~SFNativeOnlineInitCallback(){};
    virtual void onResponse(const char* r, const char* remain) = 0;
};
```

设置接口为：

```
static void SFGameNativeInterface:: setInitCallback(
                        SFNativeOnlineInitCallback * callback);
```

示例：

```
/*游戏初始化回调类*/
class SFNativeOnlineInitCallbackImpl: public SFNativeOnlineInitCallback {
    virtual void onResponse(const char* r, const char* remain) {
            log("初始化：%s", r);
    }
};
SFNativeOnlineInitCallbackImpl* initCallback =new SFNativeOnlineInitCallbackImpl();
```

```
SFGameNativeInterface:: setInitCallback (initCallback);
```

2.5.2.11 登陆登出回调类（必选接口）

```
class SFNativeOnlineLoginCallback
{
public:
    SFNativeOnlineLoginCallback() {};
    virtual ~SFNativeOnlineLoginCallback() {};
    // 用户调用SFGameNativeInterface::logout时，会被调用
    virtual void onLogout(const char* remain) = 0;
    // 用户调用SFGameNativeInterface::login登陆成功时，会被调用
    // user为登陆时返回相关参数包含id token 渠道号等
    virtual void onLoginSuccess(SFNativeOnlineUser* user, const char* remain) = 0;
    // 用户调用SFGameNativeInterface::login登陆失败时，会被调用
    virtual void onLoginFailed(const char* r, const char* remain) = 0;
};
```

登陆回调类需要在登陆之前设置，可以只设置一次。设置接口为：

```
static void SFGameNativeInterface::setLoginCallback(
                        SFNativeOnlineLoginCallback* callback);
```

2.5.2.12 支付回调类

```
class SFNativeOnlinePayResultCallback
{
public:
   SFNativeOnlinePayResultCallback() {};
   virtual ~SFNativeOnlinePayResultCallback() {};
   // 用户支付失败时，会被调用
   virtual void onFailed(const char* remain) = 0;
   //用户支付成功时，会被调用
   virtual void onSuccess(const char* remain) = 0;
   //用户支付过程中，创建订单号时会被调用
   virtual void onOderNo(const char* orderNo) = 0;
};
```

支付回调类需要在支付之前设置，可以只设置一次。设置接口为：

```
static void SFGameNativeInterface::setPayResultCallback (
```

```
                        SFNativeOnlinePayResultCallback * callback);
```

2.5.2.13 退出回调类（必选接口）

```
class SFNativeOnlineExitCallback
{
public:
 SFNativeOnlineExitCallback() {};
   virtual ~SFNativeOnlineExitCallback() {};
   //当渠道不需要弹出渠道退出界面时，此函数会被调用
   virtual void onNoExiterProvide() = 0;
    //当渠道需要弹出渠道退出界面时，易接SDK调起渠道退出界面，用户确认之后，此函数会被调用
   virtual void onSDKExit(bool result) = 0;
};
```

退出回调类需要在退出之前设置，可以只设置一次。设置接口为：

```
 static void SFGameNativeInterface:: setExitCallback(
                        SFNativeOnlineExitCallback * callback);
```

2.5.2.14 扩展回调类（可选接口）

```
class SFNativeOnlineExtendCallback
{
public:
    SFNativeOnlineExtendCallback() {};
    virtual ~SFNativeOnlineExtendCallback() {};
    virtual void onResponse(int index,const char* r, const char* remain) = 0;
};
```

扩展回调类需要在调用扩展接口之前设置，可以只设置一次。设置接口为：

```
static void SFGameNativeInterface:: setExtendCallback(
                        SFNativeOnlineExtendCallback* callback);
```

示例：

```
//扩展接口回调
class SFNativeOnlineExtendCallbackImpl: public SFNativeOnlineExtendCallback {
    virtual void onResponse(int index ,const char* r, const char* remain){
        if(index==1){
            stringstream ss;
            ss << "扩展回调1:";
            ss << r <<','<<remain<< ';';
```

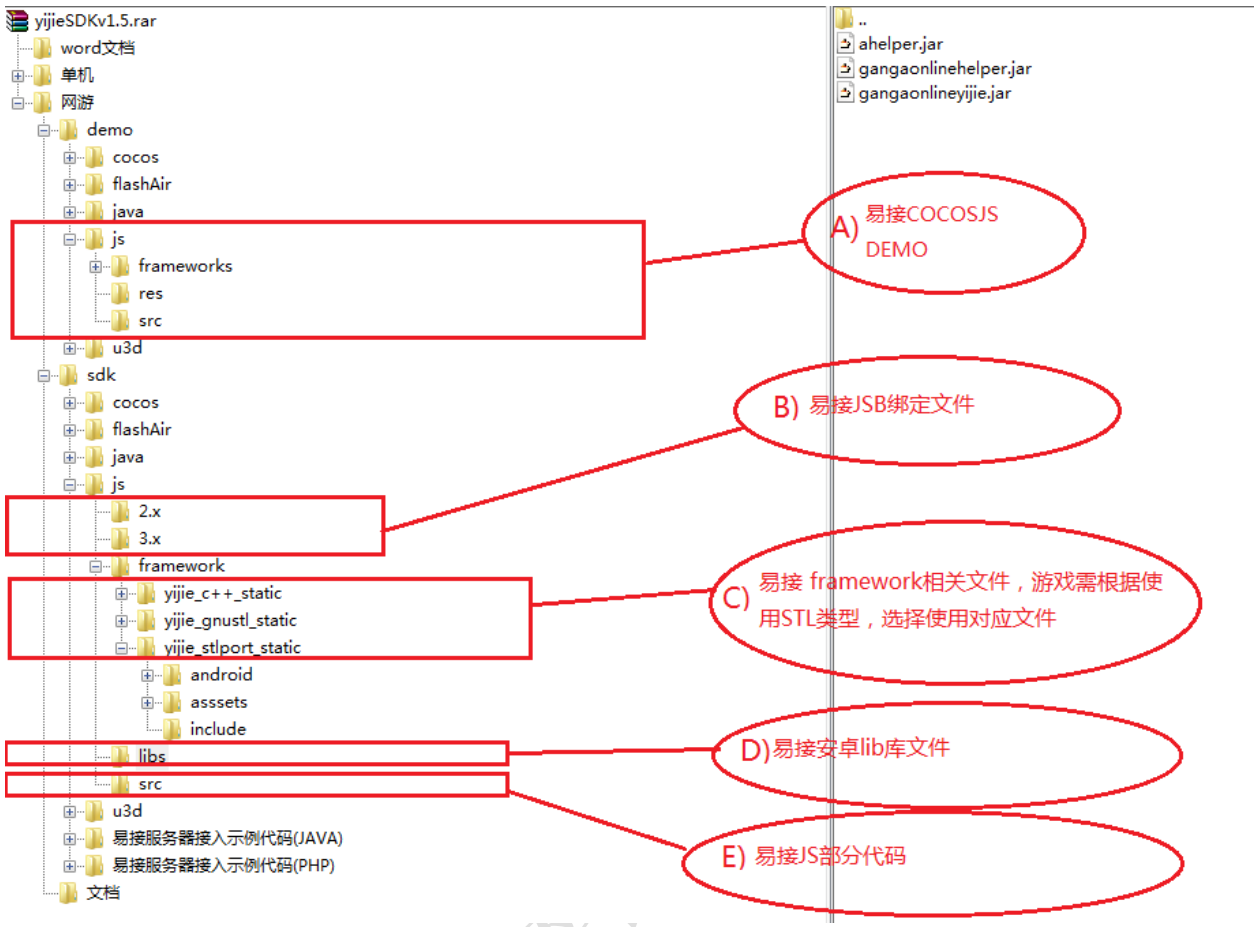
```
            textPayInfo->setString(ss.str());
        }
        if(index==2)
        {
            stringstream ss;
            ss << "扩展回调2:";
            ss << r <<','<<remain<< ';';
            textPayInfo->setString(ss.str());
        }
    }
};
SFNativeOnlineExtendCallbackImpl extendCallback =SFNativeOnlineExtendCallbackImpl();
//设置扩展回调
SFGameNativeInterface:: setExtendCallback (&extendCallback);
//调用SDK扩展接口
SFGameNativeInterface::extend("data",2);
```

2.6 JS 游戏接口

2.6.1 易接 COCOS JS 使用准备

易接官网 http://www.1sdk.cn/download.html 下载 SDK。

解压后的目录及结构 JS 部分的说明如下：



（图 2.6）

2.6.1.1 拷贝资源

1、根据游戏使用的 COCOS JS 的版本，选择 2.x 或 3.x 文件夹下内容拷贝到游戏对应 frameworks\runtime-src\Classes 目录下。

2、在游戏的 frameworks\runtime-src\proj.android 目录下新建 yijie 文件夹，根据游戏 frameworks\runtime-src\proj.android\jni\Application.mk 中 APP_STL 值的类型， 选择对应文件夹下（图 2.6 标注的 C 部分）内容拷贝到新建的 yijie 文件夹下。

3、 图 2.6 标注的 D 部分拷贝到游戏的

frameworks\runtime-src\proj.android\libs 目录下，如果没有 libs 目录，请新建此文件夹。

4、图 2.6 标注的 E 部分拷贝到游戏的 src 目录下。

2.6.1.2 添加易接提供的 Activity 引用

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 文件中添加由易接提供如下代码：

```
<service
    android:name="com.snowfish.a.a.s.ABGSvc"
    android:enabled="true"
    android:process="com.snowfish.a.a.bg">
    <intent-filter>
        <action android:name="com.snowfish.a.a.s.ABGSvc" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</service>
<meta-data
    android:name="com.snowfish.customer"
    android:value="SNOWFISH" >
</meta-data>
<meta-data
    android:name="com.snowfish.channel"
    android:value="SNOWFISH" >
</meta-data>
<meta-data
    android:name="com.snowfish.sdk.version"
    android:value="2" >
</meta-data>
<meta-data
    android:name="com.snowfish.appid"
    android:value="KEY" >
</meta-data>
<meta-data
    android:name="com.snowfish.channelid"
    android:value="{4ff036a1-3254eafe}" >
</meta-data>
```

标红处的 KEY 值是由易接用户中心申请获取格式如：{12345678-12345678}。

2.6.1.3 修改 Application

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 文件中修改 Application。

1）若无自定义 Applicatiion，则修改 AndroidManifest.xml 的 Application 如下：

```
<application android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2）若开发者自定义的Application。则自定义Application需要继承 com.snowfish.cn.ganga.helper.SFOnlineApplication ，AndroidManifest.xml修改如下:

```
<application android:name="自定义Application"/>
```

2.6.1.4 添加易接需要的 use-permision

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 中添加 use-permision 如下：

```
<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.WRITE_EXTERNAL_STORAGE"/>
<uses-permissionandroid:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permissionandroid:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permissionandroid:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permissionandroid:name="android.permission.READ_PHONE_STATE"/>
<uses-permissionandroid:name="android.permission.GET_TASKS"/>
<uses-permissionandroid:name="android.permission.WAKE_LOCK"/>
<uses-permissionandroid:name="android.permission.CHANGE_NETWORK_STATE"/>
```

2.6.1.5 增加易接闪屏 Activity

继承 com.snowfish.cn.ganga.helper.SFOnlineSplashActivity 并将该 Activity 设置为程序启动时的 Activity。

```
public class MySplashActivity extends SFOnlineSplashActivity {
```

```
    publicint getBackgroundColor() {
    // 返回闪屏的背景颜色
        return Color.WHITE;
        }
    @Override
    publicvoidonSplashStop() {
    // 闪屏结束进入游戏
        Intent intent = newIntent(this, MainActivity.class);
        startActivity(intent);
        this.finish();
    }
  }
```

AndroidManifest.xml 中关于该 Activity 的声明，要声明为启动 Activity

```
  <activity
  android:name=" com.snowfish.cn.ganga.helper.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>
```

2.6.1.6 修改 Application.mk

在游戏frameworks\runtime-src\proj.android\jni\Application.mk中增加

```
LOCAL_WHOLE_STATIC_LIBRARIES += YijieCocosJSStatic
LOCAL_SRC_FILES += ../../Classes/yijie.cpp
LOCAL_C_INCLUDES +=$(LOCAL_PATH)/../yijie/android  \
                   $(LOCAL_PATH)/../yijie/include
$(call import-module,yijie/android)
```

修改后如下图

```
LOCAL_PATH := $(call my-dir)



include $(CLEAR_VARS)

LOCAL_MODULE := cocos2djs_shared

LOCAL_MODULE_FILENAME := libcocos2djs

LOCAL_SRC_FILES := hellojavascript/main.cpp \
                   ../../Classes/AppDelegate.cpp \
                   ../../Classes/yijie.cpp

LOCAL_C_INCLUDES := $(LOCAL_PATH)/../../Classes \
                    $(LOCAL_PATH)/../yijie/android \
                    $(LOCAL_PATH)/../yijie/include

LOCAL_STATIC_LIBRARIES := cocos_jsb_static
LOCAL_WHOLE_STATIC_LIBRARIES += YijieCocosJSStatic
LOCAL_EXPORT_CFLAGS := -DCOCOS2D_DEBUG=2 -DCOCOS2D_JAVASCRIPT
include $(BUILD_SHARED_LIBRARY)


$(call import-module,bindings)
$(call import-module,yijie/android)
```

2.6.1.7 添加 yijiesdkconst.js

根据 cocos js 版本选择合适的修改方式。

a）cocos js 3.x 版本：

修改游戏 project.json 文件，"jsList" 数组中增加 "src/yijiesdkconst.js"

修改后如下图:

```
{
    "project_type": "javascript",

    "debugMode" : 1,
    "showFPS" : false,
    "frameRate" : 60,
"id" : "gameCanvas",
"renderMode" : 0,
"engineDir":"frameworks/cocos2d-html5",

"modules" : ["cocos2d",
    "extensions",
    "external" ],

"jsList" : [
"src/resource.js",
"src/app.js",
"src/yijiesdkconst.js"
]
}
```

b）cocos js 2.x 版本

修改游戏 cocos2d.js 文件(如果没有此文件，需要根据 AppDelegate.cpp 文件中调用.js 文件为准)，添加”src/yijiesdkconst.js”,修改后如下图所示：

```
//SingleEngineFile:'',
appFiles:[
    'src/resource.js',
    'src/myApp.js',//add your own files in orde
    'src/yijiesdkconst.js'
]
};
```

2.6.1.8 拷贝易接 SDK 资源及修改 NDK_MODULE_PATH

根据 cocos js 版本选择合适的修改方式。

a）cocos js 3.x 版本：

修改 frameworks\runtime-src\proj.android\build-cfg.json 文件增加"copy_resources":数组中增加

```
{
    "from": "yijie/asssets/Sonnenblume",
    "to": "Sonnenblume"
}
```

修改后如下图。

ndk_module_path

```
{
    "ndk_module_path" :[
        "../../js-bindings",
        "../../js-bindings/cocos2d-x",
        "../../js-bindings/cocos2d-x/cocos",
        "../../js-bindings/cocos2d-x/external",
        ""
    ],
    "copy_resources": [
        {
            "from": "../../../src",
            "to": "src"
        },
        {
            "from": "../../../res",
            "to": "res"
        },
        {
            "from": "../../../main.js",
            "to": ""
        },
        {
            "from": "../../../project.json",
            "to": ""
        },
        {
            "from": "../../js-bindings/bindings/script",
            "to": "script"
        }
        ,
        {
            "from": "yijie/asssets/Sonnenblume",
            "to": "Sonnenblume"
        }
    ]
}
```

b）cocos js 2.x 版本：

修改 proj.android\build_native.sh 文件，增加内容：

YIJIE_RESROUCE_ROOT="$DIR/yijie/asssets/Sonnenblume"
cp -rf "$YIJIE_RESROUCE_ROOT" "$APP_ANDROID_ROOT"/assets
${APP_ROOT}/proj.android:

修改后如图：

```
BINDINGS_JS_ROOT="$APP_ROOT/../../../scripting/javascript/bindings/js"
YIJIE_RESROUCE_ROOT="$DIR/yijie/asssets/Sonnenblume"

echo
echo "Paths"
echo "    NDK_ROOT = $NDK_ROOT"
echo "    COCOS2DX_ROOT = $COCOS2DX_ROOT"
echo "    APP_ROOT = $APP_ROOT"
                                    3个计数行
# make sure assets is exist
if [ -d "$APP_ANDROID_ROOT"/assets ]; then
    rm -rf "$APP_ANDROID_ROOT"/assets
fi

mkdir "$APP_ANDROID_ROOT"/assets
# copy "Resources" into "assets"
cp -rf "$RESROUCE_ROOT" "$APP_ANDROID_ROOT"/assets

# copy yijie resource from yijie/asssets/Sonnenblume into assets
cp -rf "$YIJIE_RESROUCE_ROOT" "$APP_ANDROID_ROOT"/assets

# copy MoonWarriors js
cp -rf "$RESROUCE_ROOT"/../src "$APP_ANDROID_ROOT"/assets
# copy MoonWarriors-native.js
cp "$RESROUCE_ROOT"/../*.js "$APP_ANDROID_ROOT"/assets

# copy bindings/*.js into assets' root
cp -f "$BINDINGS_JS_ROOT"/*.js "$APP_ANDROID_ROOT"/assets

echo "Using prebuilt externals"
echo

set -x

"$NDK_ROOT"/ndk-build $PARALLEL_BUILD_FLAG -C "$APP_ANDROID_ROOT" $* \
    "NDK_MODULE_PATH=${APP_ROOT}/proj.android:${COCOS2DX_ROOT}:${COCOS2DX_ROOT}/cocos2
```

2.6.1.9 修改 AppDelegate. cpp 文件

在 AppDelegate.cpp 中 sc->start();之前添加如下代码：

```
#include "yijie.hpp"

sc->addRegisterCallback(register_all_yijie);
```

修改之后效果图如下：

```
#include "jsb_cocos2dx_extension_manual.h"
#include "js_bindings_chipmunk_registration.h"
#include "jsb_opengl_registration.h"
#include "yijie.hpp"
USING_NS_CC;
using namespace CocosDenshion;

AppDelegate::AppDelegate()
{
}
23个计数行
    sc->addRegisterCallback(register_all_cocos2dx);
    sc->addRegisterCallback(register_all_cocos2dx_extension);
    sc->addRegisterCallback(register_cocos2dx_js_extensions);
    sc->addRegisterCallback(register_all_cocos2dx_extension_manual);
    sc->addRegisterCallback(jsb_register_chipmunk);
    sc->addRegisterCallback(JSB_register_opengl);
    sc->addRegisterCallback(register_all_yijie);
    sc->start();

    CCScriptEngineProtocol *pEngine = ScriptingCore::getInstance();
```

2.6.2 接口调用说明

以下接口可以参考 yijiejsdemo。

2.6.2.1 易接 Android Java 部分初始化（必选接口）

找到游戏工程的主 Activity，以 Cocos2d-x 引擎游戏为例，主 Activity 即是继承了 cocos2dxActivity 的 Activity。 并在主 Activity 的 onCreate()方法中新增如下代码来初始化易接 SDK：修改 proj.android/src/org/cocos2dx/javascript/AppActivity.java 文件

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /*
     * 易接初始化
     * */
```

注：易接初始化提供两种方法，只需要调用一种即可。

1.初始化无回调接口

```
    SFOnlineHelper.onCreate(this);
```

```
        SFNativeAdapter.init(this, new SFActionCallback() {

            @Override
            public void callback(Runnable arg0) {
                runOnGLThread(arg0);
            }
        });
```

2. 初始化带回调接口

```
    SFNativeAdapter.init_listener(this, new SFActionCallback() {

            @Override
            public void callback(Runnable arg0) {
                runOnGLThread(arg0);
            }
        });

    }

    @Override
    protected void onResume() {
        super.onResume();

        /*
         * 易接 onResume
         * */
        SFOnlineHelper.onResume(this);
    }

    @Override
    protected void onPause() {
        super.onPause();

        /*
         * 易接 onPause
         * */
        SFOnlineHelper.onPause(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        /*
```

```
        * 易接 onDestroy
        * */
       SFOnlineHelper.onDestroy(this);
    }

    @Override
    protected void onStop() {
        super.onStop ();
     /*
        * 易接 onStop
        * */
    SFOnlineHelper.onStop(this);
    }

    @Override
    protected void onRestart() {
        super.onRestart ();
     /*
        * 易接 onRestart
        * */
    SFOnlineHelper.onRestart(this);  ;
    }
```

2.6.2.2 易接 JNI 部分初始化

修改 proj.android\jni\hellojavascript\main.cpp 文件
增加下图红色部分。

```
#include "SFGameNative.hpp"
#define  LOG_TAG    "main"
#define  LOGD(...)
__android_log_print(ANDROID_LOG_DEBUG,LOG_TAG,__VA_ARGS__)

using namespace cocos2d;
void cocos_android_app_init (JNIEnv* env, jobject thiz) {
LOGD("cocos_android_app_init");
AppDelegate *pAppDelegate = new AppDelegate();
    SFGameNative::init(env);
}
```

2.6.2.3 实例化易接 SDK 接口

```
//实例化易接 SDK， 全局唯一
 self.gameNativeInterface = new SFGameNativeInterface();
```

2.6.2.4 设置初始化回调接口

```
  注：在易接初始化的时候调用不带回调接口，这里不需要设置
  //设置初始化回调

  self.gameNativeInterface.setInitCallback(self.initListener, self);

  //初始化回调
  initListener: function(result,msg) {
         var self = this;

         self.tipinlogoui(result);      },
```

2.6.2.5 易接用户登陆接口（必选接口）

```
//设置登陆回调
self.gameNativeInterface.setLoginCallback(self.loginListener, self);

//调用登陆接口
self.gameNativeInterface.login("");

//易接用户登陆回调
loginListener: function(result, user, remain){
    var self = this;
    if  (result == YiJieLoginResultCode.YJLoginSuccess) {
        /*收到登陆成功消息后，请求登陆校验，具体流程如下：
          1  游戏客户端请求游戏服务器
          2  游戏服务器请求易接服务器
        */
        var appid = user.getProductCode();
         /*易接平台用来标示应用的 ID 格式如： {XXXXXXXX-XXXXXXXX}*/
        var sdkid = user.getChannelId();
         /*易接平台标示的渠道的 ID 格式如： {XXXXXXXX-XXXXXXXX}*/
        var userid = user.getChannelUserId();
         /*渠道的用户 ID， 游戏开发者可以根据 sdkid & userid 标示唯一用户，
         部分渠道 sdk 的 userid 中可能包含特殊字符，如& 数据传输时请加密处理*/
        var token = user.getToken();
         /*会话标示 部分渠道 sdk 的 token 中可能包含特殊字符
```

```
        ，如& 数据传输时请加密处理*/

        //此处游戏可以处理请求游戏服务器做登陆校验
    } else if (result == YiJieLoginResultCode.YJLoginFail) {
        self.tipinlogoui("登陆失败");
    } else if (result == YiJieLoginResultCode.YJLoginOut) {
        self.tipinlogoui("切换账号");
        self.login();/*收到切换账号信息，游戏需主动调用此接口 切换账号*/
    }
},
```

2.6.2.6 定额计费接口 （必选接口）

部分渠道 SDK 不支持定额支付接口，即：用户在支付界面时可以修改支付金额，所以游戏支付成功的标准一定以服务支付回调回来的金额为准

```
this.gameNativeInterface.setPayResultCallback(self.payListener, self);
this.gameNativeInterface.pay(
parseInt(price)/*道具价格单位分*/,
name/*道具名称*/,
 1 /*个数默认为1*/,
 "callbackinfo"/*此次支付透传参数*/,
 "http://game.pay?id=1"/*游戏接收支付回调地址*/);

//支付回调
payListener: function(result, msg) {
    var self = this;
    switch (result) {
        case YiJiePayResultCode.YJPaySuccess:
            break;
        case YiJiePayResultCode.YJPayFail:
            break;
        case YiJiePayResultCode.YJPayOderNo:
            break;
    }
},
```

2.6.2.7 非定额计费接口（可选接口）

非定额支付接口，即：用户在支付界面时可以修改支付金额，所以游戏支付成功的标准一定以服务支付回调回来的金额为准

```
this.gameNativeInterface.setPayResultCallback(self.payListener,self);
this.gameNativeInterface.charge(
```

```
"100"/*道具名称*/,
100/*价格单位分*/,
1/*道具个数*/,
"test callbackinfo"/*此次支付透传参数*/,
"http://game.pay?id=1"/*游戏接收支付回调地址*/);
```

2.6.2.8 设置角色基本数据（必选接口）

游戏登陆完成后根据用户选择的角色调用此接口，切记游戏内用户切换角色也需要调用

```
this.gameNativeInterface.setRoleData(
"001"/*角色唯一标识*/,
"法师"/*角色名*/,
"30 级"/*角色等级*/,
"1000"/*区域唯一标识*/,
"一区"/*区域名称*/);
```

2.6.2.9 退出回调接口（必选接口）

在游戏退出按键上调用如下代码：

```
self.gameNativeInterface.setExitCallback(self.exitcallback, self);
self.gameNativeInterface.exit();
```

退出回调函数说明：

```
exitcallback: function (result, msg){
    var self = this;
    switch (result) {
        case YiJieExitResultCode.YJExitNoExiterProvide:
            //渠道 SDK 没有提供渠道界面， 游戏客户端可以在此处设计退出界面，亦或者直接调用
cc.director.end();
            break;
        case YiJieExitResultCode.YJExitSDKExit:
            if (msg == true) {
                //渠道 SDK 退出界面上，用户选择退出
                cc.director.end();
            } else {
                //渠道 SDK 退出界面上，用户选择取消
            }
            break;
    }
},
```

2.6.2.10 扩展接口 1（必选接口）

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要上报角色信息，为接入规范，所以为必选接口。

接口：**this.gameNativeInterface**.setData(key,value);

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下：

roleId :当前登录的玩家角色 ID，必须为数字
roleName :当前登录的玩家角色名，不能为空，不能为 null
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
balance :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 :当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”

@key 键值定义和调用实例如下：

```
var roledata = new Object();
 roledata.roleId = "1";
 roledata.roleName = "猎人"
 roledata.roleLevel = "100";
 roledata.zoneId = "1";
 roledata.zoneName = "阿里一区";
 roledata.balance = "0";
 roledata.vip = "1";
 roledata.partyName = "无帮派";
 var json = JSON.stringify(roledata);
this.gameNativeInterface.setData("createrole",json);
```

key	接口描述	调用示例
createrole	创建新角色时调用	**this.gameNativeInterface**.setData ("createrole" , value);
levelup	玩家升级角色时调用	**this.gameNativeInterface**.setData ("levelup" , value);
enterServer	选择服务器进入时调用	**this.gameNativeInterface**.setData ("enterServer" , value);

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个渠道的“参数填写帮助”

2.6.2.11 扩展接口 2（可选接口）

扩展接口，有些 SDK， 要求必须接入统计接口或者其它特殊的接口，并且有返回值或者回调的函数，用户可以使用此接口调用，具体可以参考易接工具上的 SDK 的参数填写帮助。

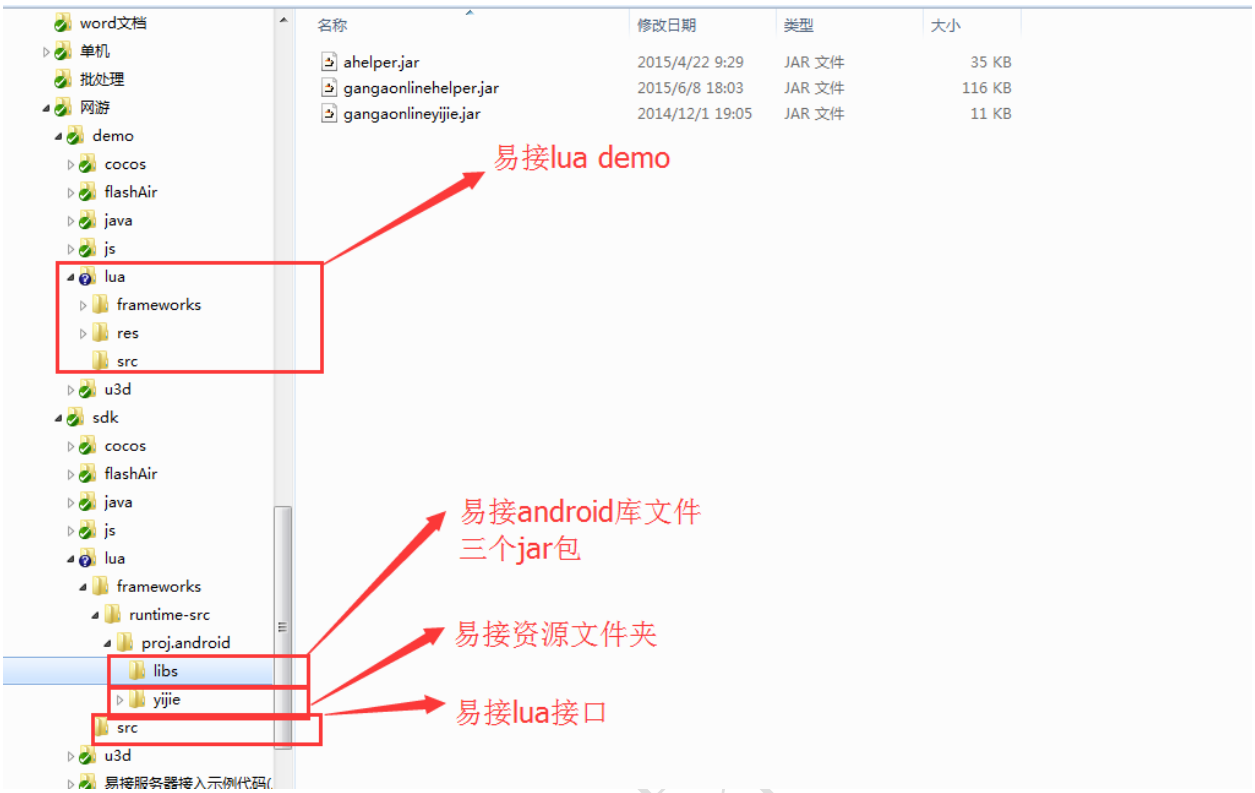
```
//注册扩展函数回调
self.gameNativeInterface.setExtendCallback(self.extendListener, self);
 //扩展方法调用
//data 是需要传入接口的数据，如果参数比较多，可以传 json 字符串
//count 需要注册多少个回调函数
self.gameNativeInterface.extend("data"/*需要传入接口的数据，如果参数比较多，可以传 json 字符串*/,
2 /*需要注册多少个回调函数*/
 );
//扩展函数回调
extendListener: function(index,result,msg) {
      var self = this;
      self.tipinlogoui("extend");
    },
```

2.7 Lua 游戏接口

2.7.1 易接 COCOS lua 使用准备

易接官网 http://www.1sdk.cn/download.html 下载 SDK。

解压后的目录及结构 lua 部分的说明如下：



（图 2.7-1）

2.7.1.1 拷贝资源

1、将 SDK\lua 目录下 frameworks\runtime-src\proj.android 中的 libs 和 yijie 文件夹拷贝到对应目录中。

2、将 SDK\lua 目录下 src\YijieInterface.lua 文件拷贝到对应目录中。

2.7.1.2 添加易接提供的 Activity 引用

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 文件中添加由易接提供如下代码：

```
<service
    android:name="com.snowfish.a.a.s.ABGSvc"
    android:enabled="true"
    android:process="com.snowfish.a.a.bg">
    <intent-filter>
        <action android:name="com.snowfish.a.a.s.ABGSvc" />
```

```
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </service>
    <meta-data
        android:name="com.snowfish.customer"
        android:value="SNOWFISH" >
    </meta-data>
    <meta-data
        android:name="com.snowfish.channel"
        android:value="SNOWFISH" >
    </meta-data>
    <meta-data
        android:name="com.snowfish.sdk.version"
        android:value="2" >
    </meta-data>
    <meta-data
        android:name="com.snowfish.appid"
        android:value="KEY" >
    </meta-data>
    <meta-data
        android:name="com.snowfish.channelid"
        android:value="{4ff036a1-3254eafe}" >
    </meta-data>
```

标红处的 KEY 值是由易接用户中心申请获取格式如：{12345678-12345678}。

2.7.1.3 修改 Application

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 文件中修改 Application。

1）若无自定义 Applicatiion，则修改 AndroidManifest.xml 的 Application 如下：

```
<application android:name="com.snowfish.cn.ganga.helper.SFOnlineApplication"/>
```

2）若开发者自定义的Application。则自定义Application需要继承 com.snowfish.cn.ganga.helper.SFOnlineApplication ，AndroidManifest.xml修改如下:

```
<application android:name="自定义Application"/>
```

2.7.1.4 添加易接需要的 use-permision

在游戏 frameworks\runtime-src\proj.android\AndroidManifest.xml 中添加 use-permision 如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
```

2.7.1.5 增加易接闪屏 Activity（必选接口）

继承 com.snowfish.cn.ganga.helper.SFOnlineSplashActivity 并将该 Activity 设置为程序启动时的 Activity。

```
public class SplashActivity extends SFOnlineSplashActivity {
 publicint getBackgroundColor() {
 // 返回闪屏的背景颜色
     return Color.WHITE;
     }
 @Override
 public void onSplashStop() {
 // 闪屏结束进入游戏
     Intent intent = newIntent(this, AppActivity.class);
     startActivity(intent);
     this.finish();
 }
}
```

AndroidManifest.xml 中关于该 Activity 的声明，要声明为启动 Activity

```
<activity
android:name="org.cocos2dx.lua. SplashActivity"
android:configChanges="orientation|keyboardHidden|screenSize">
<intent-filter>
```

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

2.7.1.6 拷贝易接 SDK 资源

修改 frameworks\runtime-src\proj.android\build-cfg.json 文件增加 "copy_resources":数组中增加

```
{
    "from": "yijie/asssets/Sonnenblume",
    "to": "Sonnenblume"
}
```

修改后如下图。

```
    ],
    "copy_resources": [
        {
            "from": "../../../src",
            "to": "src"
        },
        {
            "from": "../../../res",
            "to": "res"
        },
        {
            "from": "../../cocos2d-x/cocos/scripting/lua-bindings/script"
            "to": ""
        },
        {
            "from": "yijie/asssets/Sonnenblume",
            "to": "Sonnenblume"
        }
    ]
```

2.7.2 接口调用说明

以下接口可以参考易接 lua demo。

2.7.2.1 易接 Android Java 部分初始化（必选接口）

找到游戏工程的主 Activity，以 Cocos2d-x 引擎游戏为例，主 Activity 即是继承了 cocos2dxActivity 的 Activity。 并在主 Activity 的 onCreate()方法中新增如下代码来初始化易接 SDK：修改 proj.android/src/org/cocos2dx/lua/AppActivity.java 文件

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /*初始化Lua组件*/
    SFLuaAdaper.init(this, new SFActionCallback() {

        @Override
        public void callback(Runnable run) {
            AppActivity.this.runOnGLThread(run);
        }
    });
}

@Override
protected void onResume() {
    super.onResume();
    /*易接 onResume*/
    SFOnlineHelper.onResume(this);
}

@Override
protected void onPause() {
    super.onPause();
    /*易接 onPause*/
    SFOnlineHelper.onPause(this);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    /*易接 onDestroy*/
    SFOnlineHelper.onDestroy(this);
}

@Override
protected void onStop() {
    super.onStop();
    /*易接 onStop*/
    SFOnlineHelper.onStop(this);
}

@Override
protected void onRestart() {
    super.onRestart ();
     /*易接 onRestart*/
```

```
        SFOnlineHelper.onRestart(this);  ;
    }
```

2.7.2.2 实例化易接 SDK 接口（必选接口）

```
--获取易接接口
 require "YijieInterface"
 local yijie = YijieInterface.new()
```

2.7.2.3 易接 SDK 初始化回调接口（必选接口）

```
--注册 SDK 初始化回调
yijie:setSDKInitListener(sdkInitCallbackFunc)
--SDK 初始化回调
local function sdkInitCallbackFunc(jPrama)
     require "json"
     local data = json.decode(jPrama);
     if data["tag"] == "success" then --SDK 初始化成功
     elseif data["tag"] == "fail" then --SDK 初始化失败
     end
end
```

2.7.2.4 易接用户登陆登出接口（必选接口）

```
--注册登陆回调
yijie:setLoginListener(loginCallback)

--调用登陆接口
yijie:login("login")
--调用登出接口
yijie:logout("logout")

--登陆回调
function loginCallback(jPrama)
        require "json"
        local data = json.decode(jPrama);
        if data["result"] == "success" then--登录成功
                local productCode = data ["productCode"]
                local channelId = data ["channelId"]
                local channelUserId = data ["channelUserId"]
                local token = data ["token"]
                local id = data ["id"]
```

```
                local userName = data ["userName"]
        elseif data["result"] == "fail" then--登陆失败
        elseif data["result"] == "logout" then--登出回调
        end
end
```

2.7.2.5 定额计费接口 （必选接口）

部分渠道 SDK 不支持定额支付接口，即：用户在支付界面时可以修改支付金额，所以游戏支付成功的标准一定以服务支付回调回来的金额为准

```
yijie:pay(
price/*道具价格单位分*/,
name/*道具名称*/,
 1 /*个数默认为1*/,
 "callbackinfo"/*此次支付透传参数*/,
 "http://game.pay?id=1"/*游戏接收支付回调地址*/,
payCallback);

--支付回调
function payCallback(jPrama)
    require "json"
    local data = json.decode(jPrama);
    if data["result"] == "success" then--支付成功
    elseif data["result"] == "oderno" then--支付订单
        print("支付订单号", data["remain"])
    elseif data["result"] == "fail" then--支付失败
    end
end
```

2.7.2.6 非定额计费接口（可选接口）

非定额支付接口，即：用户在支付界面时可以修改支付金额，所以游戏支付成功的标准一定以服务支付回调回来的金额为准

```
yijie:charge (
"100"/*道具名称*/,
100/*价格单位分*/,
1/*道具个数*/,
"test callbackinfo"/*此次支付透传参数*/,
"http://game.pay?id=1"/*游戏接收支付回调地址*/,
payCallback/*支付回调函数*/,);
```

2.7.2.7 设置角色基本数据（必选接口）

游戏登陆完成后根据用户选择的角色调用此接口，切记游戏内用户切换角色也需要调用

```
yijie:setRoleData(
"001"/*角色唯一标识*/,
"法师"/*角色名*/,
"30 级"/*角色等级*/,
"1000"/*区域唯一标识*/,
"一区"/*区域名称*/);
```

2.7.2.8 退出回调接口（必选接口）

在游戏退出按键上调用如下代码：

```
yijie:exit(exitCallback);
```

```
--退出回调
function exitCallback(jPrama)
    require "json"
    local data = json.decode(jPrama);
    if data["result"] == "exit" then--退出游戏
        os.exit()
    elseif data["result"] == "noExit" then--继续游戏
    elseif data["result"] == "noProvide" then--使用游戏自己的退出界面
    end
end
```

2.7.2.9 扩展接口 1（必选接口）

扩展接口，部分渠道要求在创建新角色，或者升级角色时、选择服务器时要上报角色信息，为接入规范，所以为必选接口。

接口：yijie:setDataString(key,value)

参数描述：

@value 请按要求传入 json 格式的字符串，JSONObject 键值定义如下

roleId :当前登录的玩家角色 ID，必须为数字
roleName :当前登录的玩家角色名，不能为空，不能为 null
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
balance :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 :当前角色所属帮派，不能为空，不能为 null，若无，传入“无帮派”

@key 键值定义和调用实例如下：

```
local json = require "json"
            local tab = {}
            tab["roleId"]="1"
            tab["roleName"]="猎人"
            tab["roleLevel"]="100"
            tab["zoneId"]="1"
            tab["zoneName"]="阿狸一区"
            tab["balance"]="0"
            tab["vip"]="1"
            tab["partyName"]="无帮派"
 local jsonData = json.encode(tab)
```

key	接口描述	调用示例
createrole	创建新角色时调用	yijie:setDataString("createrole ", jsonData);
levelup	玩家升级角色时调用	yijie:setDataString("levelup", jsonData);
enterServer	选择服务器进入时调用	yijie:setDataString("enterServer", jsonData);

注意：还有一些渠道需要接入一些特殊的接口，具体可参考易接工具上各个渠道的“参数填写帮助”

2.7.2.10 扩展接口（可选接口）

yijie:extend(data, count, callbackFunc)
data 是需要传入接口的数据，如果参数比较多，可以传 json 字符串
count 需要注册多少个回调函数
callbackFunc lua 回调函数
使用示例：

```
    yijie:extend (“data”, 3, extendCallback)
    function extendCallback (jPrama)
            require "json"
            local data = json.decode(jPrama);
            local index = data["result"]
            local tag = data["tag"]
        local value = data["value"]
```

```
end
```