海游充值SDK 对接文档

一、说明

充值 SDK

二、接入SDK

1、导入相关jar包

1. Eclipse 工程

> 无

2. Android Studio 工程

> 将sdk android_studio 目录下的 libs 中文件复制到你项目libs中
>
> 其中 universal-image-loader-1.9.5.jar 是图片加载框架，如果你的项目中也使用了此框架，无需重复加载
>
> 项目中引用 aar 包
>
> 在 android 同级标签中增加以下代码
>
> repositories { flatDir { dirs 'libs' } }
>
> 引用

```
implementation(name: 'haiyou_common_xx', ext: 'aar')
implementation(name: 'haiyou_topup_xx', ext: 'aar')
```

> 完整配置参考demo中的配置

2、配置工作

1. 权限

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<!-- 谷歌支付权限 -->
<uses-permission android:name="com.android.vending.BILLING" />

<!-- 图片缓存需要 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

2. 配置 appid 和 支付使用的 Activity

   在AndroidManifest.xml 中配置

```
<application>
    <activity
        android:name="com.walletfun.pay.task.WalletActivity“
        android:configChanges="screenSize|keyboardHidden|orientation"
        android:theme="@style/WalletPayTheme"/>
   <activity
         android:name="com.walletfun.common.app.HaiYouShowActivity"
         android:theme="@style/WalletPayTheme"
         android:configChanges="screenSize|keyboardHidden|orientation" />

    <receiver
         android:name="com.walletfun.common.HaiYouReceiver"
         android:exported="true">
        <intent-filter>
         <action
            android:name="com.android.vending.INSTALL_REFERRER" />
        </intent-filter>
   </receiver>

    <!-- 此处不配置可以在初始化时传入 -->
    <meta-data
        android:name="wallet_app_id"
        android:value="YOUR APP ID" />

</application>
```

3、初始化

在Activity 中 或者 Application 中进行初始化操作

在Activity 中需要在 onDestroy 中进行解绑

```
// 添加充值SDK

      WalletHelp.addSDK(new WalletPaySDK());
      //  设置应用环境 测试环境:true  线上环境:false
      // 测试环境第三方支付会始终打开，方便调试
      WalletHelp.setEnvDebug(false);
      // 设置用户 id ，可选择
      // WalletPaySDK.setUserId("abc");
      // 设置悬浮窗口状态回调
      WalletHelp.init(this,appid,key,WalletHelp.SANDBOX_CLOSE);
```

1. 开启悬浮窗口 只能在 Activity 中

```
// 悬浮窗口监听事件（非必须）
WalletFloatWindowListener windowListener = new WalletFloatWindowListener() {
        @Override
        public void onWindowShow() {
           //("悬浮窗展示\n");
        }

        public void onWindowDismiss() {

          //  textHistory.append("悬浮窗消失\n");
        }


        public void onThirdPayChange(String buttonName,boolean open) {

        //    ("第三支付打开状态:"  open );
        }

        /**
         * 返回一个自定义的值，会在支付完成的时候一块回调
         * @return
         */

       @Override
        public String onThirdPayClick(String buttonName) {
           switch (buttonName) {
                case FloatUtils.FLOATBUTTON_ACTIVITY:
                   // 点击活动按钮
                    return "  ";
                case FloatUtils.FLOATBUTTON_TOPUP:
                      // 点击充值按钮
                    return "额外参数：支付结束后 在out_order_id中展示";
                   // return  FloatUtils.FLOATBUTTON_NOTOPEN
            //  return FloatUtils.FLOATBUTTON_NOTOPEN 是拦截单击事件
            }
        }
```

```
    };

// WalletFloatWindowListener 悬浮窗口状态回调，可传 null
// 设置悬浮窗口状态回调
 WalletHelp.addFloatListener(Activity  activity,
                             WalletFloatWindowListener windowListener);

// 显示悬浮窗口
WalletHelp.showFloat(Activity activity);

//  线上环境时  必须达到额定关卡 等级 时长（SDK内部统计时长）是才能显示充值选项（后台配置 参照服务器配置文档）
//  测试环境 不调用此方法也能显示充值选项
WalletHelp.updatePlayer(String grade , String level );


@Override
protected void onDestroy() {
    // 关闭悬浮窗口

     WalletHelp.onDestory(this);

    super.onDestroy();
}

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        WalletHelp.onConfigurationChanged();
    }
```

2. 监听支付结果，**需要在Activity中开启，并且和打开悬浮窗的Activity保证为同一对象，否则无法监听**

```
private WalletPayHelper payHelper;

// 初始化，此处 Context 可以是任意 Context
payHelper = new WalletPayHelper(context);

// 设置支付状态回调
payHelper.setWalletCallback(WalletCallback callback);

// 进行一次查询，如果上次支付后异常退出或者
payHelper.queryOrder();

// 自己也可以主动调用查询订单接口，传入订单号即可
// payHelper.queryOrder(String orderId);

 @Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {

 // 以下代码是保证支付结果回调的重要代码，也是为什么要和打开悬浮窗必须是同一activity的主要原因
```

```
 // 并且在使用 startActivityForResult 时 requestCode 不要设置为 0x1025，即4133，防止出现意外情况
        if (payHelper.handleActivityResult(requestCode, resultCode, data)) {
            return;
        }

        super.onActivityResult(requestCode, resultCode, data);
    }


@Override
protected void onDestroy() {

    // 需要销毁
    if (payHelper != null) {
        payHelper.onDestroy();
    }

    super.onDestroy();
}

// 支付回调
private WalletCallback walletCallback = new WalletCallback() {
        @Override
        public void onQueryResult(PayResult result) {

            // 支付结果回调
            if (result != null && result.isValid()) {

                switch (result.status) {

                    case PayResult.PAY_STATUS_PENDING:
                        // 支付中，请重新查询或者从自己的服务器上获取数据

                        break;
                    case PayResult.PAY_STATUS_SUCCESS:
                        // 支付成功


                        break;
                    case PayResult.PAY_STATUS_FAIL:
                        // 支付失败

                        break;
                    case PayResult.PAY_STATUS_REFUND:
                        // 已经退款

                        break;
                    default:
                        // 未知状态，请重新查询或者从自己的服务器上获取数据

                        break;
```

```
                }


            }
        }
```

3. class 引用路径

```
import com.walletfun.common.app.WalletHelp;
import com.walletfun.common.floatwindow.WalletFloatWindowListener;
import com.walletfun.pay.PayEntry;
import com.walletfun.pay.PayResult;
import com.walletfun.pay.WalletCallback;
import com.walletfun.pay.WalletPayHelper;
import com.walletfun.pay.WalletPaySDK;
```

4. 其他参见接口文档和demo

三、商品参数

```
public class PayResult {

    /**
     * 订单 id
     */
    public String orderId;

    /**
     * 支付成功所得“游戏币数量”
     */
    public String gameCoin;

    /**
     * 支付成功后赠送的“游戏币数量”
     */
    public String giftGameCoin;

    /**
     * 额外参数
     * <p>
     * {@link WalletFloatWindowListener#onThirdPayClick()} 时传入的
     */
    public String extraString;
    }
```

四、更新说明

- 2018-12-25 v2.2

  1) 公共模块升级 将活动模块放入公共模块

- 2018-11-16 v2.1

  1) haiyou-pay-xx 依赖包更名 haiyou_topup_xx

  2)WalletPaySDK.updatePlayer(String grade , String level);

  更改为 WalletHelp.updatePlayer(String grade , String level);

  *

- 2018-11-16 v2.0.0 1）增加公共模块 haiyou-common 2）将WalletPay类名修改为WalletPaySDK 3) 将充值模块名称 修改为 haiyou-pay 4) 修改初始化方式 将嗨游SDK公共参数通过WalletHelp传递 各SDK保留独有参数的配置 5) 接口统一增加签名验证 6) 增加沙盒测试功能（初始化时配置） *

- 2018-11-5 v1.2.2
  -
- 2018-10-20 v1.2.1
  -
- 2018-09-20 v1.0.0
  - SDK 基础功能：支付支持