SDK接入文档

时间	文档版本	备注
2019/11/01	1.0.0	SDK基本功能已经实现
2019/11/20	1.0.1	demo增加了获取授权码的实现，修复64位手机无法使用x5问题

接入准备

1. 将资源文件中的aar文件拷贝到项目的libs目录下

2. 在app module的build.gradle下做如下配置

2.1 在android和dependencies节点中间加入如下代码：**此配置是为了引入aar**

```
repositories {
 flatDir {
 dirs 'libs'
 }
}
```

2.2 在app module的build.gradle下做如下引入:

```
defaultConfig {
  // 根据项目需要，加入以下代码之一,注意与下方的so文件所在位置保持相同。
    //ndk { abiFilters 'armeabi' }
    //ndk {abiFilters "armeabi-v7a"}
}
```

```
dependencies {
  implementation(name:'sdk-release', ext:'aar')
}
```

2.3 添加.so文件

so加载目录默认为：src/main/jniLibs，需要将.so文件放置在so加载目录的armeabi文件夹下
(没有该目录则新建一个，拷贝资源文件下的liblbs.so文件)，so加载目录请只保留armeabi或者armeabi-v7a。

SDK功能说明及调用方法

1、初始化(必接)

```
public void init(final Activity activity, final MTSDKCallback callback){}
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity
callback	MTSDKCallback	是	初始化，登陆，支付，退出等结果回调。 可以通过implements SDKCallback接口或new SDKCallback来操作

调用示例

```
SDK.getInstance().init(activity,callback);
```

2、登录(必接)

```
public void login(final Activity activity);
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity

调用示例

```
SDK.getInstance().login(activity);
```

3、登录授权(必接)

```
public void loginAuth(final Activity activity,String authCode);
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity
authCode	String	是	授权码获取方法见文档最下方

调用示例

```
SDK.getInstance().loginAuth(MainActivity.this,"ORbjEaCq");
```

4、支付(必接)

```
public void pay(final Activity activity, String orderId){}
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity
orderId	String	是	后台生成的订单号

调用示例

```
SDK.getInstance().pay(activity, "100000001");
```

5、退出(选接)

```
public void logout(final Activity activity);
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity

调用示例

```
SDK.getInstance().logout(activity);
```

6、显示悬浮窗(选接)

```
public void showFloatWindow(){}
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity

调用示例

```
@Override
public void onAttachedToWindow() {
    super.onAttachedToWindow();
    //只有activity被添加到windowmanager上以后才可以调用show方法。
   SDK.getInstance().showFloatWindow();
}
```

7、隐藏悬浮窗(选接)

```
public void hideFloatWindow(){}
```

参数说明

参数名	类型	是否必填	描述
activity	Activity	是	当前页面的Activity

调用示例

```
@Override
public void onDetachedFromWindow() {
    super.onDetachedFromWindow();
   SDK.getInstance().hideFloatWindow();
}
```

接口回调（SDKCallback说明）

1、Code值说明

code	说明
0	success
非0	failed

2、初始化完成

```
@Override
public void onInit(int code, String msg)
```

`如果code不等于0，游戏需要弹出error的提示。`

3、获取用户名完成

```
@Override
public void onLoginUserName(int code,  String msg)
```

`如果code不等于0，游戏需要弹出error的提示。其中 msg 为登录名。`

`游戏需要根据用户appid 和该用户名，向服务器请求授权码，拿到授权码后执行 loginAuth 方法。`

4、登录完成

```
@Override
public void onLogin(int code, String msg)
```

如果code不等于0，游戏需要弹出error的提示。

5、支付完成

```
@Override
public void onPay(int code, String error)
```

如果code不等于0，游戏需要弹出error的提示。

6、退出登录完成

```
@Override
public void onLogout(int code, String error)
```

注意:
1, 退出登录，游戏需退出当前账号和正在进行的游戏，并且弹出登录框。
2, 游戏退出登录如果不弹登录框，不退出进程情况下用户将没办法再次登录。

授权码获取方法：

接口地址: http://18.138.229.12:8888/appauth/login/thirdAuthCode

请求方式: POST

consumes: ["application/json"]

produces: [" */* "]

请求参数:

参数名称	参数说明	请求类型	是否必须	数据类型	schema
client_id	客户端ID	query	true	string	
client_secret	客户端验证码	query	true	string	
username	登录名，onLoginUserName 中获取	query	true	string	onLoginUserName 中获取

响应状态:

状态码	说明	schema
200	OK	响应结果«string»
201	Created	
401	Unauthorized	
403	Forbidden	
404	Not Found	

响应参数:

参数名称	参数说明	类型	schema
code	响应编码:0-请求处理成功	integer(int32)	integer(int32)
data	响应数据	string	
extra	附加数据	object	
message	提示消息	string	
path	请求路径	string	
timestamp	响应时间	integer(int64)	integer(int64)

响应示例:

```
{
    "code": 0,
    "data": "O9BUYjGO",
    "extra": {},
    "message": "success",
    "path": "",
    "timestamp": 1574218769592
}
```