**360 单机SDK安卓版接入文档**

|  |  |  |
| --- | --- | --- |
| **版本号** | **修改时间** | **内容** |
| 1002.1.4(670) | 2018-07-10 | 功能优化 |
| 1002.1.4(672) | 2018-08-03 | 功能优化 |
| 1002.1.4(674) | 2018-08-21 | 更细微博sdk（jar和so文件）  逻辑优化 |
| 1002.1.4(676) | 2018-09-13 | 优化微信支付流程  修复bug |
| 1002.1.4(678) | 2018-09-18 | 修复bug  支付流程优化 |
| 1002.1.6(680) | 2018-10-09 | 功能优化 |
| 1002.1.6(682) | 2018-10-30 | 修复bug |
| 1002.1.8(690) | 2018-11-15 | 适配安卓9.0系统 |
| 1002.1.8(692) | 2019-01-15 | 功能优化 |
| 1002.1.8(694) | 2019-03-14 | 优化支付功能  优化实名认证功能  Bug修复 |
| 1002.1.8(696) | 2019-03-29 | 精简申请权限配置；  支付bug修复；  流程优化； |
| 1002.1.8(698) | 2019-04-28 | 支付流程优化 |
| 1002.2.0(700) | 2019-06-05 | 登录和支付流程优化 |
| 1002.2.0(702) | 2019-07-04 | 登录和支付功能优化  bug修复 |
| 1002.2.0(704) | 2019-08-08 | 适配targetSdk=28 |
| 1002.2.2(710) | 2019-09-30 | 支付功能优化 |
| 1002.2.4(720) | 2019-12-20 | 实名认证功能优化 |
| 1002.2.4(722) | 2019-12-25 | bug修复 |
| 1002.2.6(730) | 2020-03-18 | 浮球和实名认证功能优化 |
| 1002.2.8(740) | 2020-04-20 | 防沉迷功能优化 |
| 1002.2.8(742) | 2020-04-30 | bug修复 |
| 1002.2.8(744) | 2020-05-20 | bug修复 |
| 1002.2.8(746) | 2020-06-01 | bug修复 |
| 1002.3.0(750) | 2020-07-09 | 游客模式支持支付防沉迷功能 |
| 1002.3.0(752) | 2020-07-23 | 功能优化 |
| 1002.3.0(754) | 2020-09-23 | 增加运营商一键登录功能  注：请向对接运营申请开通此功能 |
| 1002.3.0(756) | 2020-11-20 | 优化支付功能 |
| 1002.3.0(758) | 2021-01-05 | 优化支付功能 |
| 1002.3.2(760) | 2021-02-01 | 优化隐私权限弹窗展示时机 |
| 1002.3.2(762) | 2021-03-31 | 隐私权限弹窗支持定制化，若游戏需要定制，请联系对接运营 |
| 1002.3.2(764) | 2021-05-25 | 隐私权限弹窗优化 |
| 1002.3.2(766) | 2021-07-05 | 功能优化 |

目录

1. 重要提醒 - 1 -

【重要】单机V 1002.0.0(600)或网游V 1.8.0(540)以下版本升级到最新版必读 - 1 -

1.1 【重要】必接API说明 - 4 -

1.2 应用服务器 - 4 -

1.3 编译版本要求 - 5 -

1.4 PrivateKey计算方法 - 5 -

1.5 主Activity设置 - 5 -

1.6 Unity相关问题 - 5 -

1.7 测试须知 - 5 -

1.8 apk签名 - 6 -

1.9 apk包名 - 6 -

2. 接入流程 - 6 -

2.1 申请APPID、APPKEY和APPSECRET - 6 -

2.2 导入资源包 - 6 -

2.2.1 导入jar包 - 8 -

2.2.2 添加assets、res目录下的文件 - 8 -

2.3 配置应用工程的AndroidManifest.xml - 8 -

2.3.1 添加权限 - 8 -

2.3.2 添加activity - 10 -

2.3.3 在AndroidManifest.xml中添加meta-data - 15 -

2.3.4 targetSdk=28适配说明 - 17 -

2.4 编码接入 - 17 -

2.5 混淆编译 - 17 -

2.6 测试 - 18 -

2.7 提交 - 18 -

2.8 审核 - 18 -

2.9 上线 - 19 -

3. 编码接入详细介绍 - 19 -

3.1 初始化接口【客户端调用】（必接） - 19 -

3.2 登录授权流程 - 20 -

3.2.1 登录接口【客户端调用】（选接） - 21 -

3.2.2 切换帐号接口【客户端调用】（必接） - 22 -

3.2.3 获取用户信息【服务端调用】（选接） - 23 -

3.3 支付流程 - 24 -

3.3.1 支付接口【客户端调用】（必接） - 24 -

3.3.2 支付结果通知接口【服务端收取】（必接） - 28 -

3.3.3 订单核实接口【服务端调用】（选接） - 30 -

3.4 销毁接口【客户端调用】（必接） - 32 -

3.5 退出接口【客户端调用】（必接） - 32 -

3.6 游戏角色信息上传 (选接) - 33 -

3.6.1 游戏角色信息上传接口【客户端调用】 - 33 -

3.6.2 游戏角色信息上传接口【服务端调用】 - 37 -

3.6.3 角色信息上传自测接口【有浏览器就能测】（必测！） - 37 -

3.7 游戏activity生命周期接口【客户端调用】(必接) - 38 -

3.8 其他接口 - 38 -

3.8.1 实名状态查询接口【客户端\服务端调用】（选接） - 38 -

3.8.2 打开实名认证界面接口【客户端调用】（选接） - 40 -

3.8.3 分享接口【客户端调用】（选接） - 42 -

3.8.4 打开论坛接口【客户端调用】（选接） - 43 -

3.8.5 打开客服接口【客户端调用】（选接） - 44 -

3.8.6 使用Matrix的get方法，获取基本信息【客户端调用】（选接） - 44 -

3.8.7 游戏退出时设置杀死应用进程标志位【客户端调用】（选接） - 45 -

3.8.8 初始化前修改浮窗默认位置【客户端调用】（选接） - 45 -

3.8.9 打开浮球里的浮窗【客户端调用】（选接） - 46 -

4. 附录 - 46 -

4.1 签名算法 - 46 -

4.2 Demo工程简介 - 47 -

4.3 服务端SDK参考程序 - 48 -

4.4 常见问题 - 48 -

4.5 联系我们 - 48 -

# 1. 重要提醒

本文档面向安卓开发者。

本文档用于指导开发者快速接入360手游SDK，本SDK为安卓应用提供登录、注册、社交、支付等功能。

SDK接入请以SDK包内相应版本文档为准。

## 1.1 【重要】必接API说明

游戏提交审核时，若文档中的必接接口未接入或接入错误，**均会导致审核不通过！！请务必给予重视，以免延误游戏上线时间**！！

文档中每一个小节标题上写明【必接】二字的，为必接API，不接则无法过审。未写明【必接】二字的可根据实际需求选择接入。

**角色信息接口**为必接接口，不接入、接入之后不上传数据、上传过多空数据或上传字段不符合文档要求，均无法通过平台上线审核。**请务必接入之后调试调用此接口，然后调用角色信息自测接口，确认data中有数据且数据正确，再上传游戏包。**上传游戏包时，请提供至少3个测试使用的360帐号及密码，其中至少1个高级别帐号。

**浮球-福利页面内的切换帐号功能**为必接功能，如未接入、接入错误，均会导致审核不通过；游戏内部的切换帐号为选接功能，可根据游戏需求自行接入。

游戏若不需要支付功能，须先接入支付相关API，但不调用支付模块即可。其他功能亦是。

## 1.2 应用服务器

游戏方必须使用自己搭建的应用服务器，不可使用DEMO服务器，url(http://sdbxapp.msdk.mobilem.360.cn) 仅限DEMO示范使用，禁止正式上线游戏把DEMO应用服务器当做正式应用服务器使用。

## 1.3 编译版本要求

target版本需设置为小于或者等于28。

编译时建议选择Android 4.2或以上的版本。

## 1.4 PrivateKey计算方法

必须把APPID、APPKEY、PRIVATEKEY三个值填写在AndroidManifest文件中，**不能使用@string引用**；禁止把AppSecret保存在手机客户端，AndroidManifest中存放的是Private Key，而非App-Secret。

Private Key的算法为：QHOPENSDK\_PRIVATEKEY = MD5(appSecret + "#" + appKey)，**格式为32位小写**。

**利用自检工具计算privatekey：**打开压缩包目录:\05-自检工具及测试用例\Check\_Tools，双击cal\_private\_key.bat文件，输入游戏应用在开平申请到的appkey和secret参数，工具将自动计算privatekey，将计算得出的32位小写字符串直接黏贴到代码中即可。

## 1.5 主Activity设置

游戏（无论横竖屏，参照demo示例）必须在主Activity设置成@android:style/Theme.Translucent.NoTitleBar.Fullscreen

## 1.6 Unity相关问题

暂不支持Unity、cocos等其他引擎直接编译生成apk包，仅支持eclipse/AndroidStudio或ant编译出包。请将工程先导入eclipse/AndroidStudio再出apk包。

如有问题请尝试升级adt。

## 1.7 测试须知

**自检工具（check tools）**用来检验SDK接入是否正确，在SDK包的“05-自检工具及测试用例”文件夹内。**自检时请不要将代码混淆或者加固**，请用未加固、未混淆的代码自检。

自检常见问题及解决方法：

* 自检结果显示：提取classes.dex失败。

(1) 不应把代码混淆或加固。

(2) 没有把assets下的文件拷贝到工程下。

(3) 编译版本未达到最低要求。

(4) 缺少文件，请对比demo检查是否缺少部分文件夹

(5) 使用Unity或者其他引擎出的apk包，请先将工程导入eclipse然后导出apk包。

* 提取smali文件失败。

(1) 不应把代码混淆或加固。

(2) 因工程方法太多，需要进行拆分造成的。把sdk放到第一个dex里面即可。

1. **测试用例**是用来帮助CP遍历测试SDK接入之后功能否正常使用，在SDK包的“05-自检工具及测试用例”文件夹内。

## 1.8 apk签名

接入360sdk后，apk出包时请务必使用V1签名，360平台暂不支持V2签名和V1V2双签名。

签名错误，将导致游戏在Android 7.0设备上无法运行！请务必重视！

## 1.9 apk包名

游戏包名请务必使用.qihoo结尾。

游戏一旦在开平提交过apk包之后，包名不能修改！！第一次传包包名错误，必须重新新建游戏！请务必保证游戏包名不能变动，一旦包名和第一次提交的apk包名不一致将影响登录、支付、角色信息上传等功能！！

# 2. 接入流程

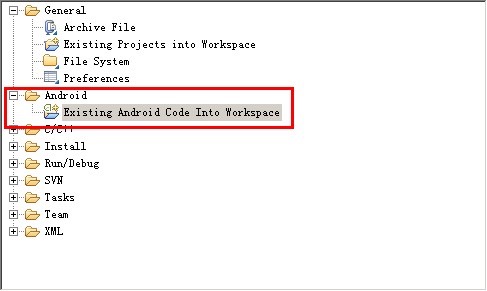


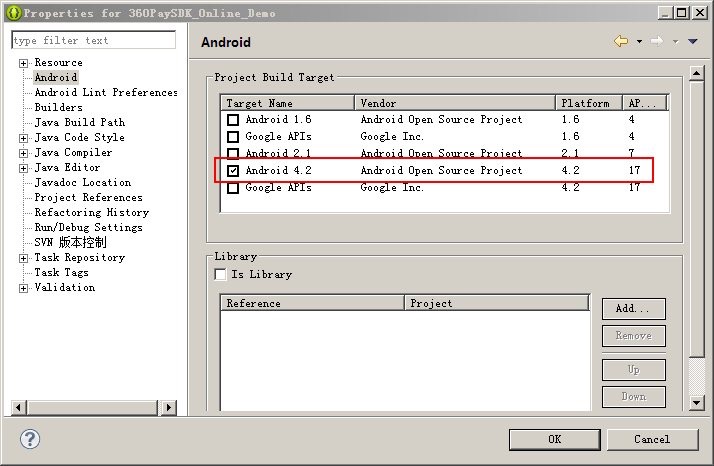
## 2.1 申请APPID、APPKEY和APPSECRET

企业开发者需要在360开放平台<http://dev.360.cn/> 申请APPID、APPKEY及APPSECRET（一个应用只能申请一个appkey）。

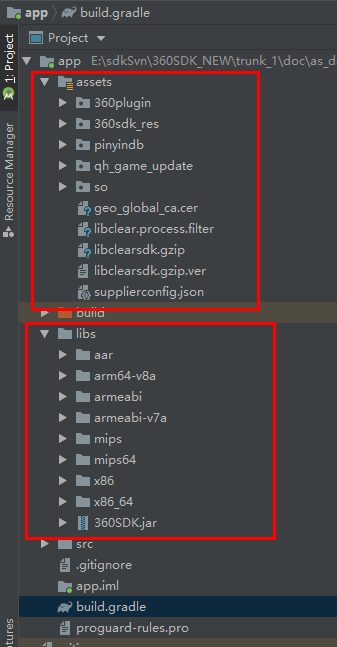
## 2.2 导入资源包

本SDK目前支持Android2.2及以上的系统版本，为兼容Android 4.0及以上系统手机，编译时请使用Android4.2 或以上版本，360SDK开发环境的配置流程如下（参考下图demo开发环境配置，导入demo工程请以utf-8编码，Android4.2导入）：





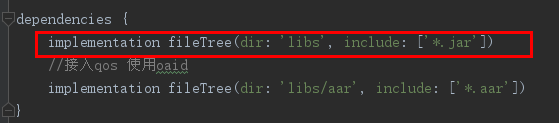
AndroidStudio环境（参考下图AndroidStudio开发环境配置，导入360SDKDemoAs.zip里的demo工程，注意该工程只是辅助接入，最新代码请参照eclipse工程代码）：



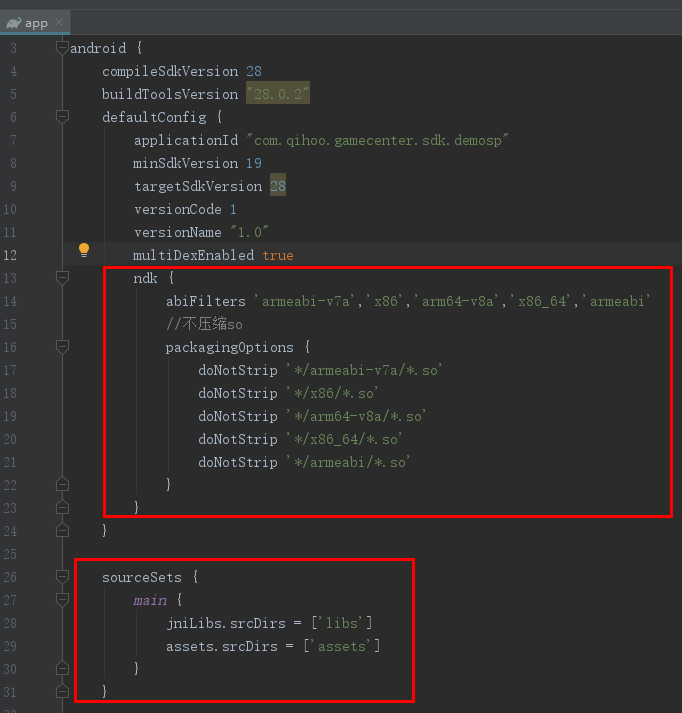
### 2.2.1 导入jar包

将SDK包内的“04-资源文件\libs\”目录下的文件（夹）放到应用工程的libs目录下。

AndroidStudio环境需要在build.gradle文件中添加：



libs下的so文件环境需要在build.gradle文件中额外添加如下配置：



### 2.2.2 添加assets、res目录下的文件

将SDK包内的“04-资源文件\assets\”目录下的文件（夹）复制到应用工程assets目录下。

将SDK包内的“04-资源文件\res\”目录下的文件（夹）复制到应用工程res目录下。

## 2.3 配置应用工程的AndroidManifest.xml

具体配置参见SDK包内的“03-demo工程”目录下的demo源代码。

### 2.3.1 添加权限

<!-- 添加360SDK必需要的权限。begin -->

<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.CHANGE\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.REQUEST\_INSTALL\_PACKAGES" />

<uses-permission android:name="android.webkit.permission.PLUGIN" />

<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />

<uses-permission android:name="android.permission.WAKE\_LOCK" />

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

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

<uses-permission android:name="android.permission.BIND\_NOTIFICATION\_LISTENER\_SERVICE" />

<uses-permission android:name="android.permission.READ\_LOGS" />

<uses-permission android:name="android.permission.BIND\_JOB\_SERVICE" />

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />

<uses-permission android:name="android.permission.GET\_ACCOUNTS" />

<uses-permission android:name="android.permission.USE\_CREDENTIALS" />

<!—运营商一键登录权限 -->

<uses-permission android:name="android.permission.GET\_TASKS" />

<uses-permission android:name="com.asus.msa.SupplementaryDID.ACCESS" />

<uses-permission android:name="android.permission.WRITE\_SETTINGS"

tools:ignore="ProtectedPermissions" />

<!-- 添加360SDK必需要的权限。end -->

### 2.3.2 添加activity

注意:必须放入<application>元素区块内

**<!-- 添加360SDK必需的activity：*com.qihoo.gamecenter.sdk.activity.ContainerActivity* -->**

<activity

android:name=*"com.qihoo.gamecenter.sdk.activity.ContainerActivity"*

android:configChanges=*"fontScale|orientation|keyboardHidden|locale|navigation|screenSize|uiMode|layoutDirection"*

android:theme=*"@android:style/Theme.Translucent.NoTitleBar"*

android:exported=*"true"*>

<!-- 支付宝签约后自动跳转到sdk配置 -->

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<action android:name=*"android.intent.action.VIEW"* />

<category android:name=*"android.intent.category.DEFAULT"* />

<category android:name=*"android.intent.category.BROWSABLE"* />

<!-- android host的值必须为游戏的包名 -->

<data android:host=*"游戏包名"*/>

<!-- android scheme的值必须不能变-->

<data android:scheme=*"qihooonlinepay"* />

</intent-filter>

</activity>

<!-- payment activities begin -->

<!--添加360SDK必需的activity：com.qihoopp.qcoinpay.QcoinActivity -->

<activity

android:name="com.qihoopp.qcoinpay.QcoinActivity"

android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|uiMode"

android:theme="@android:style/Theme.Translucent.NoTitleBar"

android:windowSoftInputMode="stateAlwaysHidden|adjustResize" >

</activity>

<!--alipay sdk begin -->

<activity

android:name="com.alipay.sdk.app.H5PayActivity"

android:screenOrientation="portrait" >

</activity>

<!--alipay sdk end -->

<!-- 微信支付界面-->

<!-- 汇付宝SDK 需添加 -->

<activity

android:name="com.heepay.plugin.activity.HyNotityActivity"

android:configChanges="orientation|keyboardHidden|screenSize"

android:screenOrientation="behind"

android:theme="@android:style/Theme.Translucent.NoTitleBar"/>

<!-- 以下Activity是SDK插件化使用的代理Activity -->

<activity

android:name="com.qihoo.sdkplugging.host.activity.HostProxyActivity"

android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|uiMode|layoutDirection"

android:theme="@android:style/Theme.Translucent.NoTitleBar"

android:windowSoftInputMode="adjustPan"

android:label="360SDK" >

<intent-filter>

<action android:name="com.qihoo.sdkplugging.host.proxy.activity.VIEW" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

<!-- payment activities end -->

**<!-- 以下是360游戏新版本实时推送PUSH SDK声明，除了标红的，其他不可修改 -->**

<service

android:name="com.qihoo.pushsdk.cx.PushService"

android:exported="false"

android:process=":PushClient">

</service>

<service

android:name="com.qihoo.pushsdk.cx.PushService$InnerService"

android:exported="false"

android:process=":PushClient"/>

<service

android:name="com.qihoo.pushsdk.keepalive.PushJobService"

android:permission="android.permission.BIND\_JOB\_SERVICE"/>

<service android:name="com.qihoo.pushsdk.cx.PushLocalService"/>

<service

android:name="com.qihoo.pushsdk.keepalive.NotificationService"

android:exported="false"

android:permission="android.permission.BIND\_NOTIFICATION\_LISTENER\_SERVICE">

<intent-filter>

<action android:name="android.service.notification.NotificationListenerService" />

</intent-filter>

</service>

<receiver android:name="com.qihoo.pushsdk.keepalive.PushWakeUpReceiver">

<intent-filter>

<action android:name="android.net.conn.CONNECTIVITY\_CHANGE" />

<action android:name="android.intent.action.BOOT\_COMPLETED" />

<action android:name="android.intent.action.MEDIA\_MOUNTED" />

<action android:name="android.intent.action.MEDIA\_UNMOUNTED" />

<action android:name="android.intent.action.SCREEN\_OFF" />

<action android:name="android.intent.action.SCREEN\_ON" />

<action android:name="android.intent.action.USER\_PRESENT" />

<action android:name="android.intent.action.PACKAGE\_ADDED" />

<action android:name="android.intent.action.PACKAGE\_REMOVED" />

<action android:name="android.intent.action.SIG\_STR" />

</intent-filter>

</receiver>

<service android:name="com.qihoo.pushsdk.QPushHandleService" />

**<!-- push sdk 新增 必须是包名 -->**

<provider

android:name="com.qihoo.pushsdk.sphelper.SPContentProvider"

android:authorities="**包名.**sdk.sp.helper"

android:exported="false"

android:process=":PushClient" />

<service android:name="com.qihoo.qpush.report.network.SurvivalService" />

<receiver

android:name="com.qihoo.pushsdk.QPushMessageReceiver"

android:exported="false">

<intent-filter>

<action android:name="com.qihoo.qpush.INFO\_RECEIVE" />

</intent-filter>

</receiver>

**<!-- 360游戏新版本实时推送PUSH SDK声明 end -->**

<service

android:name="com.qihoo360.qos.QosService"

android:exported="false"

android:process=":qos" >

<intent-filter>

<action android:name="com.qihoo360.qos.QosService" />

</intent-filter>

</service>

<!—移动运营商登录activity-->

<activity

android:name="com.cmic.sso.sdk.activity.LoginAuthActivity"

android:launchMode="singleTop"

android:configChanges="orientation|keyboardHidden|screenSize"

android:theme="@style/cmTheme"/>

<!--微信相关的activity，请直接使用demo中的WXEntryActivity类的代码实现，

类的全名为：“应用包名.wxapi.WXEntryActivity”。

**注意**

**1、微信appid是必须要有的参数，用于微信登录和分享功能，在微信官网申请，不需要的话可以随便填一个，不调用即可。**

**2、如果使用微信登录功能，在Manifest中配置QHOPENSDK\_WEIXIN\_APPID字段，同时需发送邮件至shouyou360@qq.com，邮件需提供：游戏名称、游戏在360平台申请的appkey、游戏在微信申请的AppId和AppSecret。**

**3、微信登录WXEntryActivity请务必按照demo示例来进行接入。**

Manifest中的QHOPENSDK\_WEIXIN\_APPID 字段可以先配成默认值，后续申请到微信的appid可以联系我们在服务端配置。

**注意2：不要修改此actitiy的android:name属性**。

例如：如果游戏的包名为com.a.b.c，那么应该将demo中的WXEntryActivity.java放到

“${工程目录}/src/com/a/b/c/wxapi/WXEntryActivity.java”这个路径下。

并将此java文件中的第一行package声明修改为“package com.a.b.c.wxapi;”

-->

<activity

android:name=".wxapi.WXEntryActivity"

android:label="@string/demo\_app\_name"

android:theme="@android:style/Theme.Translucent.NoTitleBar"

android:exported="true" />

<!-- QQ登录相关的Activity 。

**注意：**

**1、qq登录为选接内容，如果不接入，则不用添加此部分activity内容**

**2、如果使用qq登录功能，在Manifest中配置QHOPENSDK\_QQAPPID字段，同时需发送邮件至shouyou360@qq.com，邮件需提供：游戏名称、游戏在360平台申请的appkey、游戏在QQ开放平台申请的AppId。**

-->

<activity

android:name="com.tencent.tauth.AuthActivity"

android:launchMode="singleTask"

android:noHistory="true" >

<intent-filter>

<action android:name="android.intent.action.VIEW" />

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.BROWSABLE" />

<data android:scheme="**tencent101434511**" />

</intent-filter>

</activity>

<activity

android:name="com.tencent.connect.common.AssistActivity"

android:configChanges="orientation|keyboardHidden"

android:screenOrientation="behind"

android:theme="@android:style/Theme.Translucent.NoTitleBar" />

<!-- QQ登录相关activity结束 -->

<!-- UpdateLib start -->

<service

android:name="com.qihoo.appstore.updatelib.CheckUpdateService"

android:exported="false" />

<activity android:name="com.qihoo.updatesdk.lib.UpdateTipDialogActivity"

android:configChanges="keyboardHidden|orientation|screenSize"

android:exported="false"

android:screenOrientation="behind"

android:theme="@android:style/Theme.Translucent.NoTitleBar"/>

<!-- UpdateLib end -->

<!-- 360 sdk 内部 service、activity start -->

<service

android:name="com.qihoo.sdkplugging.host.service.CommonHostProxyService"

android:persistent="true"

android:process= ":gcpasremote">

<intent-filter>

<action android:name="com.qihoo.sdkplugging.host.service.CommonHostProxyService"/>

</intent-filter>

</service>

<!-- 360 sdk 内部 service、activity end -->

<provider

android:name="android.support.v4.content.FileProvider"

android:authorities="**您的游戏包名.**FileProvider"

android:exported="false"

android:grantUriPermissions="true"

tools:replace="android:authorities">

<meta-data

android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths"

tools:replace="android:resource" />

</provider>

<!-- 应用内拦截下载，执行下载的service start -->

<service android:name="com.qihoo.gamecenter.sdk.hook.DownloadService"

android:launchMode="singleTask"> </service>

<!-- 应用内拦截下载，执行下载的service end -->

### 2.3.3 在AndroidManifest.xml中添加meta-data

注意：必须放入<application>元素区块内:

**<!-- 添加360SDK必需的meta-data：QHOPENSDK\_APPID。此处value为APPID。请在360应用开放平台注册申请-->**

<meta-data

android:name="QHOPENSDK\_APPID"

android:value="102094835">

</meta-data>

**<!--添加360SDK必需的meta-data：QHOPENSDK\_APPKEY。此处value为APPKEY。请在360应用开放平台注册申请-->**

<meta-data

android:name="QHOPENSDK\_APPKEY"

android:value="8689e00460eabb1e66277eb4232fde6f">

</meta-data>

**<!--必需的meta-data：QHOPENSDK\_PRIVATEKEY。此处value为PRIVATEKEY不是APPSECRET，而是md5(app\_secret +”#”+ app\_key)，全小写，APPSECRET不允许保存在客户端！详见**[1.4 PrivateKey计算方法](#_1.4_PrivateKey计算方法)**-->**

<meta-data

android:name="QHOPENSDK\_PRIVATEKEY"

android:value="4e04fe9ac8e2a73cbb27ba52ac076eb9">

</meta-data>

**<!—注意**

**1、微信appid是必须要有的参数，用于微信登录和分享功能，在微信官网申请，不需要的话可以随便填一个，不调用即可。**

**2、如果使用微信登录功能，在Manifest中配置QHOPENSDK\_WEIXIN\_APPID字段，同时需发送邮件至shouyou360@qq.com，邮件需提供：游戏名称、游戏在360平台申请的appkey、游戏在微信申请的AppId和AppSecret。**

**3、微信登录WXEntryActivity请务必按照demo示例来进行接入。-->**

**<!-- 从微信分享相关的meta-data：QHOPENSDK\_WEIXIN\_APPID。此处value为在微信开放平台申请的APPID。请在微信开放平台申请 -->**

<meta-data

android:name="QHOPENSDK\_WEIXIN\_APPID"

android:value="wx38c06d349170365c" >

</meta-data>

**<!—注意**

**1、QQ 登录为选接内容，如果不需要的话，可以不添加此appid。**

**2、如果使用qq登录功能，在Manifest中配置QHOPENSDK\_QQAPPID字段，同时需发送邮件至shouyou360@qq.com，邮件需提供：游戏名称、游戏在360平台申请的appkey、游戏在QQ开放平台申请的AppId。**

**<!—与QQ登录相关的meta-data：QHOPENSDK\_QQAPPID。此处value为在QQ开放平台申请的APPID。请在QQ开放平台申请 -->**

<meta-data

android:name="QHOPENSDK\_QQAPPID"

android:value="101434511" >

</meta-data>

**<!-- 默认参数，不需要修改，直接复制就行 -->**

<meta-data android:name="DC\_APPKEY" android:value="02522a2b2726fb0a03bb19f2d8d9524d"/>

<meta-data android:name="APP\_KEY" android:value="n7a5gpvhzkny" />

<meta-data android:name="APP\_SECRET" android:value="ggm5oqotrjs8spkgwk5qjw5447bj7zl5" />

**<!-- 默认参数，不需要修改，直接复制就行，此段代码为可选。作用是适配安卓全面屏手机，比如小米mix系列。如果不加此段代码，会出现SDK的悬浮球在横屏游戏模式下不能贴边的问题。如果加入此段代码，那么你的游戏比例会被强制拉伸到全面屏的比例。比如正常手机都是16:9的屏幕比例，而有的全面屏是18:9，那么加入了此段代码，你的游戏画面就会被拉伸到18:9 。如果不加，则不会被拉伸。所以你需要根据自己的游戏适配画面比例来判断是否需要加。-->**

<meta-data android:name="android.max\_aspect" android:value="2.1" />

### 2.3.4 targetSdk=28适配说明

从2.2.0(704)开始，SDK对targetSdk=28进行了适配。如果开发者也适配了targetSdk=28，则需要添加如下内容：

1、在<application/>根标签添加（**二选一**）：android:usesCleartextTraffic="true"**或者**  
android:networkSecurityConfig="@xml/network\_security\_config"  
其中network\_security\_config的文件内容可以直接复制使用SDK demo工程路径下res/xml/ network\_security\_config.xml文件.

2、在<application>元素区块内添加：

<!-- 防止target为28时在android P手机上关于apache http报错 -->

**<uses-library android:name=*"org.apache.http.legacy"* android:required=*"false"*/>**

## 2.4 编码接入

详见编码接入详细介绍

## 2.5 混淆编译

如果要混淆java代码，请不要混淆联编的jar包中的类。可以添加以下类到proguard配置，排除在混淆之外：

-keep class cn.pp.\*\* { \*; }

-keep class cn.pp.\*\* { \*; }

-keep class org.json.alipay.\*\* { \*; }

-keep class com.alipay.\*\* {\*;}

-keep class com.nearme.atlas.\*\* {\*;}

-keep class com.qihoo.\*\* {\*;}

-keep class com.qihoo360.\*\* { \*; }

-keep class com.qihoopp.\*\* { \*; }

-keep class com.qihoosdk.\*\* {\*;}

-keep class qihoohttp.\*\* {\*;}

-keepnames class qihoohttp.\*\* {\*;}

-keep class com.heepay.plugin.\*\*{\*;}

-keepnames class com.heepay.plugin.\*\* {\*;}

-keep class com.yeepay.safekeyboard.\*\* { \*; }

-keep class com.ipaynow.\*\* {\*;}

-keep class com.ta.utdid2.\*\* {\*;}

-keep class com.ut.device.\*\* {\*;}

-keep class com.tencent.\*\* {\*;}

-keepnames class okhttp3.internal.publicsuffix.PublicSuffixDatabase

-keep class okhttp3.\*\* {\*;}

-keepnames class okhttp3.\*\* {\*;}

-keep class com.cloudplay.\*\*{\*;}

-keep class com.qihoo.antispam.holmes.\*\*{\*;}

-keep class com.qihoo360.qos.\*\*{ \*;}

-keep class com.qihoo.gamecenter.sdk.common.stat.qos.\*\*{\*;}

-keep class com.qihoo360.qos.staticmsa.\*\* {\*;}

-keep class com.bun.miitmdid.core.\*\* {\*;}

-keep class com.bun.supplier.\*\* {\*;}

-dontwarn com.cmic.sso.sdk.\*\*

-keep class com.cmic.\*\* {\*;}

-keep class com.cmic.sso.sdk.\*\* {\*;}

-keep class cn.com.chinatelecom.\*\*{\*;}

-keep class cn.com.chinatelecom.account.api.\*\*{\*;}

-keep class com.unicom.sdklibrary.\*\* {\*;}

-keep interface com.unicom.sdklibrary.ResultListener {\*;}

-keep class com.qihoo.gamecenter.sdk.common.isp.\*\*{\*;}

-keep class com.qihoo.gamecenter.sdk.common.view.PrivacyModel { \*;}

-keep class com.qihoo.gamecenter.sdk.common.view.PrivacyName { \*;}

<!—重要： webview兼容5.0以上 -->

-keepclassmembers class \* extends android.webkit.WebChromeClient {

public void openFileChooser(...);

public boolean onShowFileChooser(...);

}

关闭混淆警告可以使用-ignorewarnings参数，或者使用如下配置只关闭SDK类的混淆警告：

-dontwarn cn.pp.\*\*

-dontwarn com.qihoo.\*\*

-dontwarn com.qihoo360.\*\*

-dontwarn com.qihoopp.\*\*

-dontwarn com.qihoosdk.\*\*

-dontwarn qihoohttp.\*\*

-dontwarn com.alipay.\*\*

-dontwarn com.alipay.android.app.\*\*

-dontwarn com.yeepay.safekeyboard.\*\*

-dontwarn com.heepay.plugin.\*\*

-dontwarn okhttp3.\*\*

-dontwarn okio.\*\*

-dontwarn javax.annotation.\*\*

-dontwarn org.conscrypt.\*\*

-dontwarn android.\*\*

-dontwarn java.net.\*\*

-dontwarn com.qihoo.antispam.holmes.\*\*

-dontwarn com.bun.miitmdid.core.\*\*

-dontwarn com.bun.supplier.\*\*

## 2.6 测试

1. 请使用自测工具检查接入完整性；
2. 在 SDK 包中提供了测试用例，用于功能点测试。

## 2.7 提交

经过测试后的 app，请在360移动开放平台http://dev.360.cn/提交 apk 文件

## 2.8 审核

应用提交审核后，360平台会将审批结果以邮件形式进行反馈。

客服电话：010-58781044

客服邮箱：360box@360.cn

## 2.9 上线

应用通过审核后将在1个小时后发布上线。

# 3. 编码接入详细介绍

## 3.1 初始化接口【客户端调用】（必接）

* 在Application中的OnCreate中初始化SDK

游戏在Application中的OnCreate中必须调用360手游SDK的初始化，否则360手游SDK将无法正常工作，请参见Demo，如下图：



* 在应用主Activity中，使用Matrix的setActivity方法（注：一定要在主线程调用）

public static void setActivity (final Activity activity, MatrixCallBack matrixCallBack, boolean debugMode)用于加载360SDK功能，在应用主Activity的onCreate()函数中必须调用1次该方法，否则360SDK处于未初始化状态，无法使用其他接口。

当360SDK处于未初始化状态时，调用其任何接口都会返回错误，错误码为-101。

**参数：**

activity：activity对象

matrixCallBack: 开发者实现该回调接口，达到切换帐号功能，请务必接入。（详见demo源码）

debugMode: 360SDK内部提供了自动检测SDK接入是否完整的功能和输出调试LOG的功能，默认为 false，这两项功能关闭，true则打开这两个功能，上线前请务必设置为 false。（570版本添加的参数, 详见demo源码）

**使用示例：**

protected MatrixCallBack mSDKCallback = new MatrixCallBack() {

@Override

public void execute(Context context, int functionCode, String functionParams) {

if (functionCode == ProtocolConfigs.FUNC\_CODE\_SWITCH\_ACCOUNT) {

doSdkSwitchAccount(getLandscape(context));

}else if (functionCode == ProtocolConfigs.FUNC\_CODE\_INITSUCCESS) {

//这里返回成功之后才能调用SDK 登录接口

}else if (functionCode == ProtocolConfigs.FUNC\_CODE\_LOGIN){

// sdk 浮窗内登录的回调，具体参见demo示

}else if (functionCode == ProtocolConfigs.FUNC\_CODE\_LOGINAFTER\_REALNAME\_CALLBACK){

// 当收到此回调后才可调用SDK提供的实名状态查询接口、打开实名认证界面接口。具体的返回内容是functionParams，数据格式如下。

}

}

};

Matrix.setActivity(this, mSDKCallback, false);

**备注**：实名认证回调数据格式说明

（1）正确的数据格式，会返回实名状态 0是未实名、1是已实名未成年、2是已实名已成年

{

"error\_code": "0",

"error\_msg": "请求成功",

"content": {

"ret": [{

"qid": "3190816167",

"status": "2"

}]

}

}

（2）错误的数据格式，当遇到以下两个错误码的时候，如果游戏需要实名状态信息，请主动调用SDK提供的接口查询用户实名状态。 {

"error\_code": -3001,

"error\_msg": "实名查询关闭，请手动调用查询接口查询"

}

{

"error\_code": -3002,

"error\_msg": "实名查询异常，请手动调用查询接口查询"

}

## 3.2 登录授权流程

360开放平台的登录流程使用Oauth2协议标准授权流程。

**登录流程**

1. 应用客户端调用SDK进行登录 (见本章的接口介绍)；
2. 360SDK与360 服务器通信进行用户登录，返回登录结果及用户信息；
3. 360SDK把登录结果返给应用客户端，登录完成。

**注意事项**

1. 360SDK登录接口返回的用户信息中没有qid。如果游戏想要获取qid，需要通过应用服务器端向360服务器端发起请求，用access token换取qid，具体调用方式见『获取用户信息接口』。
2. 如果用户重新登陆，会获取到新的token，原token即失效。用户若用同样帐号在不同设备上登录同一游戏, 只有最后一次登录获取的token是最终有效的。
3. 调用任何其他需要登录后才能调用的接口时，如果360SDK未处于登录状态，会直接返回错误，错误码为-100。
4. 调用注销接口或销毁接口会让360SDK回到未登录的状态。

### 3.2.1 登录接口【客户端调用】（选接）

**功能说明：**

展示登录界面, 让用户登录。若登录成功，返回errno为0并带有用户信息。

登录成功SDK会返回结果，登录失败会停留在登录界面，除非用户按返回键取消登录尝试。

注意：1）**调用登录接口必须在初始化接口callback返回值后调用（**[**详见初始化接口调用示例**](#_3.1_初始化接口【客户端调用】（必接）)**）；**

2）游戏方获取用户信息后，需要保存自身帐号与360帐号的绑定关系。

**接口示例：**

/\*\*

\* 使用360SDK的登录接口, 生成intent参数

\*

\* @param isLandScape 是否横屏显示登录界面

\*/

**private Intent** getLoginIntent(boolean isLandScape) {

Intent intent = new Intent(this, ContainerActivity.class);

// 必需参数，使用360SDK的登录模块

intent.putExtra(ProtocolKeys.FUNCTION\_CODE,

ProtocolConfigs.FUNC\_CODE\_LOGIN);

// 可选参数，360SDK界面是否以横屏显示，默认为true，横屏

intent.putExtra(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE,

isLandScape);

//-- 以下参数仅仅针对自动登录过程的控制

// 可选参数，自动登录过程中是否不展示任何UI，默认展示。

intent.putExtra(ProtocolKeys.IS\_AUTOLOGIN\_NOUI,

getCheckBoxBoolean(R.id.isAutoLoginHideUI));

return intent;

}

// 调用接口

**protected** **void** doSdkLogin(boolean isLandScape) {

mIsInOffline = false;

Intent intent = getLoginIntent(isLandScape);

IDispatcherCallback callback = mLoginCallback;

if (getCheckBoxBoolean(R.id.isSupportOffline)) {

callback = mLoginCallbackSupportOffline;

}

Matrix.execute(this, intent, callback);

}

**返回数据格式：**

{

"data": {

"expires\_in": "36000",

"scope": "",

"refresh\_token": "",

"access\_token": "6461171100c3bfa3cba24cc332d7d78b311d2bf590f4877c9" // token

},

"errno": 0

}

**callback示例：**

// 登录、注册的回调

privateIDispatcherCallback mLoginCallback = newIDispatcherCallback() {

@Override

publicvoid onFinished(String data) {

// press back

if (isCancelLogin(data)) {

return;

}

// 显示一下登录结果

Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH\_LONG).show();

mIsInOffline = false;

mQihooUserInfo = null;

// 解析access\_token

mAccessToken = parseAccessTokenFromLoginResult(data);

if (!TextUtils.isEmpty(mAccessToken)) {

// 需要去应用的服务器获取用access\_token获取一下用户信息

getUserInfo();

} else {

Toast.makeText(SdkUserBaseActivity.this, "get access\_token failed!",

Toast.LENGTH\_LONG).show();

}

}

};

### 3.2.2 切换帐号接口【客户端调用】（必接）

**功能说明：**

**浮球-福利页面内的切换帐号功能**为必接功能，如未接入、接入错误，均会导致审核不通过；

游戏内部的切换帐号为选接功能，可根据游戏需求，自行在游戏的菜单中添加“切换帐号”的入口，方便用户切换帐号。

应用调用360SDK切换帐号接口, 360SDK显示登录页面，用户可更换帐号进行登录。之后的登录流程和返回结果与登录接口一样。

**示例代码：**

/\*\*

\* 使用360SDK的切换帐号接口

\*

\* @param isLandScape 是否横屏显示登录界面

\*/

**protected void** doSdkSwitchAccount(boolean isLandScape) {

Intent intent = getSwitchAccountIntent(isLandScape);

Matrix.invokeActivity(this, intent, mAccountSwitchCallback);

}

**intent参数说明：**

除FUNCTION\_CODE外，其他同登录接口

intent.putExtra(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_SWITCH\_ACCOUNT);

**callback的 json格式说明：**

同登录接口

### 3.2.3 获取用户信息【服务端调用】（选接）

应用服务端获取access token后，可调用360开放平台服务器端接口/user/me，获取360用户id以及其它用户信息。

获取用户信息后，应用需要保存自身帐号与360帐号的绑定关系，请务必使用360用户ID作为绑定关系的标示。并且妥善保存用户信息留待以后使用。请求该接口时，请务必设置超时时间，建议超时时间设置在5秒以下，游戏方可根据自身需求合理设置超时时间。

接口地址为：https://openapi.360.cn/user/me.json

**参数说明：**

|  |  |  |
| --- | --- | --- |
| 参数 | 必选 | 参数说明 |
| access\_token | Y | 授权的access token |
| fields | N | 允许应用自定义返回字段，多个属性之间用英文半角逗号作为分隔符。不传递此参数则缺省返回id,name,avatar |

返回参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 必选 | 参数说明 |
| id | Y | 360用户ID, 缺省返回 |
| name | Y | 360用户名, 缺省返回 |
| avatar | Y | 360用户头像, 缺省返回 |
| sex | N | 360用户性别，仅在fields中包含时候才返回,返回值为：男，女或者未知 |
| area | N | 360用户地区，仅在fields中包含时候才返回 |
| nick | N | 用户昵称，无值时候返回空 |

**请求示例：**

https://openapi.360.cn/user/me.json?access\_token=12345678983b38aabcdef387453ac8133ac3263987654321&fields=id,name,avatar,sex,area

**返回示例：**

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

{

"id": "201459001",

"name": "360U201459001",

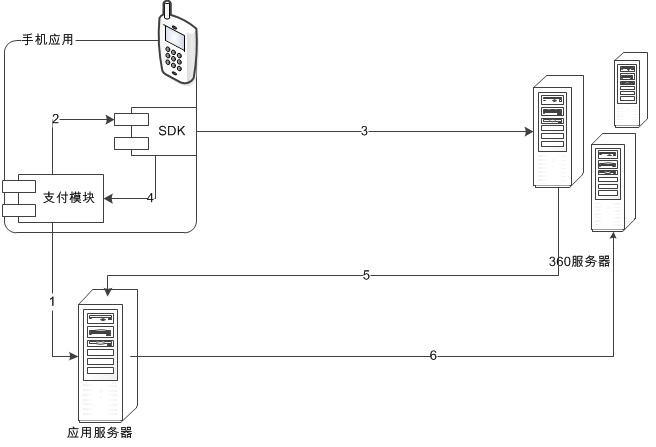
"avatar": "http://u1.qhimg.com/qhimg/quc/...ed6e9c53543903b",

"sex": "未知"

"area": ""

}

## 3.3 支付流程



1. 应用调用应用服务器进行下单（此部分视游戏自身实际情况）；
2. 应用调用360SDK支付接口进行下单；
3. 360SDK根据应用下单时所选支付模块类型调用不同支付方式并引导用户完成支付；
4. 支付完成或失败、终止后，360SDK客户端会返回支付结果给应用客户端的支付模块；
5. 支付成功后，360服务器将支付结果发送至应用客户端下单时所填写的服务端支付结果通知地址以通知支付结果；
6. 应用服务器调用360服务器端订单确认接口，验证支付通知的合法性(可选)。

### 3.3.1 支付接口【客户端调用】（必接）

**功能说明：**

**对于未联网且安装SIM卡用户，建议调用自行申请的运营商派生计费，其余情况建议一律调用此接口进行计费。**

应用调用360SDK支付接口时，360SDK根据应用下单信息通过360聚合收银台或应用所选支付方式发起支付。

360SDK通知用方发放道具时，会返回应用订单号,同时提供360订单号。

**接口示例：**

**注意：**

1. **必选参数不能为空, 不能为0，否则支付失败。**
2. **参数名，以ProtocolKeys中定义的常量为准。**
3. **请务必对case 0、1、-1加入处理语句，如果为空会导致游戏崩溃。**

**protected void** doSdkPay(QihooUserInfo usrinfo, **boolean** isLandScape,**int** functionCode) {

if (!checkLoginInfo(usrinfo)) {

return;

}

if(!isAccessTokenValid) {

Toast.makeText(SdkUserBaseActivity.this, R.string.access\_token\_invalid, Toast.LENGTH\_SHORT).show();

return;

}

if(!isQTValid) {

Toast.makeText(SdkUserBaseActivity.this, R.string.qt\_invalid, Toast.LENGTH\_SHORT).show();

return;

}

// 支付基础参数

QihooPayInfo payInfo = getQihooPayInfo(functionCode);

Intent intent = getPayIntent(isLandScape, payInfo, functionCode);

// 必需参数，使用360SDK的支付模块:CP可以根据需求选择使用 带有收银台的支付模块 或者 直接调用微信支付模块或者直接调用支付宝支付模块。

//functionCode 对应三种类型的支付模块：

//ProtocolConfigs.FUNC\_CODE\_PAY;// 360聚合支付模块。（有收银台，基于各类因素推荐用户使用每笔订单支付最便捷的支付方式，常规手游可以此方式接入支付，以最小开发成本快速接入上线）

//ProtocolConfigs.FUNC\_CODE\_WEIXIN\_PAY;//微信支付模块。（无收银台，直接调用微信发起支付，用户设备中需安装微信客户端）

//ProtocolConfigs.FUNC\_CODE\_ALI\_PAY;//支付宝支付模块。（无收银台，直接调用支付宝发起支付，用户设备中可不安装支付宝客户端）

intent.putExtra(ProtocolKeys.FUNCTION\_CODE, functionCode);

Matrix.invokeActivity(this, intent, mPayCallback);

}

**protected** Intent getPayIntent(**boolean** isLandScape, QihooPayInfo pay,**int** functionCode) {

Bundle bundle = **new** Bundle();

// 界面相关参数，360SDK界面是否以横屏显示。

bundle.putBoolean(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE, isLandScape);

// \*\*\* 以下非界面相关参数 \*\*\*

// 设置QihooPay中的参数。

//可选参数，360帐号id。

//如果CP在启动支付接口时传递了该字段，则表示：调用的是有登录状态的支付

// 如果CP在启动支付接口时未传递该字段，则表示：调用的是未登录状态的支付

bundle.putString(ProtocolKeys.QIHOO\_USER\_ID, pay.getQihooUserId());

// 必需参数，所购买商品金额，以分为单位，最小金额1分。

bundle.putString(ProtocolKeys.AMOUNT, pay.getMoneyAmount());

// 必需参数，所购买商品名称，应用指定，建议中文，最大10个中文字。

bundle.putString(ProtocolKeys.PRODUCT\_NAME, pay.getProductName());

// 必需参数，购买商品的商品id，应用指定，最大36中文字。

bundle.putString(ProtocolKeys.PRODUCT\_ID, pay.getProductId());

// 必须参数，应用方提供的支付结果通知uri，最大255字符。360服务器将把支付接口回调给该uri，具体协议请查看文档中，支付结果通知接口–应用服务器提供接口，若无应用服务端请填写空字符串，不可为null。

bundle.putString(ProtocolKeys.NOTIFY\_URI, pay.getNotifyUri());

// 必需参数，游戏或应用名称，最大16中文字。

bundle.putString(ProtocolKeys.APP\_NAME, pay.getAppName());

// 必需参数，应用内的用户名，如游戏角色名。 若应用内绑定360帐号和应用帐号，则可用360用户名，最大16中文字。（充值不分区服，充到统一的用户账户，各区服角色均可使用）。

bundle.putString(ProtocolKeys.APP\_USER\_NAME, pay.getAppUserName());

// 必需参数，应用内的用户id。

// 若应用内绑定360帐号和应用帐号，充值不分区服，充到统一的用户账户，各区服角色均可使用，则可用360用户ID最大32字符。

bundle.putString(ProtocolKeys.APP\_USER\_ID, pay.getAppUserId());

// 可选参数，应用扩展信息1，原样返回，最大255字符。

bundle.putString(ProtocolKeys.APP\_EXT\_1, pay.getAppExt1());

// 可选参数，应用扩展信息2，原样返回，最大255字符。

bundle.putString(ProtocolKeys.APP\_EXT\_2, pay.getAppExt2());

// 必需参数，应用订单号，应用内必须唯一，最大32字符。应用方需要生成自己的订单号app\_order\_id，应用订单号不能重复提交，并且一个应用订单不管是否支付成功，都仅可支付一次，以避免重复支付。若游戏无服务端，此订单号可通过年月日时分秒+设备号等随机数生成即可。

bundle.putString(ProtocolKeys.APP\_ORDER\_ID, pay.getAppOrderId());

// 必需参数，使用360SDK的支付模块:CP可以根据需求选择使用 带有收银台的支付模块 或者 直接调用微信支付模块或者直接调用支付宝支付模块。

//functionCode 对应三种支付模块：

//ProtocolConfigs.FUNC\_CODE\_PAY;//表示 带有360收银台的支付模块。

//ProtocolConfigs.FUNC\_CODE\_WEIXIN\_PAY;//表示 微信支付模块。

//ProtocolConfigs.FUNC\_CODE\_ALI\_PAY;//表示支付宝支付模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE,functionCode);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

return intent;

}

**callback的 json数据格式：**

成功返回

{error\_code: 0, error\_msg: "支付成功", content:""}

失败返回

{error\_code: 1, error\_msg: "支付失败", content:""}

取消返回

{error\_code: -1, error\_msg: "支付取消", content:""}

支付正在进行

{error\_code: -2, error\_msg: "正在进行", content:""}

access\_token失效

{error\_code: 4010201, error\_msg: " token已失效", content:""}

QT失效

{error\_code: 4009911, error\_msg: " 登录已失效", content:""}

**callback示例：**

/\*\*

\* 支付的回调

\*/

**protected** IDispatcherCallback mPayCallback = new IDispatcherCallback() {

@Override

public void onFinished(String data) {

Log.d(TAG, "mPayCallback, data is " + data);

if(TextUtils.isEmpty(data)) {

return;

}

boolean isCallbackParseOk = false;

JSONObject jsonRes;

try {

jsonRes = new JSONObject(data);

// error\_code 状态码： 0 支付成功， -1 支付取消， 1 支付失败， -2 支付进行中。

**// 请务必对case 0、1、-1、-2加入处理语句，如果为空会导致游戏崩溃。若应用有支付服务端，则需以360服务端通知给应用服务端的结果进行道具发放；若应用无服务端，则0、-2需发放道具；-1、1无需发放道具。**

// error\_msg 状态描述

int errorCode = jsonRes.optInt("error\_code");

isCallbackParseOk = true;

switch (errorCode) {

case 0:

case 1:

case -1:

case -2: {

isAccessTokenValid = true;

String errorMsg = jsonRes.optString("error\_msg");

String text = getString(R.string.pay\_callback\_toast, errorCode, errorMsg);

Toast.makeText(SdkUserBaseActivity.this, text, Toast.LENGTH\_SHORT).show();

}

break;

case 4010201:

isAccessTokenValid = false;

Toast.makeText(SdkUserBaseActivity.this, R.string.access\_token\_invalid, Toast.LENGTH\_SHORT).show();

break;

case 4009911:

//QT失效

isQTValid = false;

Toast.makeText(SdkUserBaseActivity.this, R.string.qt\_invalid, Toast.LENGTH\_SHORT).show();

break;

default:

break;

}

} catch (JSONException e) {

e.printStackTrace();

}

// 用于测试数据格式是否异常。

if (!isCallbackParseOk) {

Toast.makeText(SdkUserBaseActivity.this, getString(R.string.data\_format\_error),

Toast.LENGTH\_LONG).show();

}

}

};

### 3.3.2 支付结果通知接口【服务端收取】（必接）

**以下服务端调整兼容接入各版本SDK的游戏包，除单机SDK所产生的免登录支付订单外，各版本游戏产生的订单均请按以下方式修改。**

1. 应用客户端调用支付接口时，需指定支付结果的通知回调地址notify\_uri。支付完成后，360服务器会把支付结果以GET方式通知到此地址(建议应用服务端接口同时支持GET和POST)。
2. **应用接收验证参数后，必须检查360SDK服务端支付结果通知接口中的以下参数与下单时游戏方记录的参数一致性，防止出现篡改。**

需校验参数：应用app key；360帐号id；应用分配给用户的id；应用自定义的商品id；商品金额；应用扩展信息1；应用扩展信息2

* 如果一致，则正常发货；
* 如果出现不一致，则可能出现交易信息篡改，需要停止发货。

支付结果通知的参数如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **必选** | **参数类型** | **最大长度** | **参数说明** | **是否参与签名** |
| app\_key | Y | varchar | 32 | 应用app key | Y |
| product\_id | Y | varchar | 36 | 应用自定义的商品id | Y |
| amount | Y | int unsigned | 11 | 商品金额,以分为单位 | Y |
| app\_uid | Y | varchar | 50 | 应用分配给用户的id | Y |
| app\_ext1 | N | varchar | 255 | 应用扩展信息1原样返回 | Y |
| app\_ext2 | N | varchar | 255 | 应用扩展信息2原样返回 | Y |
| user\_id/m2\_i  d | Y | bigint  unsigned/varchar | 20/32 | 360帐号id/设备id,其中user\_id为bigint类型，长度20，m2\_id为varcahr类型，长度32  （登录状态的订单，此参数为360帐号id；未登录状态的订单，此参数为设备id） | Y |
| order\_id | Y | varchar | 32 | 360返回的支付订单号 | Y |
| gateway\_flag | Y | varchar | 16 | 如果支付返回成功，返回success  应用需要确认是success才给用户加钱 | Y |
| sign\_type | Y | varchar | 8 | 定值 md5 | Y |
| app\_order\_id | N | varchar | 64 | 应用订单号  支付请求时传递，原样返回 | Y |
| sign\_return | Y | varchar | 32 | 应用回传给订单核实接口的参数  不加入签名校验计算 | N |
| sign | Y | varchar | 32 | 签名 | N |

应用接收到支付平台回调的请求，参见附录的签名算法对参数进行签名，然后和平台传递的签名sign比较，从而校验平台请求的合法性.

通知消息样例（登录状态订单）:

order\_id=1211090012345678901&app\_key=1234567890abcdefghijklmnopqrstuv&product\_id=p1&amount=101&app\_uid=123456789&app\_ext1=XXX201211091985&app\_order\_id=order1234&user\_id=987654321&sign\_type=md5&gateway\_flag=success&sign=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&sign\_return=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

通知消息样例（未登录状态订单）:

order\_id=1211090012345678901&app\_key=1234567890abcdefghijklmnopqrstuv&product\_id=p1&amount=101&app\_uid=123456789&app\_ext1=XXX201211091985&app\_order\_id=order1234&m2\_id=4315da531b0c125035a9a13cb866e5f4&sign\_type=md5&gateway\_flag=success&sign=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&sign\_return=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

样例的签名字段排列( 登录状态， 列出来仅供参考, 请根据实际参数情况用程序排序产生, 不要写死在程序里)

amount, app\_ext1, app\_key, app\_order\_id, app\_uid, gateway\_flag, order\_id, product\_id, sign\_type, user\_id

样例的签名字段排列( 未登录状态，列出来仅供参考, 请根据实际参数情况用程序排序产生, 不要写死在程序里)

amount, app\_ext1, app\_key, app\_order\_id, app\_uid, gateway\_flag, m2\_id, order\_id, product\_id, sign\_type

样例的签名串

101#XXX201211091985#1234567890abcdefghijklmnopqrstuv#order1234#123456789#success#1211090012345678901#p1#md5#987654321#应用app\_secret

1. **应用服务端在接收到通知消息并校验后, 需将通知接收状态、商品/道具最终发货结果状态及未发货原因回应给360服务端：**

以utf8 编码 json格式应答;

应答格式及字段：{"status":"ok","delivery":"success","msg":""}

字段说明：（加粗字段为不可更改key值，且必须为小写）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **必选** | **参数类型** | **最大长度** | **参数说明** | **取值范围** |
| status | Y | string | 16 | 接收通知状态，status应答"ok"表示通知已经接收，360通知不再重发。如果该字段回应其他值或者不回应,则被认为通知失败,360会尝试多次通知.以避免掉单。 | ok,error |
| delivery | Y | string | 32 | 商品/道具最终发货结果状态，其中“mismatch”表示通知中360用户id或游戏用户id与游戏方下单时记录的值不同 | success,mismatch,other |
| msg | Y | string | 64 | 当发货状态为非success时说明原因，用于用户向360咨询时进行说明 | 自定义 |

注:应用应做好接收到多次通知的准备，防止多次加钱。对于重复的通知，应用可能发现订单已经成功处理完毕，无需继续处理，也应返回应答。否则，360会认为未成功通知，会继续发送通知。

### 3.3.3 订单核实接口【服务端调用】（选接）

1. 验证接口地址为:<https://mgame.360.cn/pay/order_verify.json>
2. 为了安全起见，验证参数不需要传client\_id,client\_secret参数，如果传了服务端会报错
3. 需要计算签名

为了防止伪造的支付成功通知, 应用可以使用本接口做通知数据的校验.把支付结果通知接口(3.3.2节)收到的通知消息里的参数, 计算签名后调用接口, 即可校验数据是否正确.

此接口仅支持登录状态订单，未登录状态订单无此功能。

**接口地址：https://mgame.360.cn/pay/order\_verify.json?参数**

**参数说明：**

|  |  |  |
| --- | --- | --- |
| **参数** | **必选** | **参数说明** |
| app\_key | Y | 应用app key |
| product\_id | Y | 应用自定义的商品id |
| amount | Y | 总价,单位：分 |
| app\_uid | Y | 应用分配给用户的id |
| order\_id | Y | 360支付订单号 |
| app\_order\_id | N | 应用订单号 下单时若指定验证时也要指定 |
| app\_ext1 | N | 应用扩展信息1 |
| app\_ext2 | N | 应用扩展信息2 |
| is\_sms | N | 是否短信支付 |
| bank\_code | N | 支付方式 |
| pay\_ext | N | 扩展信息 |
| sign\_type | Y | 当前仅支持md5 |
| sign\_return | Y | 应用传给订单核实接口的参数sign\_return |
| sign | Y | 签名(计算方法参考附录4.1节，本表格中除sign以外的所有参数均参与签名) |

参数均来自应用加钱接口收到的支付通知消息, 原样提供即可。

如果参数提供正确, 订单核实接口返回为json格式数据.

**验证成功返回**

{"ret":"verified"}

**验证不成功返回**

{"ret":"{错误信息}"}

返回结果中的错误信息包括但不限于

|  |  |
| --- | --- |
| **错误信息** | **错误说明** |
| order not exists | 订单不存在 |
| product\_id not match | 订单验证传入的product\_id和下单时传入的product\_id不一致 |
| amount not match | 验证金额与下单时金额不一致 |
| user\_id not match | 验证360用户id和下单时360用户id不一致 |
| bank\_code not match | 验证支付方式和下单时支付方式不一致 |

## 3.4 销毁接口【客户端调用】（必接）

在应用主Activity中，使用Matrix的destroy方法

**public static void** destroy(Contextcontext)

在应用主Activity的onDestroy()函数中调用，以退出登录状态并释放资源。调用完该接口后，360SDK又回到未初始化状态。

**参数：**

context上下文

**使用例子：**

@Override

protected void onDestroy() {

super.onDestroy();

Matrix.destroy(this);

}

## 3.5 退出接口【客户端调用】（必接）

**功能说明：**

用户点击back及其他方式退出游戏时，需调用此接口。

**接口示例及参数说明：**

/\*\*

\* 使用360SDK的退出接口

\*

\* @param isLandScape 是否横屏显示支付界面

\*/

**protected void** doSdkQuit(boolean isLandScape) {

Bundle bundle = new Bundle();

// 界面相关参数，360SDK界面是否以横屏显示。

bundle.putBoolean(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE, isLandScape);

// 可选参数，登录界面的背景图片路径，必须是本地图片路径

bundle.putString(ProtocolKeys.UI\_BACKGROUND\_PICTRUE, "");

// 必需参数，使用360SDK的退出模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_QUIT);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

Matrix.invokeActivity(this, intent, mQuitCallback);

}

**callback的 json数据格式：**

进入论坛

{"which": 1,"label": "进入论坛"}

退出游戏

{"which": 2,"label": "退出游戏"}

返回按键/右上角X图标

{"which": 0,"label": "返回键/X关闭"}

**callback示例：**

// 退出的回调

**private** IDispatcherCallback mQuitCallback = new IDispatcherCallback() {

@Override

public void onFinished(String data) {

// TODO your job

}

};

## 3.6 游戏角色信息上传 (选接)

**功能说明：**

采集游戏内相关数据，用于等级礼包等深度运营功能，**未接入或必传字段传值不满足要求将无法通过上线审核。**

如果不接入、接入错误、没有严格按照文档调用时机调用接口，**无法申请悬赏任务！请务必保证角色信息没有漏传现象！**

该功能支持客户端调用及服务端调用两种方式，游戏**任选其中一种接入即可**。

接入3.6.1游戏角色信息上传接口【客户端调用】或3.6.2游戏角色信息上传接口【服务端调用】后，请务必调用3.6.3角色信息上传自测接口自测是否上传成功，并是否上传

### 3.6.1 游戏角色信息上传接口【客户端调用】

**调用时机（以下四个时机均需调用，漏传不传将导致审核不通过！）：**

1、玩家进入游戏区服时调用该接口。

2、角色创建时调用该接口。

3、角色升级时调用该接口。

4、角色退出游戏时调用该接口。（包括用户退出游戏、用户注销角色帐号）

**上传参数说明：**

注：上传游戏包的时候，如果文档里面的必传参数，您们没传（例如角色性别、帮派等），需要说明一下“由于游戏确实没有xxx参数，所以角色信息接口中传了无或0” 。审核同学会亲自测试游戏，检测是否真的没有这些参数，请务必将游戏里有的必传参数上传。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 大类 | 子类 | 字段说明 | 对应字段 | 是否必填 | 类型 | 传参说明 |
| 上传场景 | 上传角色数据场景 | 上传数据时角色状态更新 | type | **必填** | **string** | 数据可根据场景定义随时上传，**以下场景必传：**  **enterServer（登录），levelUp（升级），createRole（创建角色），exitServer（退出）** |
| 角色 | 所在区服 | 当前登录的游戏区服ID | zoneid | **必填** | **Int** | **不能为空，必须为数字，**若无，传入“0” |
| 当前登录的游戏区服名称 | zonename | **必填** | **string** | **不能为空，**区服名称，要求与游戏界面展示的服务器名保持一致,长度不超过50，不能为null，若无，传入“无” |
| 角色信息 | 当前登录的玩家角色ID | roleid | **必填** | **BigInt** | 角色ID，一个角色同一个服ID保持唯一,长度不超过50 |
| 当前登录的玩家角色名 | rolename | **必填** | **string** | 不能为空，角色昵称,长度不超过50，不能为null，若无，传入“无” |
| 角色职业 | 当前登录玩家的职业ID | professionid | **必填** | **String** | **不能为空，必须为数字，**若无，传入“0” |
| 当前登录玩家的职业名称 | profession | **必填** | **string** | 不能为空，不能为null，若无，传入“无” |
| 角色性别 | 当前登录玩家的性别 | gender | **必填** | **string** | 不能为空，不能为null，可传入参数“男、女、无” |
| 角色职业称号 | 职业称号ID | professionroleid | 选填 | Int | 不能为空，不能为null，若无，传入 “0” |
| 职业称号 | professionrolename | 选填 | string | 不能为空，不能为null，若无，传入“无” |
| 角色等级 | 当前登录的玩家角色等级 | rolelevel | **必填** | **Int** | **不能为空，必须为数字**，且不能为null，若无，传入“0”, 如游戏存在转生，转职等，等级需累加，长度不超过10 |
| 战力值 | 数值 | power | **必填** | **Int** | **不能为空，必须为数字,**不能为null，若无，传入 “0” |
| VIP等级 | 当前用户VIP等级 | vip | **必填** | **Int** | **不能为空，必须为数字,**若无，传入“0” |
| 帐号余额 | 货币ID(balanceid) | balance | **必填** | **Int** | **分组传递和“好友关系”方式相同**。  **指充值现金货币，即用现金直接购买的货币**，不能为空，不能为null，若无，传入 “0”,货币ID的传参必须为数字  示例: [{“balanceid”:”1”,”balancename":"\u91d1\u5e01","balancenum":"600"},{"balanceid":"1","balancename":"\u91d1\u5e01","balancenum":"600"}] |
| 货币名称(balancename) | **必填** | **string** |
| 货币数额(balancenum) | **必填** | **Int** |
| 关系 | 所在帮派 | 当前用户所属帮派帮派ID | partyid | **必填** | **Int** | **不能为空，必须为数字,**不能为null，若无，传入 “0” |
| 当前用户所属帮派名称 | partyname | **必填** | **string** | 不能为空，不能为null，若无，传入“无” |
| 帮派职称 | 帮派称号ID | partyroleid | **必填** | **Int** | **帮派会长/帮主必传1，其他可自定义，不能为空**，不能为null，若无，传入 “0” |
| 帮派称号名称 | partyrolename | **必填** | **string** | 不能为空，不能为null，若无，传入“无” |
| 好友关系 | 关系角色ID(roleid) | friendlist | 选填 | BigInt | **传入用户的所有好友角色关系列表，不能为空，【角色ID亲密度 关系ID 关系名】为一组传递多个好友传递多组。其中 角色ID 亲密度 关系ID必须为数字不能为空，不能为null。**  **若无好友则传入“无”**  **示例:**  [{“roleid”:”1”,"intimacy":"0","nexusid":"600","nexusname":"情侣"},{"roleid":"2","intimacy":"0","nexusid":"200","nexusname":"仇人"}]  **关系预定义字段：**  以下对应方式为nexusid:nexusnam:  1:夫妻，2:结拜，3:情侣，4:师徒 ，5:仇人;其余关系从6开始自定义编号 |
| 亲密度(intimacy) | 选填 | Int |
| 关系ID(nexusid) | 选填 | Int |
| 关系名称(nexusname) | 选填 | string |
| 其他 | 排行榜列表 | 榜单ID(listid) | ranking | 选填 | Int | **分组传递和“好友关系”方式相同**。  不能为空，榜单ID必须为数字，不能为null，若无，传入 “无”  **示例:**  [{“listid”:”1”,"listname":"\u91d1\u5e01","num":"600","coin":"XX","cost":"XX"},{"listid":"1","listname":"\u91d1\u5e01","num":"600","coin":"XX","cost":"XX"}] |
| 榜单名称(listname) | 选填 | string |
| 传入角色当前排名(num) | 选填 | Int |
| 排名指标ID (coin) | 选填 | Int |
| 排名指标名称Value(cost) | 选填 | string |

**接口调用示例：**

*Matrix*调用下面函数

/\*\*

角色信息采集接口

\*/

**protected** **void** doSdkGetUserInfoByCP() {

//----------------------------模拟数据------------------------------

//帐号余额

JSONArray balancelist = new JSONArray();

JSONObject balance1 = new JSONObject();

JSONObject balance2 = new JSONObject();

//好友关系

JSONArray friendlist = new JSONArray();

JSONObject friend1 = new JSONObject();

JSONObject friend2 = new JSONObject();

//排行榜列表

JSONArray ranklist = new JSONArray();

JSONObject rank1 = new JSONObject();

JSONObject rank2 = new JSONObject();

try {

balance1.put("balanceid","1");

balance1.put("balancename","bname1");

balance1.put("balancenum","200");

balance2.put("balanceid","2");

balance2.put("balancename","bname2");

balance2.put("balancenum","300");

balancelist.put(balance1).put(balance2);

friend1.put("roleid","1");

friend1.put("intimacy","0");

friend1.put("nexusid","300");

friend1.put("nexusname","情侣");

friend2.put("roleid","2");

friend2.put("intimacy","0");

friend2.put("nexusid","600");

friend2.put("nexusname","情侣");

friendlist.put(friend1).put(friend2);

rank1.put("listid","1");

rank1.put("listname","listname1");

rank1.put("num","num1");

rank1.put("coin","coin1");

rank1.put("cost","cost1");

rank2.put("listid","2");

rank2.put("listname","listname2");

rank2.put("num","num2");

rank2.put("coin","coin2");

rank2.put("cost","cost2");

ranklist.put(rank1).put(rank2);

} catch (JSONException e) {

e.printStackTrace();

}

HashMap<String, String> eventParams=new HashMap<String, String>();

eventParams.put("type","enterServer"); //（必填）角色状态（enterServer（登录），levelUp（升级），createRole（创建角色），exitServer（退出））

eventParams.put("zoneid","2"); //（必填）游戏区服ID

eventParams.put("zonename","测试服"); //（必填）游戏区服名称

eventParams.put("roleid","123456"); //（必填）玩家角色ID

eventParams.put("rolename","冷雨夜风"); //（必填）玩家角色名

eventParams.put("professionid","1"); //（必填）职业ID

eventParams.put("profession","战士"); //（必填）职业名称

eventParams.put("gender","男"); //（必填）性别

eventParams.put("professionroleid","0"); //（选填）职业称号ID

eventParams.put("professionrolename","无"); //（选填）职业称号

eventParams.put("rolelevel","30"); //（必填）玩家角色等级

eventParams.put("power","120000"); //（必填）战力数值

eventParams.put("vip","5"); //（必填）当前用户VIP等级

eventParams.put("balance",balancelist.toString()); //（必填）帐号余额

eventParams.put("partyid","100"); //（必填）所属帮派帮派ID

eventParams.put("partyname","王者依旧"); //（必填）所属帮派名称

eventParams.put("partyroleid","1"); //（必填）帮派称号ID

eventParams.put("partyrolename","会长"); //（必填）帮派称号名称

eventParams.put("friendlist",friendlist.toString()); //（必填）好友关系

eventParams.put("ranking",ranklist.toString()); //（选填）排行榜列表

//参数eventParams相关的 key、value键值对 相关具体使用说明，请参考文档。

//----------------------------模拟数据------------------------------

Matrix.statEventInfo(getApplicationContext(), eventParams);

}

### 3.6.2 游戏角色信息上传接口【服务端调用】

详情参见压缩包内的：角色信息上传接口【服务端调用】.zip

压缩包内仅为服务端php方法以及java方法demo，其他方式接入请自行开发

### 3.6.3 角色信息上传自测接口【有浏览器就能测】（必测！）

该页面供开发者测试使用，**用于验证角色信息上传接口**。

请务必接入并调用角色信息接口，并使用自测接口测试！**游戏提审时，必须提供3个有角色数据的360账号，分别为刚创角、刚通过新手任务、超过20级3个账号。**请务必确认提审的3个账号中有数据，且数据与游戏内真实数据完全一致，再上传游戏包。否则，审核人员将拒审游戏包，或审核不通过游戏包，将严重影响游戏上线发布！

* 能从接口查询到测试用户数据且各字段值符合3.6.1或3.6.2游戏角色信息上传接口要求，则接入成功；
* 能从接口查询到测试用户数据，则说明数据上传成功，请校验各角色信息字段值是否符合3.6.1或3.6.2游戏角色信息上传接口要求，若不符合请修改；
* 若不能从接口查询到测试用户数据，则说明数据上传失败，接入有误，请根据3.6.1或3.6.2游戏角色信息上传接口要求进行修改。

**自测页面**：<http://guact.u.360.cn/roletool/so>

## 3.7 游戏activity生命周期接口【客户端调用】(必接)

*Matrix*调用下面生命周期函数

**接口示例：**

**//**游戏Activity必接 生命周期接口

**public static void** onStart(Activity activity)

**//**游戏Activity必接生命周期接口

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

**//**游戏Activity必接生命周期接口

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

**//**游戏Activity必接生命周期接口

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

**//**游戏Activity必接生命周期接口

**public static void** onReStart(Activity activity)

**//**游戏Activity必接生命周期接口

**public static void** onActivityResult (Activity activity,int requestCode, int resultCode, Intent data)

**//**游戏Activity必接生命周期接口

**public static void** onNewIntent (Activity activity,Intent intent)

## 3.8 其他接口

### 3.8.1 实名状态查询接口【客户端\服务端调用】（选接）

**功能说明：**

360sdk已包含实名注册功能及未成年防沉迷功能，在登录、支付流程中针对未实名用户展示实名认证窗口，并在实名成功后对未成年用户限制游戏时长和充值金额。同时，游戏可根据自身实际情况决定对游戏内不同实名状态用户的防沉迷逻辑进一步处理。

**使用方法**：请在初始化接口成功后再调用该查询接口，具体调用方法参见【3.1 初始化接口】

**服务端接口示例**：

https://mgame.360.cn/user/check\_authentication.json?qids=xxxx&access\_token=xxxx

**客户端接口示例及参数说明：**

/\*\*

\* 本方法中的callback实现仅用于测试, 实际使用由游戏开发者自己处理

\*

\* @param accessToken

\* @param qihooUserId 奇虎360用户ID

\*/

**protected void** doSdkAntiAddictionQuery(String accessToken, String qihooUserId) {

Bundle bundle = new Bundle();

// 必需参数，用户access token，要使用注意过期和刷新问题，最大64字符。

bundle.putString(ProtocolKeys.ACCESS\_TOKEN, accessToken);

// 必需参数，360帐号id。

bundle.putString(ProtocolKeys.QIHOO\_USER\_ID, qihooUserId);

// 必需参数，使用360SDK的防沉迷查询模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_ANTI\_ADDICTION\_QUERY);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

Matrix.execute(this, intent, new IDispatcherCallback() {

@Override

public void onFinished(String data) {

}

}

}

**callback的 json数据格式：**

结果返回

{"ret":[{"qid":"199062142","status":"2"}]}

|  |  |
| --- | --- |
| **字段** | **说明** |
| **error\_code** | 0 查询成功 其他值查询失败 |
| **error\_msg** | 错误消息 |
| **content** | json对象，包含ret数组 |
| **ret** | json对象的数组 |
| **qid** | 奇虎UserId |
| **status** | 0，此用户未进行实名注册；1，未成年；2，已成年。 |

**callback示例：**

new IDispatcherCallback() {

@Override

**public void** onFinished(String data) {

if (!TextUtils.isEmpty(data)) {

try {

JSONObject resultJson = new JSONObject(data);

int errorCode = resultJson.optInt("error\_code");

if (errorCode == 0) {

JSONObject contentData = resultJson.getJSONObject("content");

if(contentData != null) {

// 保存登录成功的用户名及密码

JSONArray retData = contentData.getJSONArray("ret");

if(retData != null && retData.length() > 0) {

int status = retData.getJSONObject(0).optInt("status");

switch (status) {

case 0: // 查询结果: 此用户未进行实名注册

Toast.makeText(SdkUserBaseActivity.this, getString(R.string.anti\_addiction\_query\_result\_0), Toast.LENGTH\_LONG).show();

break;

case 1: // 查询结果:未成年

Toast.makeText(SdkUserBaseActivity.this, getString(R.string.anti\_addiction\_query\_result\_1), Toast.LENGTH\_LONG).show();

break;

case 2: // 查询结果:已成年

Toast.makeText(SdkUserBaseActivity.this, getString(R.string.anti\_addiction\_query\_result\_2), Toast.LENGTH\_LONG).show();

break;

default:

break;

}

return;

}

}

} else {

Toast.makeText(SdkUserBaseActivity.this, resultJson.optString("error\_msg"), Toast.LENGTH\_SHORT).show();

return;

}

} catch (JSONException e) {

e.printStackTrace();

}

Toast.makeText(SdkUserBaseActivity.this, getString(R.string.anti\_addiction\_query\_exception), Toast.LENGTH\_LONG).show();

}

}

}

### 3.8.2 打开实名认证界面接口【客户端调用】（选接）

**功能说明：**

360sdk已包含实名状态注册功能，在登录、支付流程中针对未实名用户展示实名认证窗口，若游戏方有个性化定制实名认证窗口展示时机的需求，可自行在游戏中调用此接口。

注意：一定要先行调用实名状态查询接口，用户未实名才能调用此接口！已实名用户请使用其他逻辑处理！

**接口示例及参数说明**：

/\*\*

\* 使用360SDK实名注册接口

\* 实名注册

\* @param isLandScape 是否横屏显示登录界面

\* @param qihooUserId 奇虎360用户ID

\*/

//

protected void doOpenSdkRealName(QihooUserInfo usrinfo, boolean isLandScape) {

if (!checkLoginInfo(usrinfo)) {

return;

}

Intent intent = getRealNameRegisterIntent(isLandScape, (usrinfo != null) ? usrinfo.getId() : null);

Matrix.invokeActivity(this, intent, new IDispatcherCallback() {

@Override

public void onFinished(String data) {

**//具体见返回值说明**

}

});

}

private Intent getRealNameRegisterIntent(boolean isLandScape, String qihooUserId) {

Bundle bundle = new Bundle();

// 界面相关参数，360SDK界面是否以横屏显示。

bundle.putBoolean(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE, isLandScape);

// 必需参数，360账号id，整数。

bundle.putString(ProtocolKeys.QIHOO\_USER\_ID, qihooUserId);

// 可选参数，登录界面的背景图片路径，必须是本地图片路径

bundle.putString(ProtocolKeys.UI\_BACKGROUND\_PICTRUE, "");

// 必需参数，使用360SDK的实名注册模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_REAL\_NAME\_REGISTER);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

return intent;

}

**实名制提交接口返回结果说明（从720版本开始数据格式为以下）：**

**从720版本开始数据格式为以下格式，如果error\_code不为0，或者content没有实名数据，则可以认为实名制页面关闭或者异常。该结果仅供参考，如果游戏接入方需要准确结果建议提交实名制后再次调用3.8.1查询接口**

{

"error\_code": "0",

"error\_msg": "请求成功",

"content": {

"ret": [{

"qid": "xxxxxx",

"status": "2" //0是未知，1是未成年，2是成年

}]

}

}

### 3.8.3 分享接口【客户端调用】（选接）

**功能说明：**

分享接口，支持分享链接或图片形式内容，支持5种分享方式：微信好友、微信朋友圈、QQ好友、QQ空间、复制链接。

调用此接口可在游戏内调起分享聚合界面，或直接调起指定一种分享方式。

**接口示例及参数说明：**

/\*\*

\* 提供给游戏CP调用分享接口进行URL链接或图片分享

\*/

Intent intent = new Intent();

/\*

\* 必须参数：

\* shareType ：分享类型

ProtocolKeys.SHARE\_DEFAULT\_STRING//分享方式聚合页

ProtocolKeys.SHARE\_2\_WX\_FRIENDS//微信好友

ProtocolKeys.SHARE\_2\_WX\_TIMELINE//微信朋友圈

ProtocolKeys.SHARE\_2\_QQ\_FRIENDS//QQ好友

ProtocolKeys.SHARE\_2\_QQ\_SPACE//QQ空间

ProtocolKeys.SHARE\_2\_CLIP\_BOARD//复制链接

\* function\_code ：标识通知SDK要执行的功能，必须参数

\* share\_method ：分享方式，必须参数

ProtocolConfigs.SHARE\_TYPE\_URL: 链接分享

ProtocolConfigs.SHARE\_TYPE\_IMG：图片分享

\* share\_title ：分享的标题，若分享方式为链接分享，此参数为必选参数

\* share\_desc ：分享的描述，若分享方式为链接分享，此参数为必选参数

\* share\_icon ：分享的icon链接，若分享方式为链接分享，此参数为必选参数

\* share\_url ：分享的URL链接，若分享方式为链接分享，此参数为必选参数

\* share\_pic ：分享的图片链接，若分享方式为图片分享，此参数为必选参数

\*/

intent.putExtra(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_SHARE);

intent.putExtra(ProtocolKeys.SHARE\_TYPE, shareType);

intent.putExtra(ProtocolKeys.SHARE\_METHOD, shareMethod);

intent.putExtra(ProtocolKeys.SHARE\_TITLE, title);

intent.putExtra(ProtocolKeys.SHARE\_DESC, desc);

intent.putExtra(ProtocolKeys.SHARE\_ICON, icon);

intent.putExtra(ProtocolKeys.SHARE\_URL, url);

intent.putExtra(ProtocolKeys.SHARE\_PIC, pic);

Matrix.invokeActivity(SdkUserBaseActivity.this, intent, new IDispatcherCallback() {

@Override

public void onFinished(String data) {

if (null == data) {

return;

}

Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH\_SHORT).show();

}

});

**callback的 json数据格式：**

分享成功

{"errno":1,"share\_way":"weixin\_friends","errmsg":"shared","shared":true}

分享失败

{"errno":-1,"share\_way":"weixin\_friends","errmsg":"failed","shared":false}

errno: 1 分享成功;-1 分享失败;-2 取消分享; 0 未收到第三方分享结果，建议视为成功处理

### 3.8.4 打开论坛接口【客户端调用】（选接）

**功能说明：**

打开论坛接口，调用此接口可直接在游戏内调起webview页面,展示游戏专属论坛(若游戏暂未开通论坛,则默认展示论坛首页),为游戏用户提供分享、吐槽的地方，提高游戏用户的粘性。

**接口示例及参数说明：**

/\*\*

\* 使用360SDK的论坛接口

\*

\* @param isLandScape 是否横屏显示支付界面

\*/

**protected void** doSdkBBS(boolean isLandScape) {

Bundle bundle = new Bundle();

// 界面相关参数，360SDK界面是否以横屏显示。

bundle.putBoolean(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE, isLandScape);

// 必需参数，使用360SDK的论坛模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_BBS);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

Matrix.invokeActivity(this, intent, null);

}

### 3.8.5 打开客服接口【客户端调用】（选接）

**功能说明：**

打开客服接口，调用此接口可直接在游戏内调起webview页面,展示游戏客服界面,为游戏用户提供快捷的问题反馈通道。

**接口示例及参数说明：**

/\*\*\*

\* 调用360SDK客服接口

\*

\* @param isLandScape是否横屏显示支付界面

\* @return Intent

\*/

**private void** openKefu(boolean isLandScape) {

Bundle bundle = new Bundle();

// 界面相关参数，360SDK界面是否以横屏显示。

bundle.putBoolean(ProtocolKeys.IS\_SCREEN\_ORIENTATION\_LANDSCAPE, isLandScape);

// 必需参数，使用360SDK的客服模块。

bundle.putInt(ProtocolKeys.FUNCTION\_CODE, ProtocolConfigs.FUNC\_CODE\_KEFU);

Intent intent = new Intent(this, ContainerActivity.class);

intent.putExtras(bundle);

Matrix.invokeActivity(this, intent, null);;

}

### 3.8.6 使用Matrix的get方法，获取基本信息【客户端调用】（选接）

参数：

context上下文

// 获取AndroidManifest.xml中的meta-data QHOPENSDK\_APPID

**public static** String getAppId(Context context)

// 获取AndroidManifest.xml中的meta-data QHOPENSDK\_APPKEY

**public static** String getAppKey(Context context)

// 获取AndroidManifest.xml中的meta-data QHOPENSDK\_PRIVATEKEY

**public static** String getPrivateKey(Context context)

// 获取sdk的VersionName

**public static** String getVersionName(Context context)

### 3.8.7 游戏退出时设置杀死应用进程标志位【客户端调用】（选接）

**功能说明：**

游戏CP设置退出游戏时是否会杀死应用进程标志

**接口示例及参数说明：**

参数值：如果传入false：表示SDK内部将在退出游戏时杀死应用进程，则游戏无需手动杀死进程。

如果传入true：表示游戏退出时自己会杀死应用进程，则SDK内部不做任何处理。

默认值是false。

/\*\*

\* 提供给游戏CP设置退出游戏时是否会退出应用进程标志

\* **@param** isKillApp

\* **@return**

\*/

**public** **static** void setKillAppTag(boolean isKillApp)

### 3.8.8 初始化前修改浮窗默认位置【客户端调用】（选接）

**功能说明：**

游戏CP在调用初始化接口前可设置悬浮球默认悬停的位置，一共有四个位置：左上（值为：1）、左下（值为：2）、右上（值为：3）、右下（值为：4），如果此时用户已有操作浮球位置记录，则以用户记录为准。

**接口示例及参数说明：**

参数值：左上（值为：1）、左下（值为：2）、右上（值为：3）、右下（值为：4）。默认值是0和1效果一致，显示在左上角位置。

/\*\*

\* 设置悬浮球默认悬停的位置

\* @param position ：左上（值为：1）、左下（值为：2）、右上（值为：3）、右下（值为：4）

\* 可以使用下面的常量：

\* public static final int FLOAT\_WND\_POSITION\_LEFT\_TOP = 1;

\* public static final int FLOAT\_WND\_POSITION\_LEFT\_BOTTOM = 2;

\* public static final int FLOAT\_WND\_POSITION\_RIGHT\_TOP = 3;

\* public static final int FLOAT\_WND\_POSITION\_RIGHT\_BOTTOM = 4;

\*\*/

Matrix.setFloatWindowPosition(int position);

### 3.8.9 打开浮球里的浮窗【客户端调用】（选接）

**功能说明：**

游戏CP可调用该方法打开浮球里的浮窗内容，注意该方法不要跟登录同时调用。。

**接口示例及参数说明：**

参数值：传入Activity，与初始化方法中的activity保持一致。

/\*\*

\* 打开插件中悬浮球内容

\* @param 跟setActivity中的Activity保持一致

\* 适用于悬浮球被隐藏时接入方主动调起浮球内容，浮球未被隐藏时也可以使用

\*\*/

Matrix.openFloatWindow(Activity activity)

# 4. 附录

## 4.1 签名算法

签名算法不区分前后端，只要在需要签名的地方，均采用如下的算法

1. 必选参数必须有值, 而且参数值必须不为空，不为0. 字符集为utf-8
2. 所有不为空，不为0的参数都需要加入签名，参数必须为做urlencode之前的原始数值. 如中文金币, 作为参数传输时编码为%E9%87%91%E5%B8%81, 做签名时则要用其原始中文值金币 (注意字符集必须是UTF-8)
3. 对所有不为空的参数按照参数名字母升序排列(如php的ksort函数)
4. 使用符号#拼装排序后的参数值, 最后用#连接应用的app\_secret,整体用md5计算签名, 就是sign参数的值. 注意有些语言的md5计算结果里字母为大写, 需要转化为小写.
5. 拼装url进行WEB传递, 这时参数值要做urlencode

php范例如下：

// 准备签名参数

$input = array(…);

/\* 去掉为空的字段 \*/

foreach($input as $k=>$v)

{

if(empty($v)){

unset($input[$k]);

}

}

ksort($input);//对参数按照key进行排序

$sign\_str = implode('#',$input);//第四步

$sign\_str = $sign\_str.'#'.$sign\_key;//拼装密钥（如果是签名，密钥为约定处理后的密钥）

$sign = md5($sign\_str);

$input['sign'] = $sign;//得到签名

/\* 第五步得到url传递即可\*/

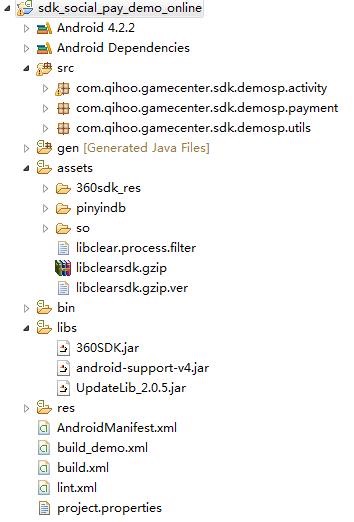
//这里地址就是一个演示的接口地址

$url = 'http://testapp.com/notify?'.http\_build\_query($input);

…

## 4.2 Demo工程简介

360SDK工程结构如图所示：



图Demo工程

Demo工程主要的包和类介绍：

com.qihoo.gamecenter.sdk.demosp.activity下的类演示SDK各个接口的调用。游戏接入SDK联调时，请着重参考类：SdkUserBaseActivity：

LandscapeSdkUserActivity.java横屏状态下演示SDK各个接口的调用

PortraitSdkUserActivity.java竖屏状态下掩饰SDK各个接口的调用

SdkUserBaseActivity.java 是LandscapeSdkUserActivity和PortraitSdkUserActivity的基类，是SDK各个接口的调用实现.

com.qihoo.gamecenter.sdk.demosp.payment下是支付相关的基本数据结构和常量。

QihooPayInfo.java , 请求360SDK支付接口时的参数信息类。

com.qihoo.gamecenter.sdk.demosp.utils，demo里面定义的一些工具类。

QihooUserInfo，360用户信息数据。

Utils.java, 一些工具函数。

## 4.3 服务端SDK参考程序

应用在接入360平台时，需要在应用服务端搭建用户授权接口和支付回调接口。具体参考程序可以从<http://dev.360.cn/wiki/index/id/73>下载。

## 4.4 常见问题

接入相关问题可以先参01文件夹下的『常见问题』文档，仍不能解决时，请联系下方人员。

## 4.5 联系我们

**接入中遇到问题请先查看“常见问题”文档并用自检工具自检，自检通过后依然查不出问题请联系对接运营协调处理。**

平台加固相关问题请加Q群

**360加固保客服**：198501978