360 账号_社交_支付_数据_推送 SDK 接入文档

版本号	修改时间	内容
1.1.6(300)	2015-04-21	增加 360 币与代金券及其它支付方式的组合支付 全新 UI 及交互方式改版，适配手机用手操作习惯 优化登录过程中出现的问题 金币商城改版，更换新的 web 页地址
1.1.8（400）	2015-08-04	支持奇酷帐号登录 增加微信支付 代金券发码兑换功能 内部逻辑功能优化
1.2.0（410）	2015-09-24	新增语音功能 登陆优化，增加快速注册 微信支付支持 360 币和代金券组合支付 增加微信、360 币组合支付 增加消费完成抽奖功能 支付失败引导
1.2.2(420)	2015-11-13	登录激活码功能
1.2.4(430)	2016-01-19	浮窗增加皮肤设置 360 钱包改版并增加代金券商城 增加抢代金券红包功能 增加 cp 读取用户级别和金币的接口 内部逻辑优化
1.3.0(460)	2016-03-24	增加直播功能
1.3.0(462)	2016-03-29	增加直播送礼物功能
1.3.0(464)	2016-04-06	增加未登录状态直播模块入口
1.3.0(466)	2016-04-11	统计优化
1.3.0(468)	2016-04-13	新增直播礼物云控功能 其他细节优化

360 移动开放平台　2016 年 4 月 13 日

目录

1.概述

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 社交带支付 SDK，本 SDK 为安卓应用提供登录、注册、社交、支付等功能。

重要提醒

游戏若不需要支付功能，可接入 SDK，但不调用支付模块。
要使用联网支付则必须使用自己搭建的服务器，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。

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 App-Secret 保存在手机客户端，AndroidManifest 中存放的是 Private Key，不是 App-Secret。
Private Key 的算法为：QHOPENSDK_PRIVATEKEY=MD5(appSecret + "#" + appKey)

2.接入流程

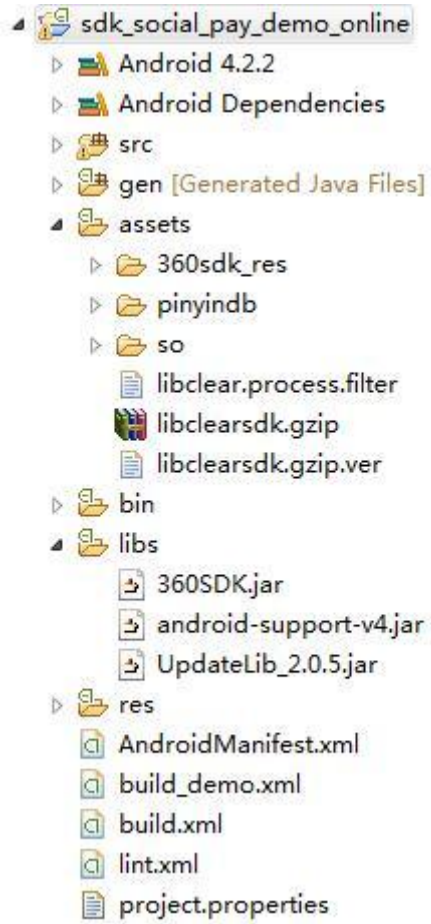
2.1 申请 APPID、APPKEY 和 APPSECRET

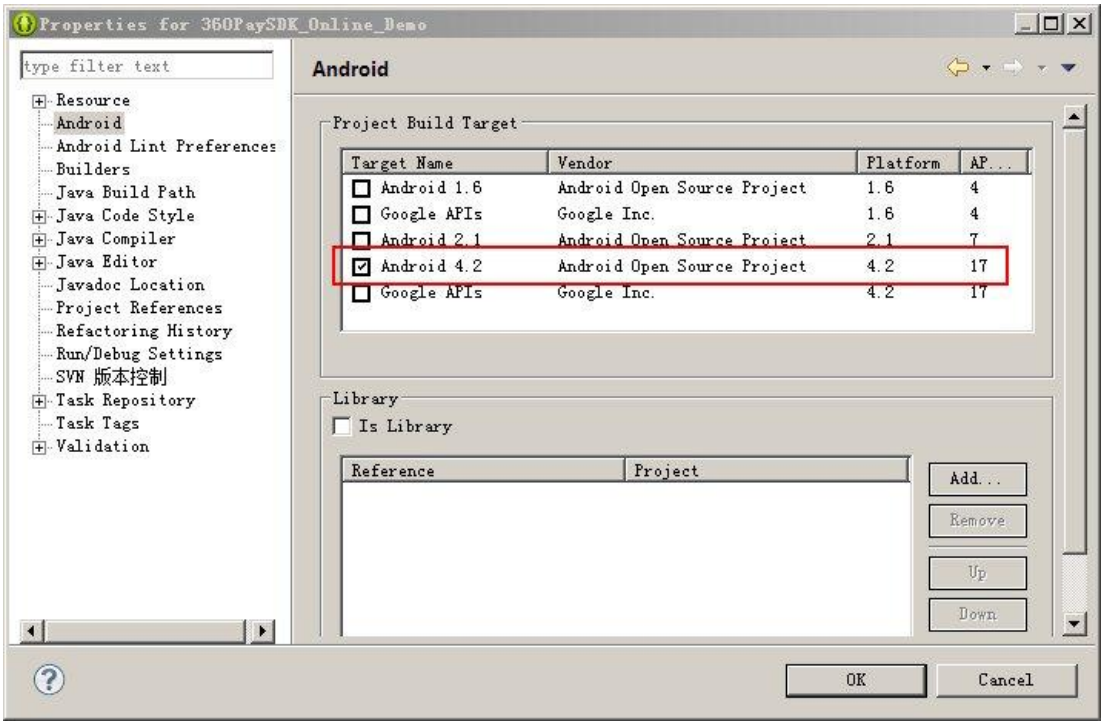
企业开发者需要在 360 开放平台 http://dev.360.cn/ 申请 APPID、APPKEY 及 APPSECRET（一个应用只能申请一个 appkey）。

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）：


General
Archive File
Existing Projects into Workspace
File System
Preferences
Android
Existing Android Code Into Workspace
C/C++
Install
Run/Debug
SVN
Tasks
Team
XML


sdk_social_pay_demo_online
Android 4.2.2
Android Dependencies
src
gen [Generated Java Files]
assets
360sdk_res
pinyindb
so
libclear.process.filter
libclearsdk.gzip
libclearsdk.gzip.ver
bin
libs
360SDK.jar
android-support-v4.jar
UpdateLib_2.0.5.jar
res
AndroidManifest.xml
build_demo.xml
build.xml
lint.xml
project.properties



2.2.1 导入 jar 包

将 SDK 包内的 libs 目录下的文件（夹）放到应用工程的 libs 目录下。

2.2.2 添加 assets 目录下的文件

将 assets 目录下的文件（夹）复制到应用工程 assets 目录下。

2.3 配置应用工程的 AndroidManifest.xml

具体配置参见 SDK 包内的 demo-src 目录下的 demo 源代码。

2.3.1 添加权限

```
<!--添加 360SDK 必需要的权限。 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<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.SYSTEM_ALERT_WINDOW" />
```

```
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <!-- payment -->
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.RESTART_PACKAGES" />
    <uses-permission android:name="android.webkit.permission.PLUGIN" />
    <!-- float sdk -->
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <!-- weixin -->
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <!-- qiku start -->
    <!-- 系统账户操作权限 -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
    <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" />
    <!-- 系统设置操作权限 -->
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.READ_SETTINGS" />
    <!-- 语音聊天模块权限（如果定制语音聊天模块，则添加该权限） -->
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <!-- QDAS 打点 SDK 所需权限 -->
    <uses-permission android:name="android.permission.READ_LOGS" />
```

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 -->
    <activity
    android:name="com.qihoo.gamecenter.sdk.activity.ContainerActivity"
    android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|u
iMode|layoutDirection"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:exported="true"
    >
    </activity>
```

```
    <!-- 360SDK 浮窗 浮窗相关配置在 1.1.0 以及以后的版本中删除了，如果以前接入过 SDK 的游戏配置中存在浮
窗相关的配置，请手动删除 -->
    <!-- 个人中心 activity -->
    <!-- activity
    android:name="com.qihoo.gamecenter.sdk.suspend.personal.PersonalActivity"
    android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|u
iMode"
    android:windowSoftInputMode="adjustPan"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:screenOrientation="portrait"  >
    <intent-filter>
      <action android:name="" />
      </intent-filter>
    </activity -->
    <!-- receiver
    android:name="com.qihoo.gamecenter.sdk.suspend.local.QBootReceiver"
    android:permission="android.permission.RECEIVE_BOOT_COMPLETED" >
    <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
    </receiver -->

    <!--service
    android:name="com.qihoo.gamecenter.sdk.suspend.remote.QRemoteService"
    android:exported="true"
    android:process=":QSuspendRemote" >
    <intent-filter>
    <action android:name="com.qihoo.gamecenter.sdk.suspend.service.action.remote" />
    </intent-filter>
    </service -->
    <!-- service
    android:name="com.qihoo.gamecenter.sdk.suspend.local.QLocalService"
    android:exported="false" >
    <intent-filter>
    <action android:name="com.qihoo.gamecenter.sdk.suspend.local.QLocalService" />
    </intent-filter>
    </service -->
    <!-- 360SDK 浮窗 End -->


    <!-- 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|u
iMode"
        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-->
    <activity
        android:name="com.iapppay.pay.channel.weixinpay.WeixinWapPayActivity"
        android:configChanges="screenSize|orientation|keyboard|navigation|layoutDirection"
        android:theme="@android:style/Theme.Translucent" />
    <activity
        android:name="com.junnet.heepay.ui.activity.WelcomeActivity"
        android:configChanges="keyboardHidden|orientation|screenSize"
        android:excludeFromRecents="true"
        android:screenOrientation="behind"
        android:theme="@android:style/Theme.Dialog"
        android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
     <activity
        android:name="com.junnet.heepay.ui.activity.WechatPaymentActivity"
        android:configChanges="keyboardHidden|orientation|screenSize"
        android:excludeFromRecents="true"
        android:screenOrientation="behind"
        android:theme="@android:style/Theme.Dialog"
        android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
     <activity
        android:name="com.ipaynow.plugin.activity.PayMethodActivity"
        android:configChanges="keyboardHidden|orientation|screenSize"
        android:exported="false"
        android:screenOrientation="behind"
        android:theme="@android:style/Theme.Dialog" />
    <activity
        android:name="com.ipaynow.plugin.inner_plugin.wechat_plugin.activity.WeChatNotifyActiv
ity"
        android:configChanges="keyboardHidden|orientation|screenSize"
        android:screenOrientation="behind"
        android:theme="@android:style/Theme.NoDisplay" />
    <!-- payment activities end -->
    <!--如下是 360 游戏实时推送 SDK 必要声明，不可修改 -->
    <receiver
    android:name="com.qihoo.psdk.local.QBootReceiver"
    android:permission="android.permission.RECEIVE_BOOT_COMPLETED" >
    <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED" />
```

```
</intent-filter>
<intent-filter>
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
</intent-filter>
</receiver>

<activity
android:name="com.qihoo.psdk.app.QStatActivity"
android:launchMode="singleInstance"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" >
</activity>

<service
android:name="com.qihoo.psdk.remote.QRemoteService"
android:exported="true"
android:process=":QRemote" >
<intent-filter>
<action android:name="com.qihoo.psdk.service.action.remote" />
</intent-filter>
</service>
<service
 android:name="com.qihoo.psdk.local.QLocalService"
 android:exported="true"
 android:process=":QLocal" >
 <intent-filter>
     <action android:name="com.qihoo.psdk.service.action.local" />
 </intent-filter>
 </service>
<!-- push sdk end -->
<!--微信相关的 activity，如需接入，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：除非游戏打算以后永远不会支持微信分享，否则建议游戏把这个 actvity 的配置加上。
此版本的 SDK 支持从服务端配置微信分享的 appid。
Manifest 中的 QHOPENSDK_WEIXIN_APPID 字段可以先不配，后续申请到微信的 appid 可以在服务端配置。
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" />
<!-- qiku 组件 start -->
```

```
    <activity
        android:name="com.coolcloud.uac.android.plug.view.LoginActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.AssistActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.AuthenticateActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.FindpwdActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.LoginActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.OAuth2Activity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <activity
        android:name="com.coolcloud.uac.android.api.view.RegisterActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" >
    </activity>
    <!-- qiku 组件 end -->
    <!-- UpdateLib start -->
    <service
        android:name="com.qihoo.appstore.updatelib.CheckUpdateService"
        android:exported="false" />
    <activity
        android:name="com.qihoo.appstore.updatelib.CheckUpdateAcitivty"
        android:exported="false"
        android:theme="@android:style/Theme.Translucent" />
    <!-- UpdateLib end -->
    <!-- gameunion plugin start -->
    <activity
         android:name="com.qihoo.gameunionforsdk.SimpleWebView"
         android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSi
ze|uiMode|layoutDirection"
          android:theme="@android:style/Theme.Translucent.NoTitleBar" />
    <!-- gameunion plugin 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！-->
<meta-data
android:name="QHOPENSDK_PRIVATEKEY"
android:value="4e04fe9ac8e2a73cbb27ba52ac076eb9">
</meta-data>
<!-- 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="wx4e203f3fdd2d4a15" >
</meta-data>
注意：此处的微信 appid 申请只与微信分享有关，微信支付功能无需 appid 即可使用

<!-- 默认参数，不需要修改，直接复制就行 -->
<meta-data android:name="DC_APPKEY" android:value="02522a2b2726fb0a03bb19f2d8d9524d"/>
```

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 com.alipay.** {*;}
-keep class com.qihoo.** {*;}
-keep class com.qihoo360.** { *; }
-keep class com.qihoopp.** { *; }
-keep class com.yeepay.safekeyboard.** { *; }
-keep class com.amap.** {*;}
-keep class com.aps.** {*;}
-keep class com.iapppay.** {*;}
-keep class com.ipaynow.** {*;}
-keep class com.junnet.heepay.** {*;}
-keep class com.tencent.mm.** {*;}
-keep class com.coolcloud.uac.android.** {*;}
-keep class tv.cjump.jni.** {*;}
-keep class HttpUtils.** {*;}
-keep class com.a.a.** {*;}
-keep class com.emoji.** {*;}
-keep class com.google.android.exoplayer.** {*;}
-keep class com.ta.utdid2.** {*;}
-keep class com.ut.device.** {*;}
-keep class com.master.flame.danmaku.** {*;}
```

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.alipay.android.app.**
-dontwarn com.qihoo.**
-dontwarn com.qihoo360.**
-dontwarn com.qihoopp.**
-dontwarn com.yeepay.safekeyboard.**
-dontwarn com.amap.**
-dontwarn org.apache.http.conn.ssl.SSLSocketFactory
```

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)

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

```
public static void init(final Activity activity)
```

用于初始化 360SDK，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

使用例子:

```
Matrix.init(this);
```

3.2 登录授权流程

3.2.1 流程介绍

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。用户若用同样账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。

3.2.2 接口介绍

3.2.2.1 登录接口【客户端调用】(必接)

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：获取用户信息后，应用需要保存自身账号与 360 账号的绑定关系。

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);
        //可选参数，是否显示关闭按钮，默认不显示
        intent.putExtra(ProtocolKeys.IS_LOGIN_SHOW_CLOSE_ICON,
                        getCheckBoxBoolean(R.id.isShowClose));
        // 可选参数，是否支持离线模式，默认值为 false
        intent.putExtra(ProtocolKeys.IS_SUPPORT_OFFLINE,
                        getCheckBoxBoolean(R.id.isSupportOffline));
        // 可选参数，是否在自动登录的过程中显示切换账号按钮，默认为 false
        intent.putExtra(ProtocolKeys.IS_SHOW_AUTOLOGIN_SWITCH,
                        getCheckBoxBoolean(R.id.isShowSwitchButton));
        // 可选参数，是否隐藏欢迎界面
        intent.putExtra(ProtocolKeys.IS_HIDE_WELLCOME,
                        getCheckBoxBoolean(R.id.isHideWellcome));
        /*
         * 指定界面背景（可选参数）：
         *  1.ProtocolKeys.UI_BACKGROUND_PICTRUE 使用的系统路径，如/sdcard/1.png
         *  2.ProtocolKeys.UI_BACKGROUND_PICTURE_IN_ASSERTS 使用的 assest 中的图片资源，
         *    如传入 bg.png 字符串，就会在 assets 目录下加载这个指定的文件
         *  3.图片大小不要超过 5M，尺寸不要超过 1280x720，后缀只能是 jpg、jpeg 或 png
         */
        // 可选参数，登录界面的背景图片路径，必须是本地图片路径
        intent.putExtra(ProtocolKeys.UI_BACKGROUND_PICTRUE, getUiBackgroundPicPath());
        // 可选参数，指定 assets 中的图片路径，作为背景图
        intent.putExtra(ProtocolKeys.UI_BACKGROUND_PICTURE_IN_ASSERTS,
                        getUiBackgroundPathInAssets());
        // 可选参数，是否需要用户输入激活码，用于游戏内测阶段。
        // 如果不需激活码相关逻辑，客户传 false 或者不传入该参数。
       intent.putExtra(ProtocolKeys.NEED_ACTIVATION_CODE,
              getCheckBoxBoolean(R.id.isNeedActivationCode));
```

```
        //-- 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));
        // 可选参数，静默自动登录失败后是否显示登录窗口，默认不显示
        intent.putExtra(ProtocolKeys.IS_SHOW_LOGINDLG_ONFAILED_AUTOLOGIN,
                        getCheckBoxBoolean(R.id.isShowDlgOnFailedAutoLogin));
        // 社交分享测试参数，发布时要去掉，具体说明见分享接口
        // intent.putExtra(ProtocolKeys.IS_SOCIAL_SHARE_DEBUG,
        //                 getCheckBoxBoolean(R.id.isDebugSocialShare));
        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：

```
    非离线模式下返回的数据格式：
    {
        "data": {
            "expires_in": "36000",
            "scope": "",
            "refresh_token": "",
            "access_token": "6461171100c3bfa3cba24cc332d7d78b311d2bf590f4877c9" // token
        },
        "errno": 0
    }
    离线模式下返回的数据格式：
    {
     data: {
      mode: "offline" // mode 值为 offline 代表进入离线模式
     },
     errno: 1 // errno 值为 1
    }
```

callback 示例：

```
// 登录接口回调（不支持离线模式）
// 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();
           }

    }
};

// 登录结果回调（支持离线模式）
private IDispatcherCallback mLoginCallbackSupportOffline
  = new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        if (isCancelLogin(data)) {
            return;
        }
        Log.d(TAG, "mLoginCallbackSupportOffline, data is " + data);
        try {
            JSONObject joRes = new JSONObject(data);
            JSONObject joData = joRes.getJSONObject("data");
            String mode = joData.optString("mode", "");
            if (!TextUtils.isEmpty(mode) && mode.equals("offline")) {
                Toast.makeText(SdkUserBaseActivity.this,
                    "login success in offline mode",
                     Toast.LENGTH_SHORT).show();
```

```
                mIsInOffline = true;
                // 显示一下登录结果
                Toast.makeText(SdkUserBaseActivity.this, data,
                    Toast.LENGTH_LONG).show();
            } else {
                mLoginCallback.onFinished(data);
            }
        } catch (Exception e) {
            Log.e(TAG, "mLoginCallbackSupportOffline exception", e);
        }
    }
};
```

3.2.2.2 切换账号接口【客户端调用】

功能说明：

游戏方需要在游戏的菜单中添加“切换账号”的入口，方便用户切换账号。

应用调用 360SDK 切换账号接口，360SDK 显示登录页面，用户可更换账号进行登录。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 的切换账号接口
     *
     * @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

3.2.2.3 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.

接口地址为 https://openapi.360.cn/user/me

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=12345678983b38aabcdef387453ac8133ac326398
7654321&fields=id,name,avatar,sex,area
```

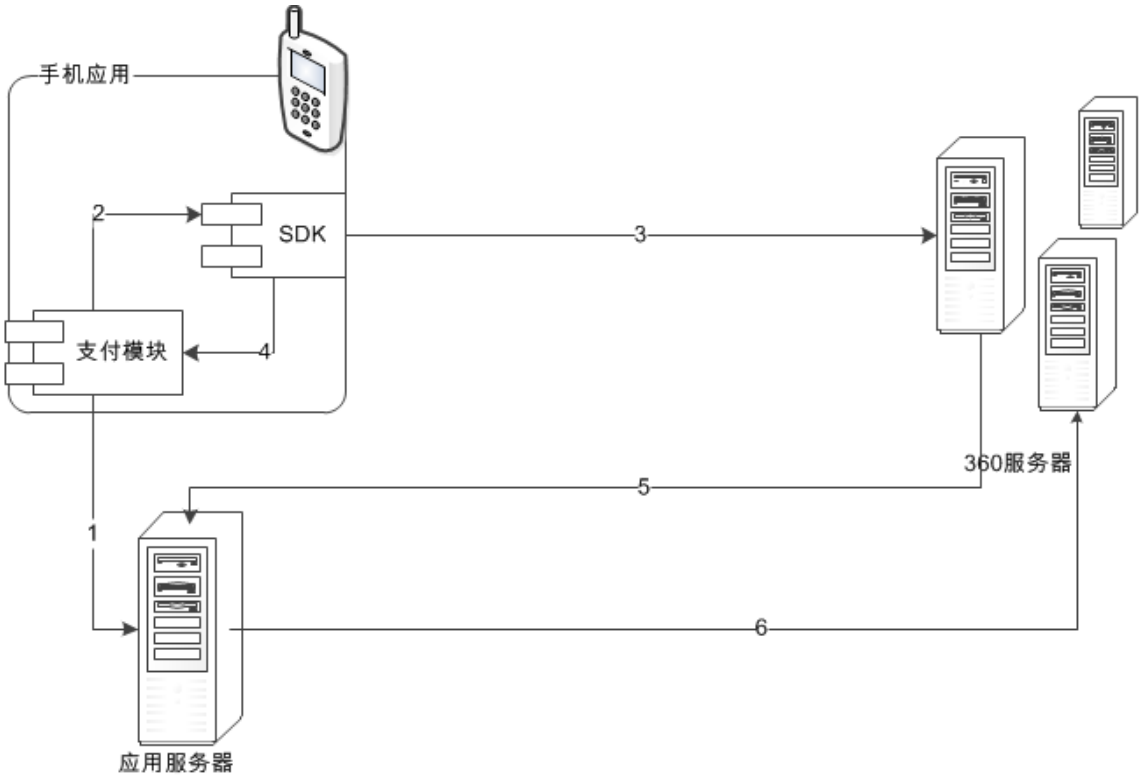
返回示例:

```
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": ""
}
```

获取用户信息后，应用需要保存自身账号与 360 账号的绑定关系。并且妥善保存用户信息留待以后使用.

3.3 支付流程

3.3.1 流程介绍



1. 应用调用应用服务器进行下单；
2. 应用调用 360SDK 支付接口；
3. 360SDK 展示支付页面，引导用户完成支付流程；
   a. 若调用接口时指定金额，则显示固定金额支付界面；
   b. 若调用接口时不指定金额，则显示不固定金额的支付界面；
4. 支付结束或退出 360SDK 支付客户端界面后，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 服务器回调应用服务器上的通知接口，通知支付结果；
6. （可选）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；

3.3.2 接口介绍

3.3.2.1 支付接口【客户端调用】（必接）

功能说明：

应用调用 360SDK 支付接口时，360SDK 弹出支付选择界面。用户在界面上完成支付。

关于应用方订单号的问题：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，都只能支付一次。这样做是为了避免重复支付。通知应用方加钱时，会返回应用订单号，同时提供 360 订单号。

Access token 由于与当前登录用户 id 绑定，因此可以加强支付安全性。但要注意 token 的时间期限（有效期为 10 小时）。过期后调用支付接口会失败。游戏可以引导用户重新登录.

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。**

```
        /**
         * 使用 360SDK 的支付接口
         *
         * @param isLandScape 是否横屏显示支付界面
         * @param isFixed 是否定额支付
         */
    protected void doSdkPay(final boolean isLandScape, final boolean isFixed) {

            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
            Intent intent = getPayIntent(isLandScape, isFixed);

            // 必需参数，使用 360SDK 的支付模块。
            intent.putExtra(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_PAY);

            // 可选参数，登录界面的背景图片路径，必须是本地图片路径
            intent.putExtra(ProtocolKeys.UI_BACKGROUND_PICTRUE, "");

            Matrix.invokeActivity(this, intent, mPayCallback);
        }

        /**
         * 生成调用 360SDK 支付接口基础参数的 Intent
         *
         * @param isLandScape 是否横屏显示登录界面
```

```
     * @param isFixed     是否定额支付
     *
     * @return Intent
     */
    protected Intent getPayIntent(boolean isLandScape, boolean isFixed) {

        Bundle bundle = new Bundle();

        QihooPayInfo pay = getQihooPayInfo(isFixed);

        // 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, "");

        // *** 以下非界面相关参数 ***
        // 设置 QihooPay 中的参数。
        // 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, pay.getAccessToken());

        // 必需参数，360 账号 id。
        bundle.putString(ProtocolKeys.QIHOO_USER_ID, pay.getQihooUserId());

        // 必需参数，所购买商品金额，以分为单位。金额大于等于 100 分，360SDK 运行定额支付流程； 金额数为 0，360SDK 运行不定额支付流程。
        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 16 字符。
        bundle.putString(ProtocolKeys.PRODUCT_ID, pay.getProductId());

        // 必需参数，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。
        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 账号和应用账号，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 账号和应用账号，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());

            // 必需参数，应用订单号，应用内必须唯一，最大 32 字符。
            bundle.putString(ProtocolKeys.APP_ORDER_ID, pay.getAppOrderId());

            // 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());

            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。
                    // 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();
            }
        }
    };
```

3.3.2.2 支付结果通知接口-应用服务器提供接口，由 360 服务器回调（必接）

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）．应用接收验证参数后，给用户做游戏内充值．

应用服务端通知接口在接收到通知消息后，需回应 ok(仅返回小写 ok 这两个字母，不要有其它输出)，表示通知已经接收．如果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．这个机制用来避免掉单。

应用应做好接收到多次通知的准备，防止多次加钱．同时，需要特别注意的是，回应的 ok 表示应用已经正常接到消息，无需继续发送通知．它不表示订单成功与否，或者应用处理成功与否．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，也要返回 ok(仅返回小写 ok 这两个字母，不要有其它输出)．否则，360 会认为未成功通知，会继续发送通知．
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	总价，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	Y	bigint unsigned	20	360 账号 id	Y
order_id	Y	bigint unsigned	20	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:

```
order_id=121109001234567890 1&app_key=1234567890abcdefghijklmnopqrstuv&product_id=p1&amoun
t=101&app_uid=123456789&app_ext1=XXX201211091985&app_order_id=order1234&user_id=987654321&sig
n_type=md5&gateway_flag=success&sign=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&sign_return=xxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
```

样例的签名字段排列（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
```

样例的签名串

```
101#XXX201211091985#1234567890abcdefghijklmnopqrstuv#order1234#123456789#success#12110900
12345678901#p1#md5#987654321#应用 app_secret
```

3.3.2.3 订单核实接口 - 服务器端接口，应用服务器调用（可选）

1. 验证接口地址为：http://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(4.2.2 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.

接口地址:
http://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 6.1 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}"}
```

返回结果中可能的错误信息包括

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 360SDK 社交接口

3.4.1 接口说明

- 360 社交游戏 SDK 无界面纯数据接口通过调用 Matrix.execute 来实现。有界面的接口通过 Matrix.invokeActivity 来实现调用。
- 所有 360 社交 sdk 的接口需要在 UI 主线程里调用。调用接口是传入的参数类型需要与 demo 代码中的参数类型保持一致。

上传积分：

游戏可自定义游戏中用于排行的数值（一般为分数或等级），上传到360服务器存储，用于后面进行排行榜展示。通常竞技类游戏在用户玩完一局后上传分数；等级排名的游戏，在用户每次登录成功后将用户最新等级上传。默认排行榜 id=0，数值在每周三上午10点清零。游戏可以定义多个不同排行榜，在上传对应排行数值的时候指定对应排行榜 id 即可（最多自定义10个排行榜，id 为1~10）。

好友排行榜：

接入方案一：采用 SDK 自带界面的好友排行榜
调用 SDK『带界面的排行榜接口』，游戏方无需绘制界面，即可实现游戏内好友排行榜的展示。
注意：调该接口前提是，游戏必须调用『上传积分接口』将玩家用于排行的数值上传到360服务器。
接入方案二：
游戏方调用『好友排行榜接口』拿到360返回的好友列表之后，到游戏服务器查询分数并进行排序，然后给用户呈现好友排行榜。游戏方可以自定义界面展现排行榜。

邀请好友：

接入方案一：采用 SDK 自带界面的邀请好友
调用 SDK『带界面的邀请好友接口』
接入方案二：游戏自制邀请界面
游戏内需要设置邀请好友按钮，点击后调用『可邀请好友列表接口』，360将返回玩家的可邀请好友列表，包含推荐好友和通讯录朋友两部分（通过 group 字段区分），其中推荐好友是玩家平台好友中的非本游戏好友。游戏方自定义邀请好友界面，区分展示这两部分。可邀请列表上设置邀请按钮，点击后调用 SDK『邀请好友接口』

微博分享：

SDK 提供分享接口，该接口目前支持微信、微博等主要分享渠道。
通过分享接口分享出去的应用，在用户下载后，会自动和分享出该应用的用户成为好友。在登录回调结果中，会返回当前用户最近两次登录期间，通过分享新增的好友信息。

为鼓励玩家分享，游戏可在获得登录回调结果后，给予用户奖励。

3.4.2 接口介绍

3.4.2.1 获取社交初始化信息接口

功能说明：

需要登录完成后才能调用。用户获取用户社交初始化相关信息。主要包括用户资料、新增好友等信息。注意：首次接入，可能接口会返回报错 invalid param appid 400，这是因为 SDK 有个后台加白机制，需要等待 10 分钟后完成。此类报错一个应用只会报错 1 次。如 10 分钟后还会报错，可参看官网 FAQ 详细排查流程：http://dev.360.cn/wiki/index/id/74

接口示例：

```
// 生成 intent 参数
private Intent getSocialInitInfoIntent() {
    Intent intent = new Intent();
    // 必须参数，标识通知 SDK 要执行的功能
    intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                    ProtocolConfigs.FUNC_CODE_GET_SOCIAL_INIT_INFO);
    return intent;
}

// 调用接口
protected void doSDKGetSocialInitInfo(QihooUserInfo usrInfo) {
    if (!checkLoginInfo(usrInfo)) {
        return;
    }

    Intent intent = getSocialInitInfoIntent();
    Matrix.execute(this, intent, new IDispatcherCallback() {

        @Override
        public void onFinished(String data) {
            Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
        }
    });
}
```

返回数据格式：

```
{
```

```
        "data": {
            "shareapp_result": 0,
            "nick": "dklsokdlslkd",
            "shareapp_source": {},
            "accessinfo": {
                "user_me": {
                    "avatar": "http://quc.qhimg.com/dm/100_100_100/t01cd9e9229573bdde8.jpg",
                    "name": "wtpr_2014"
                }
            },
            newfriends: [ // 新增好友
            {
                qid: "105883655", // 新增好友的 qid
                nick: "吉米舟舟", // 新增好友的昵称
                phone: "87d7e322b88cb6a893e91a9981ab43f0", // 新增好友的 phone
                avatar: "http://u1.qhimg.com/qhimg/1/2123.e6264a.jpg" // 新增好友的头像
                src:"sdk", // 好友来源，可能取值：sdk gamebox
                is_played_this_game:1 // 是否玩过此游戏，1，玩过；0，没玩过
            }
            ],
            shareapp_newfriend:{ // 通过游戏分享，新增的好友
                total: 3, // 通过游戏分享新增的好友总数
                users: [  // 通过游戏分享新增的好友相关信息，目前最多返回前 10 个
                {
                    "username": "demotest001", // 用户名
                    "area": "北京市", // 地区
                    "nick": "demo_test_user", // 昵称
                    "avatar": "http://xxxxx.xxx.xx/x.jpg", // 头像
                },
                …
                ]
            },
            "no_self_display_nick": "dklsokdlslkd",
            "app_inviters": 0
        },
        "time": 1425266350,
        "errno": 0,
        "errmsg": "ok",
    }
```

callback 示例：

```
    new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
```

```
            if (null == data) {
                return;
            }
            Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
            System.out.println(data);
        }
    });
```

3.4.2.2 获取可邀请好友列表接口（与邀请好友接口配合使用）

功能说明：

需要获取可邀请好友列表时，调用此接口。需要先登录。该接口会上传本地通信录到服务器，通过服务器结合本地通信录内容，返回可邀请的好友列表。

接口示例：

```
    /**
     * 使用 360SDK 的获取可邀请好友列表接口
     */
    // 生成 intent 参数
private Intent getGetContactContentIntent(TokenInfo token, boolean isLandScape){
    Intent intent = new Intent();
    //必须参数，屏幕方向
    intent.putExtra(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
    //必须参数，表示调用 SDK 接口执行的功能
    intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                    ProtocolConfigs.FUNC_CODE_GET_CONTACT_CONTENT);
    //可选参数，表示此次获取可邀请好友列表的开始位置，从 0 开始。
    //    无此参数时会获取所有的可邀请好友列表，不进行翻页。
    intent.putExtra(ProtocolKeys.START, mEditGetcontactStart.getText().toString());
    //可选参数，表示此次获取可邀请好友的个数。
    //    无此参数时会获取所有的可邀请好友列表，不进行翻页。
    intent.putExtra(ProtocolKeys.COUNT, mEditGetcontactCount.getText().toString());
    return intent;
}
// 调用接口
protected void doSdkGetContactContent(QihooUserInfo usrInfo, boolean isLandScape ){
    if(!checkLoginInfo(usrInfo)) {
        return;
    }
    mEditGetcontactStart = (EditText)findViewById(R.id.edit_getcontact_start);
    mEditGetcontactCount = (EditText)findViewById(R.id.edit_getcontact_count);
    Intent intent = this.getGetContactContentIntent(isLandScape);
    Matrix.execute(this, intent, new IDispatcherCallback() {
        @Override
            public void onFinished(String data) {
            Toast.makeText(SdkUserBaseActivity.this, data,
                           Toast.LENGTH_SHORT).show();
            System.out.println(data);
        }
    });
}
```

返回数据格式：

```
{
  errmsg:"ok",
  time:1367930392,
  errno:0,  // 服务器返回的错误码，为 0 代表成功。
  // data 是一个 json 数组，其中每个对象保存可邀请的好友信息
  data:[{
    is_friend:0, // 是否为好友
    qid:"278772014", // 好友的 qid，可能为空字串
    phone:" 87d7e322b88cb6a893e91a9981ab43f0",  // 好友的电话号码，可能为空字符串
    is_invited:0, // 是否邀请过该好友
    nick:"vrs132v", // 通信录中姓名 > nickname > 无名大侠+qid > 无名大侠
    avatar: "http://u1.qhimg.com/qhimg/quc/...01259qd123.e6264a.jpg", // 头像
    last_invited_time :1367930392, //上次邀请的时间
    is_played_this_game:0, // 用户是否玩过此游戏，1，玩过；0，没玩过。
    group:"contacts", // 标识用户属于哪个分组，可取的值：contacts(来自通信录)recommend(推荐玩家)
  }]
}
```

callback 示例：

```
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
        System.out.println(data);
    }
});
```

3.4.2.3 邀请好友接口

功能说明：

邀请好友时，调用此接口。用户必须已登录才能调用。

接口示例：

```
    /**
     * 使用 360SDK 的邀请好友接口
     */
    // 生成 intent 参数
private Intent getInviteFriendIntent(boolean bLandScape){
    Intent intent = new Intent();
    //必须参数，表示调用 SDK 接口执行的功能为批量邀请好友
    intent.putExtra(ProtocolKeys.FUNCTION_CODE,
```

```
                    ProtocolConfigs.FUNC_CODE_INVITE_FRIEND_BATCH);
        //必须参数，对方的昵称
        intent.putExtra(ProtocolKeys.NICK_NAME, strNick);
        //有值就要传入，加密后的好友电话。使用“获取可邀请好友列表”接口返回的 phone 字段中的值。
        intent.putExtra(ProtocolKeys.PHONE, strPhone);
        //有值就要传入，被邀请用户的 qid，如果没有可以不传
        intent.putExtra(ProtocolKeys.QID, strQid);
        //必须参数，邀请信息内容。
        intent.putExtra(ProtocolKeys.SMS, strSMS);
        //屏幕方向，bool 值，true 为横屏，flase 为竖屏
        intent.putExtra(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, bLandScape);
        return intent;
    }
    // 调用接口
    protected void doSdkInviteFriend(QihooUserInfo usrInfo, boolean bLandScape){
        // 检查用户是否登录
        if(!checkLoginInfo(usrInfo)) {
            return;
        }
        Intent intent = getInviteFriendIntent(bLandScape);
        Matrix.execute(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
               Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
               System.out.println(data);
            }
        });
    }
```

返回数据格式：

```
    {
      "errno": 0,
      "errmsg": "ok",
      "time": 1363155828,
      "data": {
        "status": "1"  //0：邀请失败，1：邀请成功
      }
    }
```

callback 示例：

```
    new IDispatcherCallback() {
           @Override
```

```
            public void onFinished(String data) {
                Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
                System.out.println(data);
            }
        });
```

3.4.2.4 带界面的邀请好友接口

功能说明：

该接口提供一个界面用于展示可邀请好友列表，并能对每个好友进行邀请。当界面销毁时会通过接口的回调通知调用者。必须在登陆状态下才能调用此接口。

接口示例：

```
    /**
     * 使用 360SDK 的带界面的邀请好友接口
     */
    // 生成 intent 参数
private Intent getInviteFriendBySdkIntent(boolean bLandScape) {
    Intent intent = new Intent(this, ContainerActivity.class);
    //必须参数，表示调用 SDK 接口执行的功能
    intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                    ProtocolConfigs.FUNC_CODE_INVITE_FRIEND_BY_SDK);
    // 可选参数，360SDK 界面是否以横屏显示。
    intent.putExtra(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, bLandScape);
    // 可选参数，邀请信息，不需要主语，sdk 会添加。
    intent.putExtra(ProtocolKeys.SMS, inviteMsg);
    return intent;
}
// 调用接口
protected void doSdkInviteFriendBySdk(QihooUserInfo usrinfo, boolean bLandScape) {
    if (!checkLoginInfo(usrinfo)) {
        return;
    }
    Intent intent = getInviteFriendBySdkIntent(bLandScape);
    Matrix.invokeActivity(this, intent, new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            System.out.println("result: " + data);
            Toast.makeText(SdkUserBaseActivity.this, "result: " + data,
                           Toast.LENGTH_SHORT).show();
        }
    });
}
```

返回数据格式：

```
{
  "errno":0,
  "errmsg":"finish",
  invite_count:1 // 用户在成功发出邀请的计数
}
```

callback 示例：

```
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
        System.out.println(data);
    }
});
```

3.4.2.5 上传积分接口（若接入社交相关功能，则接口必接）

功能说明：

上传用户积分/等级到服务器，更新用户积分和排行。必须在登录状态下才能调用此接口。

接口示例：

```
    /**
     * 使用 360SDK 的上传积分接口
     */
// 生成 intent 参数
private Intent getUploadScoreIntent(){
    Intent intent = new Intent();
    //必须参数，表示调用 SDK 接口执行的功能为上传积分
    intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                    ProtocolConfigs.FUNC_CODE_UPLOAD_SCORE);
    //必须参数，用户积分。
    intent.putExtra(ProtocolKeys.SCORE, strScore);
    //必须参数，数排行榜标识
    intent.putExtra(ProtocolKeys.TOPNID, topnid);
    return intent;
}
// 调用接口
protected void doSdkUploadScore(QihooUserInfo usrInfo){
    // 检查用户是否登录
    if(!checkLoginInfo(usrInfo)) return;
    Intent intent = this.getUploadScoreIntent();
```

```
        Matrix.execute(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
                Toast.makeText(SdkUserBaseActivity.this, data,
                               Toast.LENGTH_SHORT).show();
            }
        });
    }
```

返回数据格式：

```
{
  errno: 0,
  errmsg: "ok",
  time: 1363155828,
  data: {
    status: "1"  //0: 更新失败, 1: 更新成功
  }
}
```

callback 示例：

```
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        if (null == data) {
            return;
        }
        Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
    }
});
```

3.4.2.6 好友排行榜接口

功能说明：

用于获取游戏中的好友之间的排行榜，必须在登录状态下才能调用此接口。

接口示例：

```
    /**
     * 使用 360SDK 的好友排行榜接口，生成 intent 参数
     */
    private Intent getGetGameTopFriendIntent() {
        Intent intent = new Intent();
        //必须参数，表示调用 SDK 接口执行的功能为游戏的排行榜。
```

```
        intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                        ProtocolConfigs.FUNC_CODE_GET_RANK_FRIENDS);
        //可选参数，排名榜的第几位开始获取(索引从 0 开始)
        intent.putExtra(ProtocolKeys.START, strStart/*"0"*/);
        //可选参数，排名榜上从 start 开始，获取多少位,最小值为 20。
        //    start 和 count 参数如果不传的话，会返回最多前 20 个排行榜内容。
        intent.putExtra(ProtocolKeys.COUNT, strCount/*"20"*/);
        // 可选参数，排行榜类型（不填写，则为默认的排行榜）
        intent.putExtra(ProtocolKeys.TOPNID, strTopNID);
        // 可选参数，排序类型（填写为 0 则按正序排序，其他情况按倒序排序）
        intent.putExtra(ProtocolKeys.ORDERBY, strOrderby);
        return intent;
    }

    // 调用接口
    protected void doSdkGetRankFriend(QihooUserInfo usrInfo){
        // 检查用户是否登录
        if(!checkLoginInfo(usrInfo)) {
            return;
        }
        Intent intent = this.getGetGameTopFriendIntent();
        Matrix.execute(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
               Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
            }
        });
    }
```

返回数据格式：

```
{
 errno: 0,
 errmsg: "ok",
 time: 1369627632,
 data: {
  self: {     // 当前登录的帐号
   qid: "17699179", // 自己的 qid
   nick: "缤纷翼彩", // 自己的昵称
   phone: "87d7e322b88cb6a893e91a9981ab43f0", // 自己的电话号码
   avatar:"http://u1.qhimg.com/qhimg/...259qd123.e6264a.jpg", // 头像
   score: 0,// 自己的分数
   rank: 4 // 自己在此排行版中的排名
  },
  all: [ // 我的好友的分数、排名（包括自己在内）
```

```
    {
     qid: 273002130, // qid
     nick: "idwanglu2010", // 昵称
     score: 0, // 分数
     phone: "",
     rank: 1, // 排名
     avatar:"http://u1.qhimg.com/qhimg/quc/80_8...qd123.e6264a.jpg", // 头像
     src:"sdk", // 好友来源，可能取值：sdk，gamebox。如果是自己的话值为 self
     is_played_this_game:1 // 是否玩过此游戏，1，玩过；0，没玩过
    }
   ],
   count: 2 //all 数组中的项目数
  }
 }
```

callback 示例：

```
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        if (null == data) {
            return;
        }
        Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
    }
});
```

3.4.2.7 全球排行榜接口

功能说明：

用于获取游戏中所有玩家之间的排行榜，返回排名前 100 的用户，必须在登录状态下才能调用此接口。

接口示例：

```
    /**
     * 使用 360SDK 的全球排行榜接口，生成 intent 参数
     */
private Intent getGetGameGlobleRankListIntent(){
    Intent intent = new Intent();

    //必须参数，表示调用 SDK 接口执行的功能为游戏的排行榜。
    intent.putExtra(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_GET_RANK);
    //可选参数，排行榜类型（不填写，则为默认的排行榜）
    intent.putExtra(ProtocolKeys.TOPNID, strTopNID);
```

```
        //可选参数，排序类型（填写为 0 则按正序排序，其他情况按倒序排序）
        intent.putExtra(ProtocolKeys.ORDERBY, StrOrderby);
        return intent;
    }
    // 调用接口
    protected void doSdkGetGlobalRankList(QihooUserInfo usrInfo){
        // 检查用户是否登录
        if(!checkLoginInfo(usrInfo)) {
            return;
        }
        Intent intent = this.getGetGameTop100Intent();
        Matrix.execute(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
                Toast.makeText(SdkUserBaseActivity.this, data,
                               Toast.LENGTH_SHORT).show();
            }
        });
    }
```

返回数据格式：

```
{
errno: 0,
errmsg: "ok",
time: 1369629092,
data: {
  all: [
    {
      qid: 282713167,
      nick: "tks264test100",
      phone: "",
      rank: 1,
      score: "100",
      isfriend: 0   // 对方是不是自己的好友，0 不是；1 是
      avatar: "http://u1.qhimg.com/qhimg/...259qd123.e6264a.jpg" // 头像
      src:"sdk",    // 好友来源，可能取值：sdk gamebox 如果是自己的话值为 self
      is_played_this_game:1  // 是否玩过此游戏，1，玩过；0，没玩过
    },
        ...
  ],
  self: {
    qid: "17699176",
    nick: "缤纷翼彩",
    phone: "87d7e322b88cb6a893e91a9981ab43f9",
```

```
        rank: 3
        avatar: "http://u1.qhimg.com/qhimg/quc/80_80/29/01/25/2901259qd123.e6264a.jpg"
      },
      count: 2, //返回的数据记录数
      total: 2 //全球往此榜单提交过分数的人数总数
    }
  }
```

callback 示例：

```
    new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            if (null == data) {
                return;
            }
            Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
        }
    });
```

3.4.2.8 带界面的排行榜接口

功能说明：

该接口提供一个界面用于展示好友排行。当界面销毁时会通过接口的回调通知调用者。必须在登录状态下才能调用此接口。

接口示例：

```
    /**
     * 使用 360SDK 的带界面的排行榜接口，生成 intent 参数
     */
    private Intent getDisplayGameFriendRankIntent(boolean bLandScape) {
       Intent intent = new Intent(this, ContainerActivity.class);
        // 必须参数，表示调用 SDK 接口执行的功能
        intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                        ProtocolConfigs.FUNC_CODE_DISPLAY_GAME_FRIEND_RANK);
       if (!TextUtils.isEmpty(topnId)) {
              // 可选参数，排行榜 ID，与上传积分时对应的排行榜 ID 对应，
              // 使用该参数，需要确定后台存在对应 ID 的排行榜，否则会显示错误
              intent.putExtra(ProtocolKeys.TOPNID, topnId);
        }
        // 360SDK 界面是否以横屏显示。
        intent.putExtra(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, bLandScape);
        return intent;
```

```
    }

    // 调用接口
    protected void doSdkDisplayGameFriendRank(QihooUserInfo usrInfo,
           boolean bLandScape) {
        if (!checkLoginInfo(usrInfo)) {
            return;
        }
        Intent intent = getDisplayGameFriendRankIntent(bLandScape);
        Matrix.invokeActivity(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
                System.out.println("result: " + data);
                Toast.makeText(SdkUserBaseActivity.this, "result: " +
                    data, Toast.LENGTH_SHORT).show();
            }
        });
    }
```

返回数据格式：

```
    {
      errno:0,
      errmsg:"finish",
      invite_count:1 // 用户在成功发出邀请的计数
    }
```

callback 示例：

```
    new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
        }
    });
```

3.4.2.9 获取个人信息资料页接口

功能说明：

用于获取个人资料页信息。必须在登录状态下才能调用此接口。

接口示例：

```
        /**
```

```
     * 使用 360SDK 的获取个人信息资料页接口
     */
    // 生成 intent 参数
    private Intent getUserInfoIntent() {
        Intent intent = new Intent();
        // function code
        intent.putExtra(ProtocolKeys.FUNCTION_CODE,
                        ProtocolConfigs.FUNC_CODE_GET_USER_INFO);
        // 用户 QID
        intent.putExtra(ProtocolKeys.QID, qid);
        return intent;
    }
    // 调用接口
    protected void doSdkGetUserInfo(QihooUserInfo usrinfo) {
        if (!checkLoginInfo(usrinfo)) {
            return;
        }
        Intent intent = getUserInfoIntent();
        Matrix.execute(this, intent, new IDispatcherCallback() {
            @Override
            public void onFinished(String data) {
                System.out.println(data);
                Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_LONG).show();
            }
        });
    }
```

返回数据格式：

```
{
    errno: 0,
    errmsg: 'ok',
    time: 1384149855,
    data:
    {
        userinfo: {
            user_id: "668774323",
            phoneenc: "ecea6c1ce9f58cdd73bd518d7a154cc7",
            nick: "shortytall",
            face: "http://quc.qhimg.com/dm/100_100_...32792671719a.jpg", // 头像
            face_large: "http://quc.qhimg.com/dm/180_180_..71719a.jpg",
            gender: "男",
            age: 19,
            astro: "双子座",
            birthday: "1973-6-13",
```

```
            province: "云南省",
            city: "红河",
            weibonick: "",
            weibouid: "",
            motto: "", //个性签名
            role: "self", //可能的值: self/friend/pursuer/stranger
            recentgames: [ //最近在玩的游戏, 最多 4 个
                {
                    soft_id: "123456",
                    pname: "com.xxxxxx.xxx",
                    logo_url: "http://p18.qhimg.com/dr/3...7fdb9be.png",// 游戏 logo
                    name: "游戏名称 1",
                    download_times: "459077"
                },
                {
                    soft_id: "234567",
                    pname: "org.xxx.xxx",
                    logo_url: "http://p18.qhimg.com/dr/38_...7fdb9be.png",
                    name: "游戏名称 2",
                    download_times: "22155810"
                },
                ...
            ]
        }
    }
}
```

callback 示例：

```
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
    }
});
```

3.4.2.10 分享接口

功能说明：

需要分享应用时，调用此接口。该接口目前支持微信、微博等主要分享渠道。必须在登录状态下才能调用此接口。

通过分享接口分享出去的应用，在用户下载后，会自动和分享出该应用的用户成为好友。在登录回调结果中，会返回当前用户最近两次登录期间，通过分享新增的好友信息。具体见登录接口返回数据说明中的“shareapp_newfriend”字段。

为鼓励玩家分享，游戏可在获得回调结果后，给于用户奖励。

接口示例：

```
// 生成分享接口 intent 参数
private Intent getShareIntent(String title, String desc, String picture,
                              String icon, String uibg, boolean isLandScape){
    Intent intent = new Intent();
    // 必须参数，标识通知 SDK 要执行的功能
    intent.putExtra(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_SHARE);
    // 可选参数，指定横竖屏，默认为横屏
    intent.putExtra(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
    if (!TextUtils.isEmpty(uibg)) {
        // 可选参数，分享界面的背景图，不传就是透明
        intent.putExtra(ProtocolKeys.UI_BACKGROUND_PICTRUE, uibg);
    }
    // 必须参数，分享的标题，长度不可超过 512，这个字段会作为微信网页分享的标题使用
    intent.putExtra(ProtocolKeys.SHARE_TITLE, title);
    // 必须参数，分享的描述，该内容可能会通过短信、微博分享发出，
    // 长度请限制在 140 字符范围内。
    intent.putExtra(ProtocolKeys.SHARE_DESC, desc);
    // 可选参数，分享的图片路径，微博分享时会使用
    // （必须是本地路径如：/sdcard/1.png，后缀可以是 png、jpg、jpeg，
    //  大小不能超过 5M，尺寸不能超过 1280x720）
    intent.putExtra(ProtocolKeys.SHARE_PIC, picture);
    // 可选参数，分享的 icon 路径，微信分享时会使用
    // （必须是本地路径，最好是 png 文件，32k 以内）
    intent.putExtra(ProtocolKeys.SHARE_ICON, icon);
    return intent;
}
// 调用接口
protected void doSdkShare(QihooUserInfo usrInfo, final boolean bLandScape){
    if(!checkLoginInfo(usrInfo)) {
        return;
    }
    Intent intent = getShareIntent(etTitle.getText().toString(),
    etDesc.getText().toString(), etPic.getText().toString(),
    etIcon.getText().toString(), etUiBg.getText().toString(), bLandScape);
    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();
                }
            });
        }
```

返回数据格式：

```
    /*
     * errno:  错误码：0，分享完成，但不确定是否已经发出分享；1，分享已经发出；-1，分享失败
     * errmsg: 错误信息
     * shared: bool 类型，分享是否已经发出，true 为确定分享已经发出
     * share_way: 用户选择的分享渠道：
     *           clipboar，复制到剪切板；
     *           weixin_timeline，微信朋友圈；
     *           weixin_friends，微信好友；
     *           sina_weibo，新浪微博；
     *           sms，短信
     */
    {
        "errno": 0,
        "shared": false,
        "errmsg": "ok",
        "share_way": "clipboard",
    }
```

callback 示例：

```
    new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            if (null == data) {
                return;
            }
            Toast.makeText(SdkUserBaseActivity.this, data, Toast.LENGTH_SHORT).show();
            System.out.println(data);
        }
     });
```

测试：

- 分享：在应用未上线时，360 没有对应的应用下载链接，游戏只需测试分享内容成功发送到各社交渠道即可。如调试阶段收到分享失败的错误码提示，请到对应渠道官网查看错误码说明：
  微信：https://open.weixin.qq.com/zh_CN/htmledition/res/dev/document/sdk/android/index.html
  微博：http://open.weibo.com/wiki/Error_code

- 加好友：测试阶段如需模拟登录回调中返回通过分享新增的好友信息，可在登录接口调用时加入参数 ProtocolKeys.IS_SOCIAL_SHARE_DEBUG，传值为 true，这样会认为当前应用为 demotest001 分享出来的。登录测试帐号 demotest002~demotest010，会自动和 demotest001 这个帐号成为好友。当登录 demotest001 这个帐号时，会在登录结果中返回通过分享新增的好友信息。我们提供的测试账号有：

帐号	密码
demotest001	q111111
demotest002	q111111
demotest003	q111111
demotest004	q111111
demotest005	q111111
demotest006	q111111
demotest007	q111111
demotest008	q111111
demotest009	q111111
demotest010	q111111

3.5 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);
}
```

3.6 360 支付 SDK 其他接口

3.6.1 退出接口【客户端调用】(必接)

功能说明:

“退出游戏”，将直接调用退出回调函数。

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
}
};
```

3.6.2 打开论坛接口【客户端调用】（可选）

功能说明：

打开论坛接口，为游戏用户提供分享、吐槽的地方，能够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);
}
```

3.6.3 防沉迷查询接口【客户端调用】（可选）

功能说明：
防沉迷系统是中国法律对网游管理的要求，因此游戏必须接入防沉迷查询接口。
使用方法：游戏方调用防沉迷查询接口，查询该用户是否已经成年。
若已成年，则允许用户正常游戏，不做任何提醒。
若未成年；或未实名注册的用户将受到防沉迷系统的限制.
未实名注册的用户，需要调用实名注册接口，要求用户进行实名注册。
开发细则：
游戏过程，会提示游戏用户的累计在线时间。
累计游戏时间超过 3 小时，游戏收益（经验，金钱）减半。此后，每 30 分钟警示一次。累计游戏时间超过 5 小时，游戏收益为 0。此后，每 15 分钟警示一次。
如果未成年人的累计下线时间已满 5 小时，则累计在线时间清零，如再上线则重新累计在线时间。

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 账号 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

```
{"content":{"ret":[{"qid":"199062142","status":"2"}]},"error_code":"0","error_msg":""}
```

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，无此用户数据；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:无此用户数据
                                    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();
        }
    }
}
```

3.6.4 实名注册接口【客户端调用】（可选）

功能说明：

提供实名注册接口，当游戏方调用防沉迷查询接口时，发现用户未填写实名注册信息时，可调用该接口发起实名注册。

接口示例及参数说明：

```
/**
 * 使用 360SDK 实名注册接口
 *
 * @param isLandScape 是否横屏显示登录界面
 * @param qihooUserId 奇虎 360 用户 ID
 */
protected void doSdkRealNameRegister(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。
    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);

            Matrix.invokeActivity(this, intent, new IDispatcherCallback() {
                @Override
                public void onFinished(String data) {
                }
            });
        }
```

3.6.5 使用 Matrix 的 get 方法，获取基本信息

```
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)
```

3.6.6 游戏关卡信息获取【客户端调用】（可选）

功能说明：

提供在 360 后台配置的关卡信息获取接口，游戏接入方可以根据此接口获取在服务端配置的关卡信息

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
```

```
         */
        protected void doSdkGameLevelQuery(QihooUserInfo userInfo) {

            if (!checkLoginInfo(userInfo)) {
                return;
            }

            Bundle bundle = new Bundle();

            // 必需参数，使用360SDK的游戏关卡查询
            bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_GAME_LEVEL);

            Intent intent = new Intent(this, ContainerActivity.class);
            intent.putExtras(bundle);
            Matrix.execute(this, intent, new IDispatcherCallback() {

                @Override
                public void onFinished(String data) {
                    if (!TextUtils.isEmpty(data)) {
                        try {
                            JSONObject resultJson = new JSONObject(data);
                            int errorCode = resultJson.optInt("errno");
                            if (errorCode == 0) {
                                JSONObject mData = resultJson.getJSONObject("data");
                                if(mData != null) {
                                    //用户关卡信息
                                    String mContent = mData.optString("content");
                                    if(mContent.isEmpty()){
                                        Toast.makeText(SdkUserBaseActivity.this,"没有配置关卡信息
", Toast.LENGTH_SHORT).show();
                                    }else{
                                        Toast.makeText(SdkUserBaseActivity.this,"关卡信息
为:"+mContent, Toast.LENGTH_SHORT).show();
                                    }
                                    return;
                                }
                            } else {
                                Toast.makeText(SdkUserBaseActivity.this,data,
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();
                }
            }
        });
    }
```

4. 数据统计 API 说明

使用 *QHStatDo* 来调用以下各个方法。

4.1 关卡统计 API

开始关卡：

```
// level: 关卡名称
public static void startLevel(String level)
```

结束关卡

```
// level: 关卡名称
public static void finishLevel(String level)
```

失败关卡

```
// level: 关卡名称
public static void failLevel(String level, String reason)
```

4.2 任务统计 API

开始任务

```
// task: 任务名称
// type: 任务类型
public static void startTask(String task, String type)
```

结束任务

```
// task: 任务名称
public static void finishTask(String task)
```

失败任务

```
// task: 任务名称
// reason: 失败原因
public static void failTask(String task, String reason)
```

4.3 支付统计 API

```
// cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
// coin: 购买金币数量
// source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
public static void pay(int cash, int coin, int source)
```

```
// cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
// cashType: 支付货币类型（游戏自定义）
// coin: 购买金币数量
// source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
public static void pay(int cash, String cashType, int coin, int source)
```

```
// cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
// coin: 购买金币数量
// source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
// level: 关卡名称
// rank: 玩家等级
public static void pay(int cash, int coin, int source, String level, String rank)
```

```
// cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
// cashType: 支付货币类型
// coin: 购买金币数量
// source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
```

```
    // level: 关卡名称
    // rank: 玩家等级
    public static void pay(int cash, String cashType, int coin, int source, String level, String
rank)
```

```
    // cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
    // number: 购买道具数量
    // props: 购买道具名称
    // source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
    public static void pay(int cash, int number, String props,  int source)
```

```
    // cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
    // cashType: 支付货币类型
    // number: 购买道具数量
    // props: 购买道具名称
    // source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
    public static void pay(int cash, String cashType, int number, String props,  int source)
```

```
    // cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
    // number: 购买道具数量
    // props: 购买道具名称
    // source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
    // level: 关卡名称
    // rank: 玩家等级
    public static void pay(int cash, int number, String props, int source, String level, String
rank)
```

```
    // cash: 支付金额（游戏自定义，可以是分或元为单位，游戏依照实际需求来定）
    // cashType: 支付货币类型
    // number: 购买道具数量
    // props: 购买道具名称
    // source: 支付方式（游戏自定义，给每个支付渠道定义的整型值）
    // level: 关卡名称
    // rank: 玩家等级
    public static void pay(int cash, String cashType, int number, String props, int source, String
level, String rank)
```

4.4 虚拟币购买物品统计 API

```
// name: 物品名称
// number: 物品数量
// coin: 消费的虚拟币数量
public static void buy(String name, int number, int coin)
```

```
// name: 物品名称
// number: 物品数量
// method: 物品获取方式
public static void buy(String name, int number, String method)
```

```
// name: 物品名称
// number: 物品数量
// coinType: 虚拟币类型
// coin: 消费的虚拟币数量
public static void buy(String name, int number, String coinType, int coin)
```

```
// name: 物品名称
// number: 物品数量
// coin: 消费的虚拟币数量
// level: 关卡名称
public static void buy(String name, int number, int coin, String level)
```

```
// name: 物品名称
// number: 物品数量
// method: 物品获取方式
// level: 关卡名称
public static void buy(String name, int number, String method, String level)
```

```
// name: 物品名称
// number: 物品数量
// coinType: 虚拟币类型
// coin: 消费的虚拟币数量
```

```
// level: 关卡名称
public static void buy(String name, int number, String coinType, int coin, String level)
```

```
// name: 物品名称
// type: 物品类型
// number: 物品数量
// coin: 消费的虚拟币数量
public static void buy(String name, String type, int number, int coin)
```

```
// name: 物品名称
// type: 物品类型
// number: 物品数量
// method: 物品获取方式
public static void buy(String name, String type, int number, String method)
```

```
// name: 物品名称
// type: 物品类型
// number: 物品数量
// coinType: 虚拟币类型
// coin: 消费的虚拟币数量
public static void buy(String name, String type, int number, String coinType, int coin)
```

```
// name: 物品名称
// type: 物品类型
// number: 物品数量
// coin: 消费的虚拟币数量
// level: 关卡名称
public static void buy(String name, String type, int number, int coin, String level)
```

```
// name: 物品名称
// type: 物品类型
// number: 物品数量
// method: 物品获取方式
// level: 关卡名称
```

```
    public static void buy(String name, String type, int number, String method, String level)
```

```
    // name: 物品名称
    // type: 物品类型
    // number: 物品数量
    // coinType: 虚拟币类型
    // coin: 消费的虚拟币数量
    // level: 关卡名称
    public static void buy(String name, String type, int number, String coinType, int coin, String
level)
```

4.5 物品消耗统计 API

```
    // name: 物品名称
    // number: 物品数量
    // coin: 物品对应的虚拟币数量
    public static void use(String name, int number, int coin)
```

```
    // name: 物品名称
    // number: 物品数量
    // coinType: 虚拟币类型
    // coin: 物品对应的虚拟币数量
    public static void use(String name, int number, String coinType, int coin)
```

```
    // name: 物品名称
    // type: 物品类型
    // number: 物品数量
    // coin: 物品对应的虚拟币数量
    public static void use(String name, String type, int number, int coin)
```

```
    // name: 物品名称
    // type: 物品类型
    // number: 物品数量
    // coinType: 虚拟币类型
```

```
    // coin: 物品对应的虚拟币数量
    public static void use(String name, String type, int number, String coinType, int coin)
```

```
    // name: 物品名称
    // number: 物品数量
    // coin: 物品对应的虚拟币数量
    // level: 关卡名称
    public static void use(String name, int number, int coin, String level)
```

```
    // name: 物品名称
    // number: 物品数量
    // coinType: 虚拟币类型
    // coin: 物品对应的虚拟币数量
    // level: 关卡名称
    public static void use(String name, int number, String coinType, int coin, String level)
```

```
    // name: 物品名称
    // type: 物品类型
    // number: 物品数量
    // coin: 物品对应的虚拟币数量
    // level: 关卡名称
    public static void use(String name, String type, int number, int coin, String level)
```

```
    // name: 物品名称
    // type: 物品类型
    // number: 物品数量
    // coinType: 虚拟币类型
    // coin: 物品对应的虚拟币数量
    // level: 关卡名称
    public static void use(String name, String type, int number, String coinType, int coin, String
level)
```

4.6 玩家统计 API

```
    // id: 玩家标识
    // age: 年龄
    // gender: 性别
    // source: 玩家来源（游戏自定义，给每个渠道用户定义的字符串类型值，如:”qihoo360”,”weibo”）
    // rank: 玩家等级
    // server: 区域服务器名称
    // comment: 其他备注信息
    public static void player(String id, int age, int gender, String source, String rank, String server, String comment)
```

4.7 角色统计 API

```
    // role: 当前玩家标识设置的角色名
    public static void role(String role)
```

4.8 自定义事件统计 API

自定义事件

```
    // id: 事件标识
    // map: 事件属性名和值的键值对
    public static void event(String id, Map<String, String> map)
    // 如果在 service 中，请使用包含上下文参数的方法:
    public static void event(Context ctx, String id, Map<String, String> map)
```

开始自定义事件

```
    // id: 事件标识
    // map: 事件属性名和值的键值对
    public static void eventBegin(String id, Map<String, String> map)
    // 如果在 service 中，请使用包含上下文参数的方法:
    public static void eventBegin(Context ctx, String id, Map<String, String> map)
```

结束自定义事件

```
// id: 事件标识
// map: 事件属性名和值的键值对
public static void eventEnd(String id, Map<String, String> map)
// 如果在 service 中，请使用包含上下文参数的方法:
public static void eventEnd(Context ctx, String id, Map<String, String> map)
```

4.9 获取用户在线配置参数 API

```
// key: 参数名
// defValue: 默认参数值
// 返回用户自定义参数值
public static String getConfParameter(Context ctx, String key, String defValue)
```

5. 推送 API 说明

使用 *QPushAgent* 来调用以下各个方法。

5.1 设置标签 API

```
// ctx: 应用上下文
// tags: 应用标签集
// 为用户加上标签集，以便推送时按照标签来筛选。
public static void setTags(Context ctx, Set<String> tags)
```

5.2 设置别名 API

```
// ctx: 应用上下文
// alias: 玩家用户别名
// 通过设置游戏应用自定义的用户 ID，按用户 ID 向用户推送消息。
public static void setAlias(Context ctx, String alias)
```

5.3 设置标签和别名 API

```
// ctx: 应用上下文
// tags: 应用标签集
// alias: 玩家用户别名
// 同时设置用户标签集和用户 ID
public static void setTagsAndAlias(Context ctx, Set<String> tags, String alias)
```

5.4 获取应用本机推送标识号 API

```
// ctx: 应用上下文
// 返回: 应用在本机的推送标识号
// [注]也可以在应用启动后，查看 logcat 日志中的"360 push registerId"信息获取此标识号，例如:
// D/QPushAgent( 8995): -->"360 push registerId": 81FA354710D6F5C7E7450A192FB69FFC
public static String getRegisterId(Context ctx)
```

5.5 获取应用标识号 API

```
// ctx: 应用上下文
// 返回: 应用标识号
public static String getAppId(Context ctx)
```

6.附录:

6.1 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);
…
```

6.2 Demo 工程简介

360SDK 工程结构如图所示：

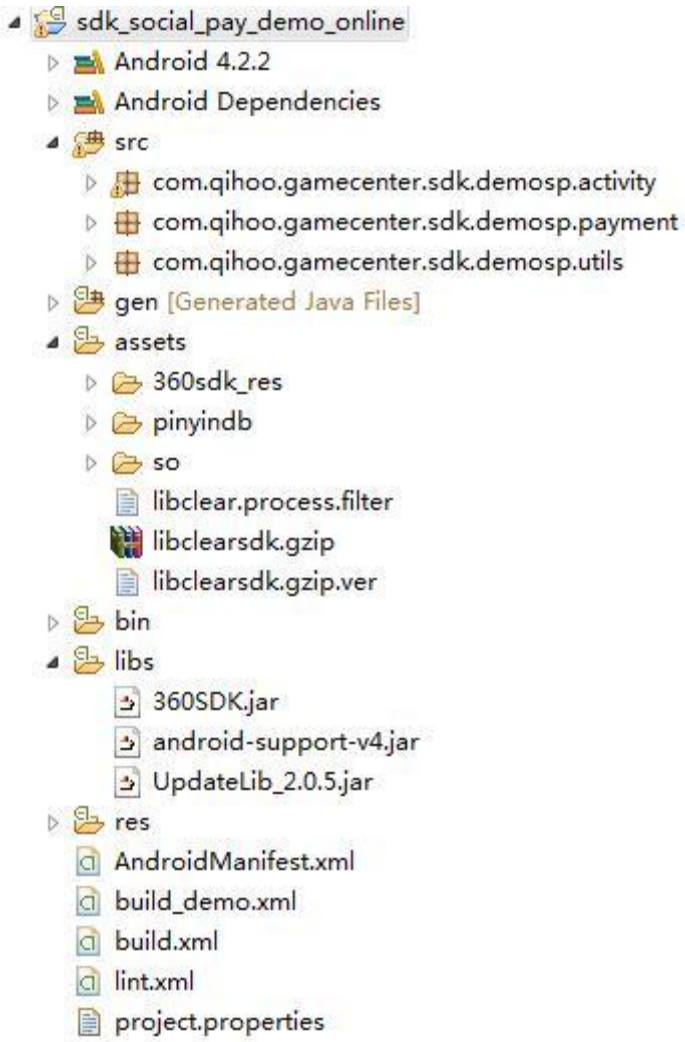


图 Demo 工程

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:

QihooUserInfo，360 用户信息数据。

Utils.java，一些工具函数。

6.3 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。

6.4 附加信息

自检工具及服务器代码示例下载地址：http://dev.360.cn/wiki/index/id/73

线上 FAQ 地址：http://dev.360.cn/wiki/index/id/74

接入中遇到问题先用自检工具自检，自检通过后依然查不出问题可咨询相关人员。

开发者平台相关问题联系

开平-包打听：2724990365

接入相关问题联系

Sdk-包打听：2833717137

其他问题联系

百晓生：2069053690