360帐号\_数据\_推送SDK接入文档

360移动开放平台 2014年8月19日

目录

[1. 概述 3](#_Toc401158554)

[重要提醒 3](#_Toc401158555)

[2. 接入流程 4](#_Toc401158556)

[2.1申请APPID和APPKEY 4](#_Toc401158557)

[2.2导入资源包 4](#_Toc401158558)

[2.2.1 导入jar包 6](#_Toc401158559)

[2.2.2 添加assets目录下的文件 6](#_Toc401158560)

[2.3配置应用工程的AndroidManifest.xml 6](#_Toc401158561)

[2.3.1添加权限 6](#_Toc401158562)

[2.3.2添加activity 6](#_Toc401158563)

[2.3.3在AndroidManifest.xml中添加meta-data 8](#_Toc401158564)

[2.4编码接入 9](#_Toc401158565)

[2.5混淆编译 9](#_Toc401158566)

[2.6测试 9](#_Toc401158567)

[2.7提交 9](#_Toc401158568)

[2.8审核 9](#_Toc401158569)

[2.9上线 10](#_Toc401158570)

[3. 编码接入详细介绍 11](#_Toc401158571)

[3.1 初始化接口【客户端调用】（必接） 11](#_Toc401158572)

[3.2 登录授权流程 11](#_Toc401158573)

[3.2.1流程介绍 11](#_Toc401158574)

[3.2.2接口介绍 11](#_Toc401158575)

[3.3 360SDK社交接口 17](#_Toc401158576)

[3.3.1接口说明 17](#_Toc401158577)

[3.3.2接口介绍 18](#_Toc401158578)

[3.4 销毁接口【客户端调用】(必接) 34](#_Toc401158579)

[3.5 360支付SDK其他接口 34](#_Toc401158580)

[3.5.1退出接口【客户端调用】（必接） 34](#_Toc401158581)

[3.5.2打开论坛接口【客户端调用】（可选） 35](#_Toc401158582)

[3.5.3防沉迷查询接口【客户端调用】（必接） 36](#_Toc401158583)

[3.5.4实名注册接口【客户端调用】（必接） 38](#_Toc401158584)

[3.5.5使用Matrix的get方法，获取基本信息 39](#_Toc401158585)

[4. 数据统计API说明 41](#_Toc401158586)

[4.1 关卡统计API 41](#_Toc401158587)

[4.2 任务统计API 41](#_Toc401158588)

[4.3 支付统计API 42](#_Toc401158589)

[4.4 虚拟币购买物品统计API 43](#_Toc401158590)

[4.5 物品消耗统计API 46](#_Toc401158591)

[4.6 玩家统计API 47](#_Toc401158592)

[4.7 角色统计API 48](#_Toc401158593)

[4.8 自定义事件统计API 48](#_Toc401158594)

[4.9 获取用户在线配置参数API 49](#_Toc401158595)

[5. 推送API说明 50](#_Toc401158596)

[5.1 设置标签API 50](#_Toc401158597)

[5.2 设置别名API 50](#_Toc401158598)

[5.3 设置标签和别名API 50](#_Toc401158599)

[5.4 获取应用本机推送标识号API 50](#_Toc401158600)

[5.5 获取应用标识号API 51](#_Toc401158601)

[6.附录： 52](#_Toc401158602)

[6.1 Demo工程简介 52](#_Toc401158603)

# 概述

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

本文档面向的读者是安卓应用开发者。

## 重要提醒

必须把APPID、APPKEY填写在AndroidManifest文件中，不能使用@string引用。

新游测试的privatekey不需要你们计算，传空就好。Sercetkey新游测试结束后会给出。详情请参照dev.360.cn

具体参数获取和后续流程请参照faq：<http://dev.360.cn/wiki/index/id/96>

# 接入流程



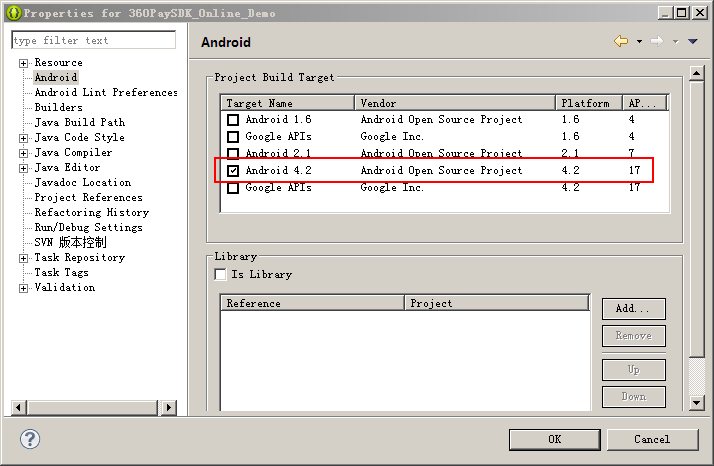
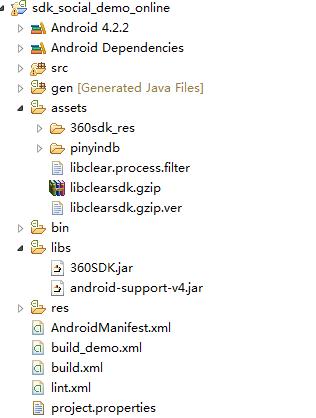
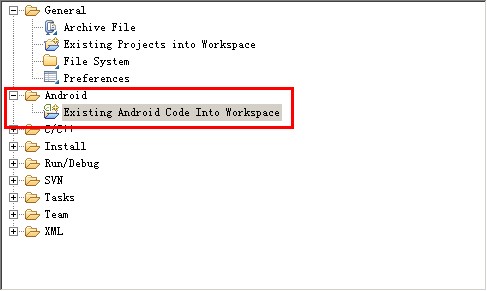
## 2.1申请APPID和APPKEY

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

申请指南：<http://aw9rrsljhj.l7.yunpan.cn/lk/QztXxKvmXwhkZ>

## 2.2导入资源包

网游SDK目前支持Android2.2及以上的系统版本，为兼容Android 4.0及以上的新手机，编译时请使用Android4.2 或以上的版本，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）：



图：开发环境配置

### 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。

### 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

<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.READ\_SMS"/>

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

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

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

<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS" />

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

### 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="orientation|keyboardHidden|navigation"

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

android:exported="true" >

</activity>

<!-- 360SDK浮窗 -->

<!-- 个人中心activity -->

<activity

android:name="com.qihoo.gamecenter.sdk.suspend.personal.PersonalActivity"

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 -->

<!-- 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” -->

<activity

android:name=".wxapi.WXEntryActivity"

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

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

android:exported="true" />

### 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>

**<!-- 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>

## 2.4编码接入

详见第三章节编码接入详细介绍

## 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 com.qihoo.\*\* { \*; }

-dontwarn org.apache.http.conn.ssl.SSLSocketFactory

## 2.6测试

1.使用自测工具检查接入完整性

下载地址：包内自带自检工具，请不要用线上下载的 APK自检工具，使用方法，请参照自检工具包内的read me。

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

应用提交完成后即可提交审核，提交后需等待360平台的审批，审批结果以邮件形式进行反馈。

未通过审核的应用我们会通过平台和邮件的方式告知您未通过的原因和改进建议。

客服电话：010-58781044

客服邮箱：[360box@360.cn](mailto: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。

# 编码接入详细介绍

## 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

**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，否则无法使用360SDK。

**参数：**

activity activity对象

**使用例子：**

Matrix.init(this);

## 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. SDK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. SDK把登录结果返给应用客户端，登录完成。

**注意事项**

1. SDK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。

### 3.2.2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());

//-- 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: {

access\_token: "11814024959cb47eef1734b15bc94c79984493a922b7c5c088" // token

expires\_in: 35963,

scope: "",

refresh\_token: "",

user\_login\_res: {

errmsg: "ok",

data: {

nick: "adfgdffcfds", // 当前用户360账号昵称，如果没有昵该字段值为”我”

accessinfo: {

user\_me: {

avatar: "http://quc.qhimg.com/...3f72.jpg", // 用户头像

name: "wt\_pt" // 用户名

}

},

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/1/25qd123.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: "adfgdffcfds", //当前用户360账号昵称

app\_inviters: 0 // 被多少人邀请的计数

time: 1405329959,

errno: 0

}

},

errno: 0 // 0，登录成功，-1登录取消

}

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;

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

// 解析User info

QihooUserInfo info = parseUserInfoFromLoginResult(data);

// 解析access\_token

mAccessToken = parseAccessTokenFromLoginResult(data);

if (!TextUtils.isEmpty(mAccessToken)) {

// 登录结果直接返回的userinfo中没有qid，

// 需要去应用的服务器获取用access\_token获取一下带qid的用户信息

getUserInfo(info);

} 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=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": ""

}

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.

## 3.4 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，以释放资源。

**参数：**

context上下文

**使用例子：**

@Override

protected void onDestroy() {

super.onDestroy();

Matrix.destroy(this);

}

## 3.5 360SDK其他接口

### 3.5.1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

}

};

### 3.5.2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);

}

### 3.5.3防沉迷查询接口【客户端调用】（必接）

**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();

}

}

}

### 3.5.4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)

# 4. 数据统计API说明

游戏只调用初始化接口，便可享受360数据大部分服务，如无特殊需求，下文可忽略

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)

# 推送API说明

游戏只调用初始化接口，便可享受360push大部分服务，如无特殊需求，下文可忽略

## 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

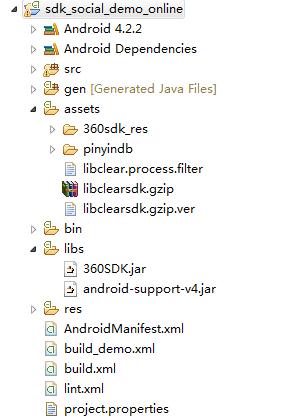
// 返回：应用标识号

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

# 6.附录：

## 6.1 Demo工程简介

360SDK工程结构如图所示：



图社交360SDK的Demo工程

Demo工程主要的包和类介绍：

om.qihoo.gamecenter.sdk.demos.activity包下的类用于演示SDK各个接口的调用，游戏接入SDK联调时，请着重参考SdkUserBaseActivity类中的代码实现 。以下为各个类的说明：

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.demos.utils:

QihooUserInfo，360用户信息数据。

Utils.java, 一些工具函数。