

# Android-SDK接入文档核心篇

## 版本记录

修改时间	修改内容
见《版本记录》文件	见《版本记录》文件

## 一、开发环境

1. 开发工具为Android Studio
2. gradle.properties配置AndroidX

```
android.useAndroidX=true
android.enableJetifier=true
```

3. minSdkVersion 版本最低为21，targetSdkVersion根据上架商店的要求设置

## 二、工程配置

### 1. 拷贝资源文件

将 AndroidStudio\SDK Resources\libs\quickgame\_hw\_x.x.x.aar 拷贝到Android项目libs目录，x.x.x为具体版本

### 2. 修改build.gradle配置，可参考demo

```
android{
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

repositories {
    flatDir {
        dirs 'libs' // aar文件目录
    }
}

dependencies {
    //必需依赖的库
    implementation(name: 'quickgame_hw_x.x.x', ext: 'aar') //x.x.x为具体版本
    implementation 'androidx.appcompat:appcompat:1.6.0'
    implementation "org.jetbrains.kotlin:kotlin-stdlib:1.9.20"
    implementation 'com.android.billingclient:billing:7.0.0'//google内购

    //根据需求添加的依赖库
    implementation 'com.google.android.gms:play-services-ads-identifier:17.0.1'//获取adid
    //下面为谷歌登录依赖
```

```
implementation 'com.google.android.gms:play-services-auth:21.3.0'
}
```

### 3. 修改AndroidManifest文件（按需配置）

在application节点下配置

```
<!--Google登录配置，没有Google登录可以不加，value值直接写在这里，不能用@string引用-->
<meta-data
    android:name="google-signin-client_id"
    android:value="替换为google后台申请的clientid"/>
```

**注：**Google后台有android client 和 web client 两种，这里使用的是web client id，要和运营人员确认清楚

## 三、接口调用（1、2、3、4、6必接，其余根据需要选接）

### 1. 获取SDK实例

接口没有说明调用类或调用实例时，默认使用该实例

```
QuickGameManager sdkInstance = QuickGameManager.getInstance();
```

### 2. 添加生命周期接口

在主Activity的生命周期调用SDK实例的同名方法

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    sdkInstance.onCreate(this);
}

@Override
protected void onStart() {
    super.onStart();
    sdkInstance.onStart(this);
}

@Override
protected void onResume() {
    super.onResume();
    sdkInstance.onResume(this);
}

@Override
protected void onPause() {
    super.onPause();
    sdkInstance.onPause(this);
}

@Override
protected void onStop() {
```

```

super.onStop();
sdkInstance.onStop(this);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    sdkInstance.onDestroy(this);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    sdkInstance.onActivityResult(requestCode, resultCode, data);
}

```

### 3. 初始化接口

调用方法：

```

public void init(Activity activity, String productCode, QuickGameManager.SDKCallback
sdkCallback)

```

`productCode` 是在Quick后台创建的应用的ProductCode

`sdkCallback` 是SDK通知游戏的回调实例。包括初始化成功、登录完成、退出登录、google订阅类商品回调这四个事件。

示例：

```

sdkInstance.init(this, "quick后台创建产品的ProdectCode", new QuickGameManager.SDKCallback() {
    @Override
    public void onInitFinished(boolean isSuccess, String error) {
        if (isSuccess) {
            //初始化成功
        } else {
            Log.d(TAG, "init failed"+error);
        }
    }
}

@Override
public void onLoginFinished(QGUserData userInfo, QGUserHolder loginState) {
    //登录成功
    if (loginState.getStateCode() == QGUserHolder.LOGIN_SUCCESS) {
        String uid = userInfo.getUid(); //sdk用户唯一标识
        String token = userInfo.getToken();
        boolean isGuest = userInfo.isGuest(); //判断是否为游客登录
        String logintype = userInfo.getOpenType();
        //登录方式 6:Facebook,8:Google,10:Twitter,11:Line,1:Guest,13: Email
    } else if (loginState.getStateCode() == QGUserHolder.LOGIN_CANCEL) {
        Log.d(TAG, "login cancel");
    } else if (loginState.getStateCode() == QGUserHolder.LOGIN_FAILED) {
        Log.d(TAG, "login failed");
    }
}

```

```

    }
}

@Override
public void onLogout() {
    //注销账号成功，注销账号成功后，游戏应从游戏的界面退回到登录界面中
}

@Override
public void onGooglePlaySub(String productId, String sdkOrder, boolean isAutoRenewing,
boolean isAcknowledged) {
    //游戏内订阅类商品的回调
}
});

```

## 4. 登录接口

请确保在初始化回调成功之后 再调用登录接口

调用方法：

```

//显示登录界面
public void login(Activity activity)

//如果游戏有登录按钮 或者 只需要某一种登录方式，可以调用下面的方法
//游客登录
public void freeLogin(Activity activity)

//谷歌登录
public void googleLogin(Activity activity)

//Facebook登录
public void facebookLogin(Activity activity)

//PlayGame登录
public void playGameLogin(Activity activity)

//Line登录
public void lineLogin(Activity activity)

//邮箱登录
public void emailLogin(String email,String password)

//注册邮箱账号
public void registerEmailAccount(String email,String password,String verificationCode,
FailedListener listener)

//邮箱账号重置密码
public void resetPassword(String email,String password, String verificationCode,
FailedListener listener)

//注册账号发送验证码
public void sendEmailCodeForRegister(String email, SendEmailCodeListener callBack)

```

```
//重置密码发送验证码
public void sendEmailCodeForFindPassword(String email, SendEmailCodeListener callBack)
```

示例：

```
sdkInstance.login(this);
```

## 5. 注销接口

调用方法：

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

示例：

```
sdkInstance.logout(this);
```

## 6. 支付接口

调用方法：

```
public void pay(Activity activity, QGOrderInfo orderInfo, QGRoleInfo roleInfo,
                QGPaymentCallback paymentCallback);
```

`orderInfo` 是订单信息

`roleInfo` 是游戏内该玩家的角色信息

`paymentCallback` 支付结果的通知回调，包括支付成功、支付失败、支付取消三种事件

**注：**使用三方支付时，sdk无法确定支付结果，可能不会通知paymentCallback

### QGOrderInfo 方法说明

方法	必填	说明
setProductOrderId	是	游戏生成的订单唯一标识
setGoodsId	是	sdk后台配置的商品ID
setOrderSubject	是	商品名称
setAmount	是	商品金额；该参数只用于三方统计，不用于支付
setSuggestCurrency	是	金额货币单位代码，需和金额对应 <a href="https://en.wikipedia.org/wiki/ISO_4217#Active_codes">https://en.wikipedia.org/wiki/ISO_4217#Active_codes</a>
setExtrasParams	否	透传参数，请勿传特殊符号，如无法避免建议进行base64编码
setSkuType	否	商品类型，默认(inapp)是消耗性，订阅请设置为subs

方法	必填	说明
setCallbackURL	否	支付回调地址，可以配置在SDK后台 同时配置时，优先读取后台配置的回调地址

**QGRoleInfo 方法说明**

方法	必填	说明
setRoleId	是	游戏内角色ID
setRoleName	是	游戏内角色的名称
setRoleLevel	是	游戏内角色的等级
setServerId	是	游戏内角色的区服ID
setServerName	是	游戏内角色区服名称

示例：

```
QGOrderInfo orderInfo = new QGOrderInfo();
orderInfo.setProductOrderId("xxxxxxxxxx");
orderInfo.setOrderSubject("60钻石");
orderInfo.setGoodsId("10001");
orderInfo.setAmount(0.99); // double类型
orderInfo.setSuggestCurrency("USD");

QGRoleInfo roleInfo = new QGRoleInfo();
roleInfo.setRoleId("Role ID");
roleInfo.setServerId("Server ID");
roleInfo.setRoleName("RoleName");
roleInfo.setRoleLevel("1");
roleInfo.setServerName("S1");

sdkInstance.pay(this, orderInfo, roleInfo, new QuickGameManager.QGPaymentCallback() {
    @Override
    public void onPaySuccess(String gameId, String sdkOrderId, String goodsId, String
extrasParams) {
        //返回支付时传入的游戏订单号、SDK订单号、商品ID和透传参数
    }

    @Override
    public void onPayFailed(String gameId, String sdkOrderId,String errorMessage) {}

    @Override
    public void onPayCancel(String gameId, String sdkOrderId,String errorMessage) {}
});
```

## 7. 上传角色信息接口

上报角色信息到SDK后台

调用方法：

```
public void submitRoleInfo(QGRoleInfo roleInfo)
```

`roleInfo` 角色信息

示例：

```
QGRoleInfo roleInfo = new QGRoleInfo();
roleInfo.setRoleId("角色ID");
roleInfo.setServerId("区服ID");
roleInfo.setRoleName("角色名称");
roleInfo.setRoleLevel("角色等级");
roleInfo.setServerName("区服名称");
roleInfo.setVipLevel("vip等级");
sdkInstance.submitRoleInfo(roleInfo);
```

## 8. 账号绑定接口

若未绑定指定平台的用户，调用会走绑定流程；若已经绑定用户，调用该方法则为解绑账。接收绑定结果需要设置绑定回调 `setUserBindCallback`，具体见下方示例代码

**注：** 邮箱绑定后客户端无法进行解绑，只能在SDK后台操作

调用方法：

```
public void bindUser(String loginOpenType)
```

`loginOpenType` 表示要绑定或解绑的平台，值如下表

loginOpenType	对应绑定的登录方式
QGConstant.LOGIN_OPEN_TYPE_GOOGLE	Google登录
QGConstant.LOGIN_OPEN_TYPE_FACEBOOK	Fackbook登录
QGConstant.LOGIN_OPEN_TYPE_TWITTER	Twitter登录
QGConstant.LOGIN_OPEN_TYPE_LINE	Line登录
QGConstant.LOGIN_OPEN_TYPE_PLAYGAME	Play游戏登录
QGConstant.LOGIN_OPEN_TYPE_EMAIL	Email登录

示例：

```
QuickGameManager.getInstance().setUserBindCallback(new QuickGameManager.QGUserBindCallback()
{
    @Override
```

```

    public void onBindInfoChanged(String uid, boolean isBindUnBindSuccess, QGUserBindInfo
qgUserBindInfo) {
        //isBindUnBindSuccess : 绑定或者解绑是否成功
        Log.d(TAG, "isBindUnBindSuccess "+isBindUnBindSuccess);
        Log.d(TAG, "isBindGoogle "+qgUserBindInfo.isBindGoogle());
        Log.d(TAG, "isBindFacebook "+qgUserBindInfo.isBindFacebook());
    }

    @Override
    public void onexitUserCenter() {
        Log.e("GameActivity", "退出用户中心");
    }
});
QuickGameManager.getInstance().bindUser(QGConstant.LOGIN_OPEN_TYPE_FACEBOOK);

```

## 9. 日志开关接口

在测试阶段可以开启，以便排查问题，默认为false，正式出包上线之前请注释或设置成false。

调用方法：

```
QGLog.setDebugMod(boolean isShow)
```

`isShow` true为显示日志，false为不显示

## 10. 获取第三方登录绑定情况

登录成功后调用，获取第三方登录绑定情况。

调用方法：

```
public QGUserBindInfo getUserBindInfo()
```

`QGUserBindInfo` 为绑定第三方登录的信息，为空（null）时表示没有绑定第三方登录。

示例：

```

QGUserBindInfo bindInfo = sdkInstance.getUserBindInfo();
if(bindInfo.isBindGoogle()){
    String name = bindInfo.getGoogleAccountName();
}
if(bindInfo.isBindFacebook()){
    String name = bindInfo.getFbAccountName();
}

```

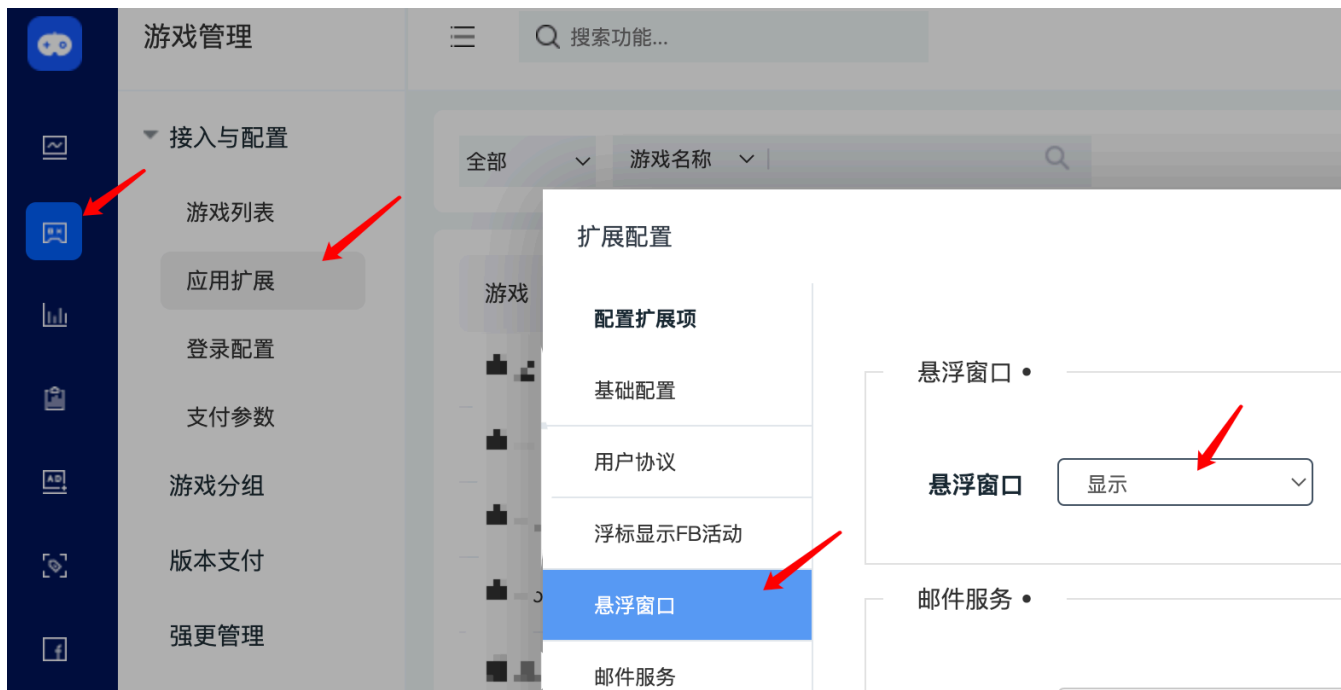
## 11. 显示悬浮按钮

如需要显示悬浮窗可以在登录状态下调用，同时 sdk 后台要打开截图的配置

调用方法：

```
public void showFloatView(Activity activity)
```





## 12. 关闭悬浮按钮

调用方法：

```
public void closeFloatView(Activity activity)
```

## 13. 直接谷歌登录

```
//谷歌登录  
public void googleLogin(Activity activity)
```

## 14. 获取用户信息

调用方法：

```
public QGUserData getUser()
```

## 15. 进入个人中心

```
public void enterUserCenter(Activity activity)
```

## 16. 获取设备唯一标识

```
public String getDeviceId(Activity activity)
```

## 17. 获取用户IP地区码和IP地址

初始化成功之后调用才有效

```
public Map<String,String> getCountryInfo()
```

示例：

```
String ip = QuickGameManager.getInstance().getCountryInfo().get(CountryInfo.IP);
String ipCountryCode =
QuickGameManager.getInstance().getCountryInfo().get(CountryInfo.IP_COUNTRY_CODE);

//如： 香港，使用的网络IP是171.212.1.2
//ip = 171.212.1.2, ipCountryCode = HK
```

## 18. 打开谷歌Play商店评价

在build.gradle添加依赖：

```
implementation 'com.google.android.play:review:2.0.1'
```

在需要的时候调用下面的方法：

```
void openReview(Activity activity, CallbackListener callbackListener)
```

示例：

```
QuickGameManager.getInstance().openReview(this, new CallbackListener() {
    @Override
    public void onFinish() {
        //评价界面关闭，不会返回评价是成功、失败还是取消
    }
});
```

## 19. 获取登录过程打点信息

```
public void setLoginEventTraceListener(LoginEventTraceListener listener)
```

示例

```
QuickGameManager.getInstance().setLoginEventTraceListener(new LoginEventTraceListener() {
    @Override
    public void onEvent(LoginEvent event, String message) {
        //event 事件名
        //message 描述
        Log.d("LoginEventTrace", event.name());
    }
});
```

## 20. 销毁账号接口

```
//销毁账号，SDK会显示警告提示
public void UserTrash(Activity activity)

//销毁账号，SDK不显示警告提示，游戏有警告提示时使用
public void UserTrashNoUI(Activity activity)
```

## 21. 网络连接状态

```
//监听网络状态
void setNetworkConnectListener(NetworkConnectListener listener)
```

示例

```
sdkInstance.setNetworkConnectListener(new NetworkConnectListener() {
    @Override
    public void onNetworkAvailable() {
        Log.d(TAG, "网络连接");
    }

    @Override
    public void onNetworkLost() {
        Log.d(TAG, "网络断开");
    }
});
```

## 22. 检查是否有上次登录的有效信息

如果用户登录了账号，并且没有退出登录，改接口会返回true，否则返回false

```
boolean checkLoginValid()
```

## 23. 隐藏个人中心退出登录按钮

```
void hideUserCenterLogout()
```

## 24. 隐藏自动登录窗口

```
void hideAutoLoginWaiting()
```

## 25. 查询商店商品信息

方法

```
void queryStoreSku(Activity activity, List<String> goodsIds, QueryStoreSkuListener listener)
```

示例

```

ArrayList<String> arrayList = new ArrayList<>();
arrayList.add("1111");
arrayList.add("2222");
arrayList.add("3333");
QuickGameManager.getInstance().queryStoreSku(this, arrayList, new QueryStoreSkuListener() {
    @Override
    public void onResult(List<StoreSku> skuDetailsList) {

        for(StoreSku sku:skuDetailsList){
            //货币单位加金额， 例如 US$1.03
            String price = sku.getPrice();
            //商品id
            String skuId = sku.getProductId();
            //货币单位代码， 例如 USD
            String currency = sku.getCurrency();
            //商品金额的1000000倍， 例如 1030162
            long amountMicros = sku.getMicrosPrice();
        }
    }
});

```

## 四、AndroidManifest更多设置

### 1. 渠道包标识

不同的包可以配置不同的标识，在sdk后台数据的渠道栏显示；如果不配置会读取手机系统设置的语言地区为渠道标识，如：CN、US

```

<!--和运营确认渠道标识，替换value值，需要为字符串-->
<meta-data
    android:name="channelId"
    android:value="replace with channel name" />

```

如果用云端分包或分包工具分包，不需要在AndroidManifest添加需要channelId

在Android项目assets目录下创建 quickgame\_sdk/channel\_id.txt 文件，文件内容为default

## 五、其他（按需接入）

### 1. 移除Http允许明文传输配置

sdk默认支持http明文传输，如果需要移除http明文传输支持

在主module的 res/xml 目录下创建 network\_security\_config.xml 文件，内容如下

（文件名必须为 network\_security\_config.xml）

```

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="false" />
</network-security-config>

```

## 2. 用户协议和隐私政策配置

用户协议和隐私政策默认显示在邮箱账户注册界面（显示内容在quick sdk后台**游戏管理**-->**应用扩展配置**）

如果想显示在首次启动或者登录界面，可以在初始化接口前设置SDKConfig，如下：

```
SDKConfig sdkConfig = new SDKConfig.Builder()
    //首次启动显示用户协议弹框，不同意无法进入登录界面
    .showServicesAndPrivacyPolicy()
    //在登录界面显示用户协议
    .showLoginServicesAndPrivacyPolicy()
    .build();
QuickGameManager.getInstance().setSdkConfig(sdkConfig);
```

## 3. 获取谷歌支付的Purchase对象

如果需要获取谷歌支付成功后的Purchase对象，可以在初始化之后调用下面的方法设置回调，谷歌支付成功后会通过这个方法返回Purchase对象

```
public void setGooglePurchaseCallback(GooglePurchaseCallback callback)
```

示例

```
sdkInstance.setGooglePurchaseCallback(new GooglePurchaseCallback() {
    @Override
    public void onPaySuccessWithPurchase(String productOrderId, String orderNo,boolean
issandbox, Purchase purchase) {
        // productOrderId 游戏订单号
        // orderNo![rsa_key](D:\01_quick_doc\02_Doc_File\内部资源\md文档_android
studio\img\rsa_key.png) sdk订单号
        // issandbox 是不是谷歌测试订单，为true表示是测试订单，配置产品为单机游戏时无效
        // purchase 谷歌支付返回的对象
    }
});
```

## 4. 欧盟DMA隐私配置弹框

在AndroidManifest配置DMAEnable为true时，如果初始化请求的ip地址属于欧盟地区会显示隐私配置弹框

```
<meta-data
    android:name="DMAEnable"
    android:value="true" />
```

## 5. 单机游戏配置

没有服务端的单机游戏，按照下方说明配置

## 5.1 sdk后台设置为单机游戏

sdk后台游戏管理 --》应用拓展 --》基础配置 --》单机游戏 开启

## 5.2 sdk后台配置 RSA Public key

sdk后台谷歌商店 tab --》服务参数 --》支付参数 --》RSA Public key

RSA Public Key从Google Play后台下方位置获取



## 5.3 游戏同步订单已发货

单机游戏根据支付成功回调发货，为了防止掉单，游戏在发货完成后，需要调用该方法通知SDK该笔订单已完成，后续sdk不再通知游戏，否则SDK还会再通知游戏

```
public void singleGameOrderFinish(String sdkOrderId)
```

**sdkOrderId**: 支付成功接口返回的sdk订单号

示例

```
QuickGameManager.getInstance().singleGameOrderFinish("xxxxxxx");
```

## 6. 调用手机邮件App发送邮件

```
//一个收件人
void sendEmail(Activity activity,String email, String subject, String content)

//多个收件人
void sendEmail(Activity activity,String[] emails, String subject, String content)
```

**email** 收件人

**subject** 邮件标题

**content** 邮件内容