

DGames_SDK 接入文档

目录

1.IDE 配置

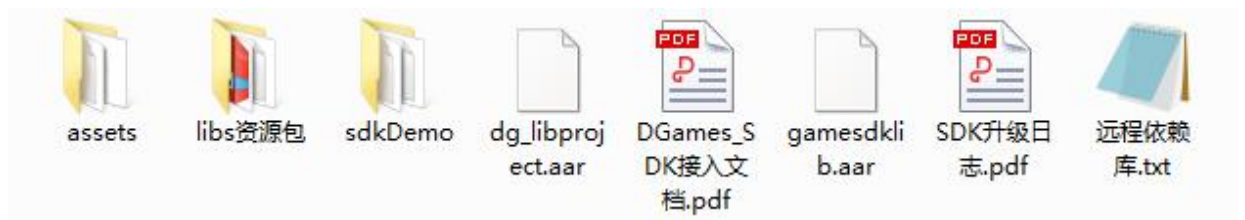
1.1 导入 assets 包.....	1
1.2 导入 aar 包.....	1
1.3 导入远程依赖库.....	1
1.4 so 文件适配.....	2
1.5 重新编译项目.....	3

2.客户端配置.....3

2.1 权限及初始化 sdk.....	3
2.2 添加生命周期回调.....	5
2.3 sdk 提供的 api.....	6

1.IDE 配置：

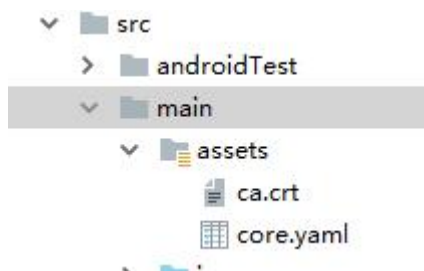
解压我方提供的接入文档压缩包后，可以在根目录下看到如下的 **assets** 证书文件夹、**libs** 资源包文件、**sdkDemo**、**aar** 包、**DGames_SDK** 接入文档、远程依赖库文件和 **sdk** 升级日志：



步骤：

1.1 导入 **assets** 文件夹

将 **assets** 文件夹导入到 **Android Studio** 工程中，位置如下图：



1.2 导入 **aar** 包

Android Studio 工程中将两个 **aar** 包直接放入 **libs** 包下，在项目的 **build gradle** 中添加

```
Repositories{  
  
    flatDir{  
  
dirs 'libs'}}
```

添加两行(如果您的 **Android studio** 是 3.0 之前的，请将以下的 **implementation** 修改成 **compile**)

```
Implementation(name: 'dg_libproject', ext: 'aar')  
Implementation(name: 'gamesdklib', ext: 'aar')
```

1.3 导入远程依赖库（注意：或者直接复制 **libs** 文件到项目相应的位置，（远程依赖库和 **libs** 资源包）只能选择其中一个，下面用的是远程依赖库）

从远程依赖库文件中复制并导入远程依赖库：

如下图：

```
dependencies{

    implementation fileTree(include:['*.jar'],dir:'libs')
    implementation 'com.android.support.appcompat-v7:27.1.1'
    implementation(name: 'dg_libproject', ext: 'aar')
    implementation(name: 'gamesdklib', ext: 'aar')
    implementation('org.web3j:core:3.3.1-android')
    implementation 'com.journeyapps:zxing-android-embedded:3.3.0@aar'
    implementation 'com.google.zxing:core:3.3.0'
    implementation 'com.squareup.okhttp3:okhttp:3.10.0'
    implementation 'com.android.support.multidex:1.0.3'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'

}
```

1.4 so 文件适配

在 app 下的 build.gradle 中添加如下代码以适配机型：

```
ndk{

    abiFilters "armeabi-v7a", "x86", "armeabi"

}
```

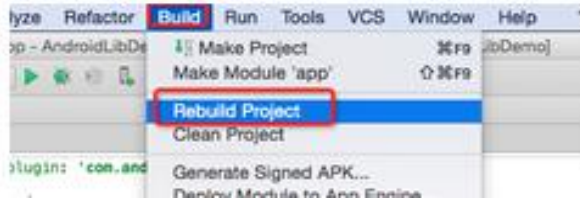
如下图：

```
android{
    compileSdkVersion 26
    defaultConfig{
        applicationId "com.dgames.sdkdemo"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

        ndk {
            abiFilters "armeabi-v7a", "x86", "armeabi"
        }
    }
}
```

1.5 重新编译项目:

从工具栏依次选择: "Build"-->"rebuild project"



温馨提示:因 gradle 的版本不同可能导致配置有些微小区别,可百度参考相关的网上配置文档引入 aar 包到工程。

注意:如果文件配置完成,项目报错:引用的 Dex 文件数超过方法数限制 65536,请参考网址:<https://www.aliyun.com/jiaocheng/804289.html>

以上 IDE 配置完毕,开始客户端配置

2.客户端配置: 当前版本: v1.0.0

2.1: 权限及初始化 sdk (必接)

```
*Config.LOGIN.....已登录
*Config.UNLOGIN.....未登录
*Config.LOGINOUT.....已退出
*config.setLandscap.....横竖屏设置 true 为横屏 false 为竖屏
*config.setAppId.....cp 传入的 appId
*config.setApibrowse.....cp 传入的正式浏览器地址
*config.setDebugApibrowse.....cp 传入的测试浏览器地址
*config.setFloatGravity.....悬浮球的初始化显示位置
*config.setDecimals.....精确度
*config.setDgameDebug....判断子链币和 dgame 充值地址和 ETH 的 url 地址以及浏览器地址
                           true 为测试地址 false 为正式地址
*config.setLanguage.....en 英文 cn 中文
*@param amount.....支付金额
*@param orderId.....订单号
*@param toAddress.....自定义支付地址
*@param comment+System.currentTimeMillis //支付信息 (特别注意: 为了避免重复
                                           交易, 支付信息后面必须加上时间戳)
*@param tokenId.....事务标识 (物品的唯一 ID)
*@param equip_info..游戏设备详情(包括 id, 名字, 价格, 图片, 备注等以 json 形式拼接)
*@param comment_erc..游戏 appId, 服务器分类 id, 游戏设备 id (以逗号连接的字符串)
PermissionUtils //权限工具类 (温馨提示: 先实例化一个工具类对象)
setPermission(); //权限接入方法
getConfig(); //得到 config 对象, 供 sdk 初始化调用
QueryGameAmount(this, new ICallback); //当前用户子链币余额查询方法 (参考 SDKDEMO)
QueryDgasAmount(new ICallback); //当前用户 DGAS 余额查询方法 (请参阅 SDKDEMO)
transErc(MainActivity.this, orderId, tokenId, equip_info, comment) //ERC_721 转账
transErcAddress(MainActivity.this, orderId, toAddress, tokenId, equip_info, comment)
//ERC_721 输入转账地址转账
gameQueryAssetErc(tokenId) //ERC_721 查询资产归属功能
subStringRecharge(); //直接调起 dgas 或 dgame 充值页面方法
```

在主 Activity 中调用:

```
PermissionUtils permissionUtils=new PermissionUtils();
Public void getConfig(){
    config=new Config();
```

```

config.setAppId( "V43lrSWOMpq3xGGJYSQTGH5ox1MBiXjJRw" );
config.setApibrowse( "http://192.168.2.32:801" );
config.setDebugApibrowse( "https://queryfb.dgame.org.4490" );
config.setFloatGravity(FloatGravity.TOP_CENTER);
config.setDecimals( "100000000" );
config.setLanguage( "cn" );
config.setDgameDebug(true);
config.setLandscape(true);
}
perms = new String[] {
    Manifest.permission.CAMERA,
    Manifest.permission.WRITE_EXTERNAL_STORAGE,
    Manifest.permission.READ_PHONE_STATE,
    Manifest.permission.ACCESS_FINE_LOCATION
};
private void setPermission() {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {

        permissionUtils.requestRunPermission(perms, new PermissionListener()) {
            @Override
            public void onGranted() {
                DGameManager.init(this, config);
            }

            @Override
            public void onDenied(List<String> deniedPermission) {
                for (int i = 0; i < deniedPermission.size(); i++) {
                    showRequestRunPermission = ActivityCompat.shouldShowRequestRunPermission
Rationale(this, .get(i));
                } if (showRequestRunPermission) {
                    setPermission();
                } else {
                    showMissingPermissionDialog();
                }
            }
        });
    } else {
        DGameManager.init(this, config);
    }
}

//Permissions callback method
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    permissionUtils.onRequestPermissionsResult(requestCode, permissions,
grantResults);
}

```

```

Private void showMissingPermissionDialog() {

AlterDialog.Builder builder=new AlterDialog.Builder(this);

    builder.setTitle(“prompt”);

    builder.setMessage(“For your normal use of SDK, please open the permissions!”);

    builder.setPositiveButton(“go Set!”,new DialogInterface.OnClickListener() {
Public void onClick(DialogInterface dialog,int which) {
        startAppSettings();
    }
});
}

    Builder.setCancelable(false);
    Builder.show();
}

Private void startAppSettings() {
    Intent intent=new Intent(Setting.ACTION_APPLICATION_DETAILS_SETTINGS);
    intent.setData(Uri.parse(“package:”+getPackageName()));
    startActivityResult(intent,REQUEST_CODE_SDK_RESULT_PERMISSIONS);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==REQUEST_CODE_SDK_RESULT_PERMISSIONS) {
        setPermission();
    }
}

```

2.2: 添加生命周期回调(必接)

在主 Activity 的 `onStart` 中添加

```
DGameManager.onStart();
```

在主 Activity 的 `onResume` 中添加

```
DGameManager.onResume();
```

在主 Activity 的 `onPause` 中添加

```
DGameManager.hideFloatintView();
```

在主 Activity 的 `onStop` 中添加

```
DGameManager.onStop();
```

在主 Activity 的 `onDestroy` 中添加

```
DGameManager.onDestroy();
```

2.3: sdk 提供的 api

如下表，以下的调用方式均为：**DGameManager.函数名(...)**，具体使用方法请参考提供的 sdkDemo，

函数名	作用	参数&类型	描述
Login(context)	登录	Context 上下文	直接调用即可
setSDKLoginCallback(new ILoginCallback())	登录回调 (请参考 demo)	ILoginCallback 回调接口	OnSuccess (message) 回调成功 Cp 端获得 signdata 登录验证参数 uname 玩家昵称 address 设备地址 OnFailed (message) 回调失败
Pay(context,orderId, amount, comment)	支付	Context 上下文 orderId 支付订单 Amount 支付金额 Comment 支付信息	直接调用即可
setSDKPayCallback(new IPayCallback)	支付回调	IPayCallback 回调接口	OnPaySuccess(json Str) 回调成功 cp 端获得： payOrderId 支付订单号 txid 交易流水号 onPayFail(message)
PayAddress (context,orderId,toAddress,amount, comment)	自定义支付地址	Context 上下文 orderId 支付订单 toAddress 支付地址 Amount 支付金额 Comment 支付信息	直接调用即可
openUserCenter()	打开个人中心	无	直接调用即可
isLogin()	判断是否登录	无	直接调用即可
queryGameAmount(context, new ICallback)	查询子链币余额	Context 上下文	直接调用返回当前用户子链币余额

queryDgasAmount(new lcallback)	查询 Dgas 余额	无	直接调用返回当前用户 dgas 余额
transErc(context,orderId,tokenId,equip_info,comment)	ERC_721 资产转账	Context 上下文 orderId 支付订单 tokenId 物品的唯一 ID equip_info 游戏装备信息 Comment 转账信息	直接调用即可
transErcAddress(context,orderId,toAddresses,tokenId,equip_info,comment)	ERC_721 资产自定义地址转账	Context 上下文 orderId 支付订单 toAddresses 转账地址 tokenId 物品的唯一 ID equip_info 游戏装备信息 Comment 转账信息	直接调用即可
setSDKTransErcCallback(new ITransErcCallback)	ERC_721 转账回调	无	onTransErcSuccess(str) 提交事务成功 onTransERCFail(str) 提交事务失败
gameQueryAssetErc(toKenId)	ERC_721 查询资产归属	tokenId 物品的唯一 ID	直接调用返回资产归属的用户地址
rechargeSubChain(context)	Dgas, dgame 充值子链币	Context 上下文	直接调用即可

注意：关于支付回调请参考服务器支付回调接口文档。