

会玩聚合接入文档

Android 客户端

版本号	内容	时间
3.0.0	1、更新、优化聚合逻辑	2019-10-15

1 说明

本文档为聚合 Android 客户端接入文档。

本文档与 Demo 描述应一致，如出现文档描述与提供的 Demo 代码不一致(不包括某些说明有多种实现方法，如有多种使用方法，会在文档中指出，Demo 只是实现其中一种)，请与我们技术联系。

提供的 Demo 为 H5 游戏接入 Demo。Cocos2D、Unity3D 等游戏接入时，亦可以 Demo 内的接入逻辑作为参考来接入。实现逻辑基本相同。

2 接入

2.1 应用配置

Android Studio 项目 build.gradle 文件配置建议：

- 1、SDK 最小支持版本为 19
- 2、在 defaultConfig 根下配置 multiDexEnabled 值为 true
- 3、配置 NDK 的支持配置

图示：

```
defaultConfig {  
    applicationId "com.player.game.union.h5.demo"  
    minSdkVersion 19  
    targetSdkVersion 29  
    versionCode 1  
    versionName "1.0"  
    multiDexEnabled true  
  
    ndk {  
        //设置支持的SO库架构  
        abiFilters 'armeabi-v7a', 'x86', 'armeabi', 'x86_64', 'arm64-v8a'  
    }  
}
```

2.2 资源导入

需要将我方的 assets、libs、res 等资源文件复制到游戏工程内对应的文件夹内，其中，在复制 libs、res/values 文件夹内的资源时，请注意合并资源的问题。以防出现资源丢失或者冲突。

将资源文件/AndroidManifest.xml 内的相关权限和 application 节点下的相关配置合并到游戏的 AndroidManifest.xml 对应的位置。

2.3 权限说明

我方所要使用到的权限有以下权限：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.READ_PHONE_NUMBERS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"
/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
<uses-permission android:name="android.permission.READ_LOGS" />
```

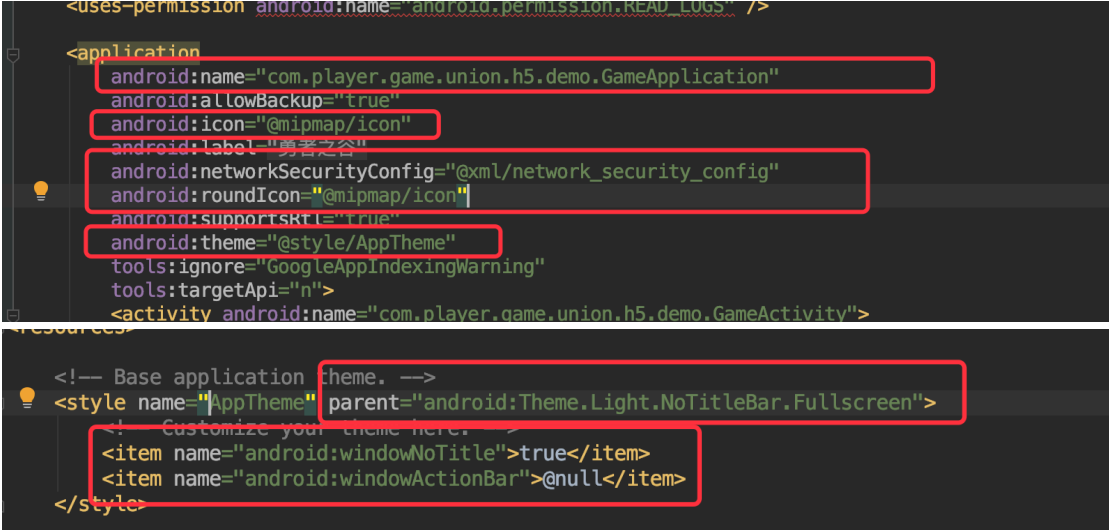
其中如果游戏适配版本为 Android6.0 及以上，需要动态申请权限。（请游戏自行做好适配方案。）我方 SDK 内部有针对我方所用的危险权限有进行权限申请处理。

2.4 AndroidManifest.xml 配置说明

游戏方在合并完资源后，需要对游戏 AndroidManifest.xml 进行一些配置。

- 1、需要在 application 节点内，增加
android:networkSecurityConfig="@xml/network_security_config" 属性配置。
- 2、需要在 application 节点内，将 android:icon 和 android:roundIcon 属性的值改为@mipmap/icon。
- 3、需要注意，application 节点内 android:theme 属性的值的配置，游戏方需要设置游戏在手机运行全屏没有 Titlebar 的属性。
- 4、application 节点内，android:name 属性的配置，请参照文档内 ‘Application 配置说明’ 来配置

图示：



```

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

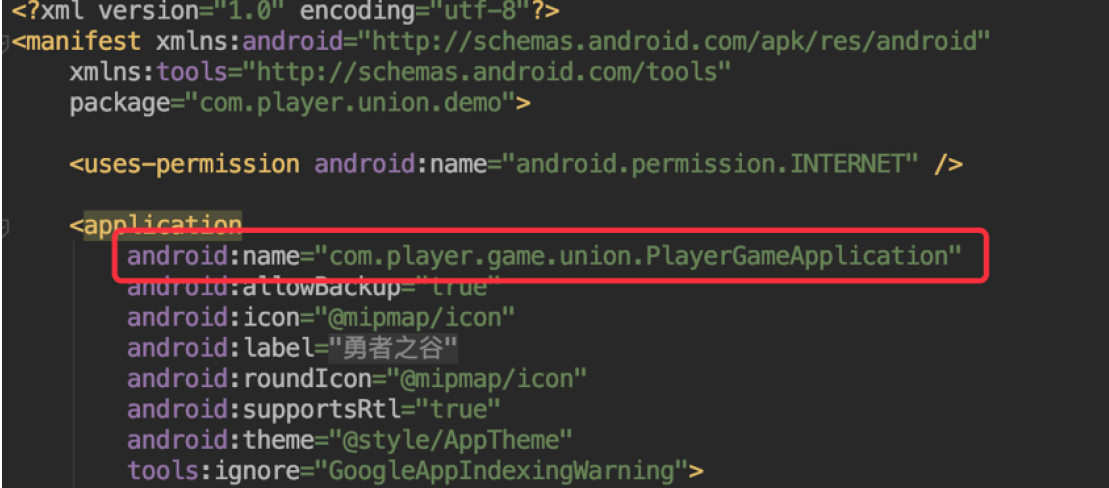
<application
    android:name="com.player.game.union.h5.demo.GameApplication"
    android:allowBackup="true"
    android:icon="@mipmap/icon"
    android:label="勇者之谷"
    android:networkSecurityConfig="@xml/network_security_config"
    android:roundIcon="@mipmap/icon"
    android:supportRtl="true"
    android:theme="@style/AppTheme"
    tools:ignore="GoogleAppIndexingWarning"
    tools:targetApi="n">
    <activity android:name="com.player.game.union.h5.demo.GameActivity">

resources
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:Theme.Light.NoTitleBar.Fullscreen">
        <!-- Customize your theme here. -->
        <item name="android:windowNoTitle">true</item>
        <item name="android:windowActionBar">@null</item>
    </style>

```

2.5 Application 配置说明

1、如果游戏没有自己的 Application 类，则直接将我方的 PlayerGameApplication 类配置到游戏应用的 AndroidManifest.xml 的 application 节点的 android:name 属性中。如图：



```

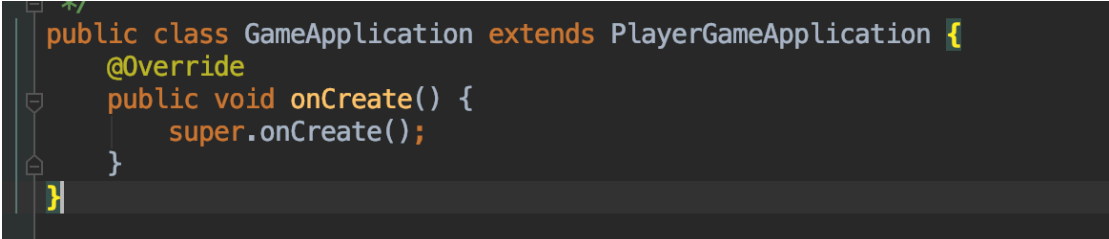
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.player.union.demo">

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

    <application
        android:name="com.player.game.union.PlayerGameApplication"
        android:allowBackup="true"
        android:icon="@mipmap/icon"
        android:label="勇者之谷"
        android:roundIcon="@mipmap/icon"
        android:supportRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="GoogleAppIndexingWarning">

```

2、如果游戏有自己的 Application 类，则需要将游戏的 Application 类继承我方的 PlayerGameApplication 类。如图：



```

public class GameApplication extends PlayerGameApplication {
    @Override
    public void onCreate() {
        super.onCreate();
    }
}

```

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

<application
    android:name="com.player.game.union.h5.demo.GameApplication"
    android:allowBackup="true"
    android:icon="@mipmap/icon"
    android:label="勇者之谷"
    android:networkSecurityConfig="@xml/network_security_config"
    android:roundIcon="@mipmap/icon"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    tools:ignore="GoogleAppIndexingWarning"
    tools:targetApi="n">
```

3、如果游戏有自己的 Application 类，并且已经继承了其他第三方的 Application 类，不方便继承我方的 PlayerGameApplication 类时，则需要重写 Application 的(onCreate、attachBaseContext、onConfigurationChanged)方法，并且接入我方的三个接口方法(onProxyCreate、onProxyAttachBaseContext、onProxyConfigurationChanged)，如图：

```
@Override
public void onCreate() {
    super.onCreate();
    PlayerGameApplication.onProxyCreate( application: this);
}

@Override
public void attachBaseContext(Context context) {
    super.attachBaseContext(context);
    PlayerGameApplication.onProxyAttachBaseContext( application: this, context);
}

@Override
public void onConfigurationChanged(Configuration configuration) {
    super.onConfigurationChanged(configuration);
    PlayerGameApplication.onProxyConfigurationChanged(configuration);
}
```

2.6 接口说明

我方 SDK，所有使用的调用方法，基本都封装在 PlayerGameUnionSdk 类内，如有其他调用，会单独说明。

2.6.1 初始化(必接)

调用接口：

```
PlayerGameUnionSdk.getInstance().initSdk(GameUnionInitParams
params , IGameUnionSdkCallback callback);
```

GameUnionInitParams 类说明：

描述：SDK 初始化参数

参数名	参数类型	参数说明
GameActivity	Activity	游戏主界面
Appld	String	我方提供的 Appld 参数

LoginKey	String	我方提供的登录 Key 参数
----------	--------	----------------

IGameUnionSdkCallback 类说明：

描述：SDK 回调接口

方法名	参数说明	操作
initSuccess()	SDK 初始化成功后回调接口	游戏方收到回调后，需要处理初始化成功之后的相关操作。 如果初始化不成功，会回调到 onErrorCallback 接口中
loginSuccess(SdkToken token)	SDK 渠道登录成功后回调接口	游戏方在此回调中，取返回的 SdkToken 的 userId 和 token_bg 值到游戏服务器，进行二次登录校验，校验成功后跳转到游戏选择服务器界面/其他界面。 如果登录失败，会回调到 onErrorCallback 接口中
submitGameInfoSuccess()	SDK 角色信息提交成功回调接口	游戏方可根据需求做处理。
logout()	SDK 账号注销成功回调接口	游戏方收到回调后，需要将游戏显示登录界面。并且调用 SDK 的登录接口。
payCallback(String sdkOrderId)	SDK 支付成功后回调接口	游戏方可根据需求做处理。 其中返回的 sdkOrderId 是 SDK 订单号
onExist(boolean isFinish)	SDK 退出游戏	游戏方在收到回调之后，需要先判断 isFinish 的值，如果

		为 True，则需要退出游戏 App，如果值为 False 则不需要进行任何处理
onErrorCallback(GameUnionCode code , String message)	SDK 错误回调接口	SDK 发生错误信息时候，会回调这个接口，游戏方可根据需求针对性处理每个错误回调

GameUnionCode 类说明

描述：SDK 错误码

参数名	参数说明
INIT_FAILED	初始化失败
LOGIN_FAILED	登录失败
SWITCH_LOGIN_FAILED	切换账号失败
LOGOUT_FAILED	注销账号失败
SUBMIT_GAME_ROLE_INFO_FAILED	角色信息上报失败
PAY_FAILED	支付失败
PERMISSION_DENIED	权限申请失败

调用示例:

```
GameUnionInitParams params = new GameUnionInitParams();
//游戏主页面
params.setGameActivity(GameActivity.this);
//我方提供的AppId
params.setAppId(getResources().getString(R.string.AppId));
//我方提供的LoginKey（登录Key）
params.setLoginKey(getResources().getString(R.string.AppKey));
PlayerGameUnionSdk.getInstance().initSdk(params, gameUnionSdkCallback);
```

```

private IGameUnionSdkCallback gameUnionSdkCallback = new IGameUnionSdkCallback() {
    @Override
    public void initSuccess() {
        log( s: "initSuccess", o: "初始化成功");
        isInit = true;
        Toast.makeText( context: GameActivity.this, text: "初始化成功", Toast.LENGTH_SHORT).show();

        runOnUiThread(() -> {
            if (loadingLayout != null && loadingLayout.getVisibility() == View.VISIBLE) {
                loadingLayout.setVisibility(View.GONE);
            }
            PlayerGameUnionSdk.getInstance().login();
        });
    }

    @Override
    public void loginSuccess(SdkToken token) {
        log( s: "loginSuccess",
            o: "登录成功\n用户信息: \n" + token +
                "\nuid:" + token.getUserID() +
                "\nuserName:" + token.getSdkUsername());
        showGame(token.getGameUrl());
    }

    @Override
    public void submitGameInfoSuccess() { log( s: "submitGameInfoSuccess", o: "提交游戏角色信息成功"); }

    @Override
    public void logout() {
        log( s: "logout", o: "注销成功");
        showLogout();
        showLogin();

        runOnUiThread(() -> {
            if (isSwitch) {
                //判断是否由切换账号触发的
                PlayerGameUnionSdk.getInstance().login();
            }
        });
    }
}

```

```

@Override
public void payCallback(String sdkOrderId) {
    log( s: "payCallback", o: "支付结果\n订单号: \n" + sdkOrderId);
}

@Override
public void onExist(boolean isFinish) {
    log( s: "onExist", o: "退出游戏, isFinish : " + isFinish);
    if (isFinish) {
        //如果isFinish为True,则退出游戏
        showLogout();

        GameActivity.this.finish();
        System.exit( status: 0);
    }
}

@Override
public void onErrorCallback(GameUnionCode code, String message) {
    log( s: "onErrorCallback", o: "Code : " + code + " , Message : " + message);
    Toast.makeText( context: GameActivity.this, text: "ErrorCode : " + code + " , Message : " + message, Toast.LENGTH_SHORT).show();
};

```

2.6.2 登录(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().login();
```

登录回调需要在初始化接口的 IGameUnionSdkCallback 参数的 loginSuccess(SdkToken token)接口或 onErrorCallback(GameUnionCode code, String message)接收。

接收到登录成功回调之后, 需要进行二次登录校验。具体接口请参考服务端文档

2.6.3 切换账号(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().switchLogin();
```

需要在游戏内，保留一个切换账号的功能。如果游戏没有此功能，需要添加一个，保证用户可以在游戏中切换不同的账号。

2.6.4 注销账号(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().logout();
```

2.6.5 支付(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().pay(GameUnionPayParams params)
```

GameUnionPayParams 类说明

描述：支付参数

参数名	参数类型	参数说明
productid	String	商品 id 最小值填为 1 不可为空
productName	String	商品名 不可为空
productDesc	String	商品描述 不可为空
price	String	商品单价（单位为元） 数量最小为 1 不可为空
ratio	Int	兑换比例(RMB 与游戏币兑换比例) 默认值为 1
buyNum	Int	购买数量 不可为空 数量最小为 1
coinNum	Int	游戏币数量 不可为空 数量最小为 1
serverID	String	游戏服务器号

		不可为空
serverName	String	游戏服务器名 不可为空
roleID	String	游戏角色 id 不可为空
roleName	String	游戏角色名 不可为空
roleLevel	Int	角色等级 不可为空 最小值为 1
payNotifyUrl	String	充值回调地址（以配置到服务器的为准，建议这里值为 ""）
vip	String	vip 等级 值最小为 1
orderId	String	游戏方订单号 不可为空
extension	String	扩展字段

*没有备注的信息，可为 Null 或 "" 值

调用示例:

```
GameUnionPayParams params = new GameUnionPayParams();
JSONObject jsonObject = new JSONObject(data);
coinNum = jsonObject.optInt( name: "coinNum");
ratio = jsonObject.optInt( name: "ratio");
productID = jsonObject.optString( name: "productID");
productName = jsonObject.optString( name: "productName");
productDesc = jsonObject.optString( name: "productDesc");
price = jsonObject.optString( name: "price");
buyNum = jsonObject.optInt( name: "buyNum");
roleID = jsonObject.optString( name: "roleID");
roleName = jsonObject.optString( name: "roleName");
roleLevel = jsonObject.optInt( name: "roleLevel");
serverID = jsonObject.optString( name: "serverID");
serverName = jsonObject.optString( name: "serverName");
vip = jsonObject.optString( name: "vip");
payNotifyUrl = jsonObject.optString( name: "payNotifyUrl");
extension = jsonObject.optString( name: "extension");
orderId = jsonObject.optString( name: "orderId");

params.setCoinNum(coinNum);           //游戏币数量 没有则传"", 不能为空
params.setRatio(ratio);               //兑换比例(RMB与游戏币兑换比例) (必填) 一般为10
params.setProductID(productID);       //商品 id (必填)
params.setProductName(productName);    //商品名 (必填)
params.setProductDesc(productDesc);    //商品描述 (必填)
params.setPrice(price);                //商品单价 (RMB:元, 必填)
params.setBuyNum(buyNum);              //购买数量 (必填)
params.setRoleID(roleID);              //游戏角色id (必填)
params.setRoleName(roleName);          //游戏角色名 (必填)
params.setRoleLevel(roleLevel);        //角色等级 (必填)
params.setServerID(serverID);          //游戏服务器号 (必填)
params.setServerName(serverName);      //游戏服务器名 (必填)
params.setVip(vip);                    //vip等级 (必填)
params.setPayNotifyUrl(payNotifyUrl);  //充值回调地址
params.setExtension(extension);         //扩展字段 (必填) 没有则传"", 不能为空
params.setOrderID(orderID);            //订单号 (一定要的)
PlayerGameUnionSdk.getInstance().pay(params);
```

2.6.6 角色信息上传(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().submitGameRoleInfo(GameUnionUserExtraData extraData);
```

游戏方需要在创建角色、选择服务器、进入游戏、等级提升、退出游戏等地方调用该接口，传递数据给我方。

GameUnionUserExtraData 类说明

描述：游戏角色信息类

参数名	参数类型	参数说明
datatype	Int	上传时机 值参考下表的 DataType 说明
serverId	String	服务器 ID
serverName	String	服务器名
userId	String	用户 ID
userName	String	用户名
payLevel	String	充值等级
roleId	String	角色 ID
roleName	String	角色名
roleLevel	String	角色等级
extension	String	透传参数
roleCTime	long	角色创建时间(单位： 秒)， 长度 10，获取 服务器存储的时间， 不可用手机本地时间

DataType 说明:

描述：游戏角色信息上传时机

参数名	值	参数说明
TYPE_SELECT_SERVER	1	选择服务器
TYPE_CREATE_ROLE	2	创建角色
TYPE_ENTER_GAME	3	进入游戏
TYPE_LEVEL_UP	4	等级提升
TYPE_EXIT_GAME	5	退出游戏

调用示例:

```

GameUnionUserExtraData userExtraData = new GameUnionUserExtraData();
JSONObject jsonObject = new JSONObject(data);
userID = jsonObject.optString( name: "userID");
roleId = jsonObject.optString( name: "roleId");
roleName = jsonObject.optString( name: "roleName");
roleLevel = jsonObject.optString( name: "roleLevel");
roleCTime = jsonObject.optLong( name: "roleCTime");
rayLevel = jsonObject.optString( name: "rayLevel");
serverID = jsonObject.optString( name: "serverID");
serverName = jsonObject.optString( name: "serverName");
userName = jsonObject.optString( name: "userName");
extension = jsonObject.optString( name: "extension");
dataType = jsonObject.optInt( name: "dataType");

switch (dataType) {
    case 2:
        dataType = GameUnionUserExtraData.TYPE_CREATE_ROLE;
        break;
    case 3:
        dataType = GameUnionUserExtraData.TYPE_ENTER_GAME;
        break;
    case 4:
        dataType = GameUnionUserExtraData.TYPE_LEVEL_UP;
        break;
}

userExtraData.setDataType(dataType);
userExtraData.setUserId(userID);
userExtraData.setRoleId(roleID);
userExtraData.setRoleName(roleName);
userExtraData.setRoleLevel(roleLevel);
userExtraData.setRoleCTime(roleCTime);
userExtraData.setPayLevel(rayLevel);
userExtraData.setServerId(serverID);
userExtraData.setServerName(serverName);
userExtraData.setUserName(userName);
userExtraData.setExtension(extension);
PlayerGameUnionSdk.getInstance().submitGameRoleInfo(userExtraData);

```

2.6.7 退出游戏(必接)

调用接口:

```
PlayerGameUnionSdk.getInstance().exit();
```

游戏方除了在游戏内需要退出游戏的地方调用该接口外，还需要重写 Activity 的 onBackPressed()方法，调用该接口

调用示例:

```
@Override
public void onBackPressed() {
    PlayerGameUnionSdk.getInstance().exit();
}
```

2.7 生命周期监听(必接)

因为部分渠道需要，需要游戏方将游戏内某些生命周期触发时回调给我方。

- 1、onActivityResult
- 2、onNewIntent
- 3、onRestart
- 4、onRequestPermissionsResult
- 5、onConfigurationChanged

以上生命周期，需要游戏方在游戏主界面重写以上方法，并调用我们 SDK 方法 (onActivityResult、onNewIntent、onRestart、onRequestPermissionsResult、onConfigurationChanged) 进行回调，图示：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    PlayerGameUnionSdk.getInstance().onActivityResult( activity: GameActivity.this, requestCode, resultCode, data);
    super.onActivityResult(requestCode, resultCode, data);
}

@Override
protected void onNewIntent(Intent intent) {
    PlayerGameUnionSdk.getInstance().onNewIntent( activity: GameActivity.this, intent);
    super.onNewIntent(intent);
}

@Override
protected void onRestart() {
    PlayerGameUnionSdk.getInstance().onRestart( activity: GameActivity.this);
    super.onRestart();
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    PlayerGameUnionSdk.getInstance().onRequestPermissionsResult( activity: GameActivity.this, requestCode, permissions, grantResults);
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    PlayerGameUnionSdk.getInstance().onConfigurationChanged( activity: GameActivity.this, newConfig);
    super.onConfigurationChanged(newConfig);
}
```

2.8 混淆配置说明

2.8.1 SDK 混淆

```
-dontwarn com.qq.gdt.action.**
-keep class com.qq.gdt.action.** {*;}

-dontwarn class com.pillowcase.**
-keep class com.pillowcase.**{
    public *;
}
```

```

-keep class com.bytedance.**{
    *;
}
-keep class com.ss.android.common.**{
    *;
}
-keep class com.bun.miitmdid.** {*; }
-keep class com.zt.emulator.EmulatorUtils{
    public *;
}

-dontwarn class com.player.sdk.**
-keep class com.player.sdk.**{
    *;
}

-dontwarn class com.player.game.union.**
-keep class com.player.game.union.**{
    *;
}

```

2.8.2 第三方库混淆

```

-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }
-dontwarn com.squareup.okhttp.**

# OkHttp3
-dontwarn com.squareup.okhttp3.**
-keep class com.squareup.okhttp3.** { *; }
-dontwarn okio.**

# Okio
-dontwarn com.squareup.**
-dontwarn okio.**
-keep public class org.codehaus.* { *; }
-keep public class java.nio.* { *; }

# Gson
-keepattributes Signature,-keepattributes,*Annotation*
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.stream.** { *; }

```

3 Q&A

Q :

A :