# EagleSDK接入文档

[EagleSDK接入文档 1](#_Toc492567437)

[1. 前述 2](#_Toc492567438)

[2. 接入流程 2](#_Toc492567439)

[3. 开始接入 3](#_Toc492567440)

[3.1 引入jar包 3](#_Toc492567441)

[3.2 配置Application 4](#_Toc492567442)

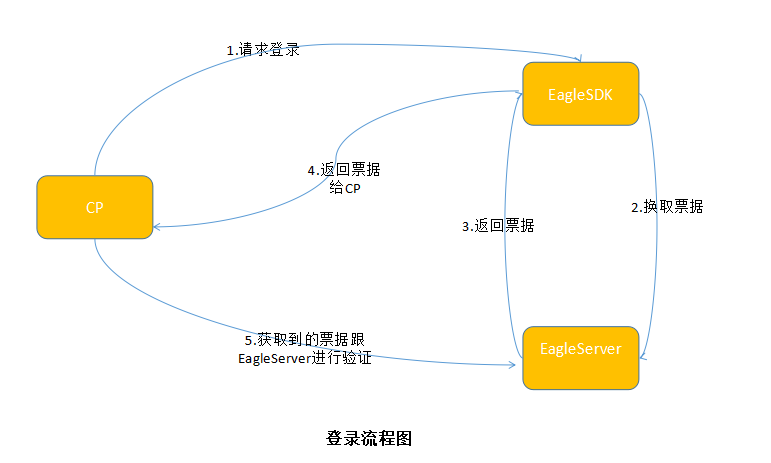
[3.3 必须调用的接口 5](#_Toc492567443)

### 前述

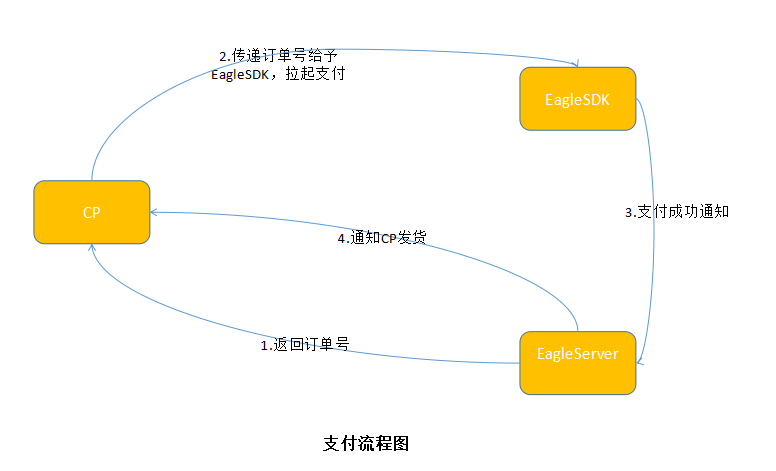
EagleSDK为CP商提供一套第三方SDK接入解决方案，整个接入过程，不改变任何SDK的功能、特性、参数等，对于最终玩家而言是完全透明无感知的。目的是让CP商能有更多的时间更专注于游戏本身的品质，所有SDK的接入工作统统交给我们吧。第三方SDK包括了渠道SDK、用户系统、支付系统、广告系统、统计系统、分享系统等等。有助于CP方轻松接入第三方SDK！

### 接入流程

1. 登录流程



1. 支付流程



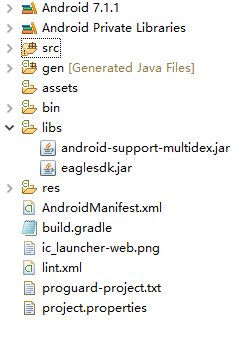
注：

1. 建议任何关于CP方与EagleServer之间的交互都应该放到CP Server方进行
2. CP方需要提供一个发货通知地址给予EagleServer
3. 在登录流程的第四步，CP方可通过EagleSDK提供的接口获取到相应的票据

### 开始接入

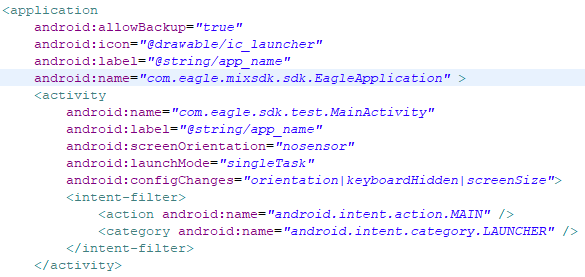
#### 引入jar包

将eaglesdk.jar和android-support-multidex.jar引入到游戏工程的libs目录下



#### 配置Application

如果游戏层没有自己的application，那么游戏需要将androidmanifest.xml中的application指定为com.eagle.mixsdk.sdk.EagleApplication.



注：

1. 如果游戏有自己的application，或者需要在Application的生命周期方法中，做一些操作。那么可以实现IApplicationListener里面的接口。比如，定义一个GameProxyApplication实现IApplicationListener接口.

**public** **class** GameProxyApplication **implements** IApplicationListener{

@Override

**public** **void** onProxyCreate() {

// **TODO** 需要在Application的onCreate中的操作，放在这里

//如果需要获取到原始Application对象，通过 //EagleSDK.getInstance().getApplication()来获取

Log.d("Game", "On Application Create");

}

@Override

**public** **void** onProxyAttachBaseContext(Context base) {

// **TODO** 需要在Application的attachBaseContext中的操作，放在这里

}

@Override

**public** **void** onProxyConfigurationChanged(Configuration config) {

// **TODO** 需要在Application的onConfigurationChanged中的操作，放在里

}

@Override

**public** **void** onProxyTerminate() {

// **TODO** 需要在Application的onTerminate中的操作，放在这里

}

}

1. 配置GameProxyApplication,上面定义了游戏的Application，紧接着，在AndroidManifest.xml中，将上面我们定义的GameProxyApplication,配置到meta-data中,android:name为Eagle\_Game\_Application.

<meta-data

android:name="Eagle\_Game\_Application"

android:value="你定义的GameProxyApplication类路径" />

#### 必须调用的接口

NOTE:所有的接口调用，都通过com.eagle.mixsdk.sdk.platform.EaglePlatform单例类来调用

游戏技术人员在调用以下接口前可以通过以下方法先自行判断对应的接口是否存在

EaglePlatform.getInstance().isSupportMethod(String method)

其中参数method既是需要判断的接口名

|  |  |
| --- | --- |
| 接口名 | 接口描述 |
| login | 登录 |
| loginCustom | 渠道登录 |
| switchLogin | 切换账号 |
| showAccountCenter | 显示账号中心 |
| logout | 登出 |
| submitExtraData | 发送扩展参数 |
| exit | 退出 |
| postGiftCode | 上传礼包兑换码 |
| realNameRegister | 实名注册方法 |
| queryAntiAddiction | 防沉迷系统查询接口 |
| queryProducts | 查询商品信息 |

1. **初始化(必接)**

该方法必须在游戏启动Activiity的onCreate方法中进行调用

EaglePlatform.getInstance().init(this, new EagleInitListener() {

@Override

public void onSwitchAccount(EagleToken data) {//必须实现接口

//游戏技术再次回调可以直接使用EagleToken内的自带参数值，游戏技术只要接收到此接口回调，回到游戏登录界面而且并不需要再次调用渠道登录接口，而是直接使用EagleToken内的自带参数值进行游戏登录。

}

@Override

public void onPayResult(int code, String msg) {

Log.d("EagleSDK", "pay result. code:"+code+";msg:"+msg);

switch(code){

case EagleCode.CODE\_PAY\_SUCCESS:

Toast.makeText(MainActivity.this, "支付成功", Toast.LENGTH\_LONG).show();

break;

case EagleCode.CODE\_PAY\_FAIL:

Toast.makeText(MainActivity.this, "支付失败", Toast.LENGTH\_LONG).show();

break;

case EagleCode.CODE\_PAY\_CANCEL:

Toast.makeText(MainActivity.this, "支付取消", Toast.LENGTH\_LONG).show();

break;

case EagleCode.CODE\_PAY\_UNKNOWN:

Toast.makeText(MainActivity.this, "未知错误", Toast.LENGTH\_LONG).show();

break;

}

}

@Override

public void onLogout() {

//用户登出回调（需要收到该回调需要返回游戏登录界面，并调用login接口，打开SDK登录界面）

}

//渠道SDK登录并去Eagleserver登录认证成功，会回调改方法。

//EagleToken中的参数如下：

//userID:EagleServer生成的唯一用户ID，游戏服务器需要将游戏账户ID和该 userID进行绑定。

//sdkUserID:渠道SDK平台用户唯一ID，一般不需要使用

//username:EagleServer生成的用户名，比如//234234234.uc,4353453453.baidu,65756756756.360

//sdkUserName:渠道SDK平台用户名，可能为空，部分渠道SDK没有返回用户名

//token:EagleServer生成的会话ID，游戏服务器拿该字段去EagleServer做二次登录验证

//extension:EagleServer返回的扩展字段，部分渠道SDK需要，游戏中无需使用该字段的值

//timestamp:EagleServer生成的时间戳，游戏服务器去EagleServer做二次登录验证时，传给EagleServer

@Override

public void onLoginResult(int code, EagleToken data) {

switch(code){

case EagleCode.CODE\_LOGIN\_SUCCESS:

//进入游戏

//从EagleToken中获取用户信息

break;

}

}

@Override

public void onInitResult(int code, String msg) {

Log.d("EagleSDK", "init result.code:"+code+";msg:"+msg);

switch(code){

case EagleCode.CODE\_INIT\_SUCCESS:

Toast.makeText(MainActivity.this, "初始化成功",Toast.LENGTH\_LONG).show();

break;

case EagleCode.CODE\_INIT\_FAIL:

Toast.makeText(MainActivity.this, "初始化失败",Toast.LENGTH\_LONG).show();

break;

}

}

@Override

public void onLoginFail(int code, String msg){

//登录失败回调

}

});

1. **登录接口(必接)**

调用登录接口,打开SDK登录界面,登录成功或者失败，会触发上面初始化监听器中的onLoginResult回调函数

EaglePlatform.getInstance().login(this);

1. **登出接口(必接)**

调用登出接口,SDK账户登出,但是不是每一个SDK都具有登出逻辑,对应没有提供登出接口的SDK,调用该函数,什么也不操作,会触发上面初始化监听器中的onLogout回调函数

EaglePlatform.getInstance().logout();

1. **提交扩展参数接口(必接)**

Note:部分渠道要求选择服务器,创建角色,登录游戏,角色升级,退出游戏等时刻,必须要上报游戏中的玩家数据,以便渠道后台统计用户数据.所以,游戏层需要在特定的地方多次调用该方法.

**private** **void** submitExtraData(**final** **int** dataType){

UserExtraData data = **new** UserExtraData();

Data.setDataType(dataType);

data.setMoneyNum(100);

data.setRoleCreateTime(System.currentTimeMillis()/1000);

data.setRoleID("role\_100");

data.setRoleName("test\_112");

data.setRoleLevel("10");

data.setRoleLevelUpTime(System.currentTimeMillis()/1000);

data.setServerID(10);

data.setServerName("server\_10");

EaglePlatform.getInstance().submitExtraData(data);

}

该方法将调用的时机分为几种类型:

1. 选择服务器
2. 创建角色
3. 进入游戏
4. 等级提升
5. 退出游戏

所以在上面5个地方都需要调用

EaglePlatform.getInstance().submitExtraData(data);

其中，UserExtraData就是游戏玩家的数据,比如:

选择服务器时,extraData中的 dataType为1；

创建角色的时候,dataType为2；

进入游戏时,dataType为3；

等级提 升时dataType为4；

退出游戏时datatType为5

Note:选择服务器时,因为还没有进入游戏，无法知道角色数据，extraData中只需要传 入服务器信息即可。

关于UserExtraData的数据结构:

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 参数说明 |
| dataType | int | 调用时机 |
| serverID | String | 玩家所在服务器的ID |
| serverName | String | 玩家所在的服务器名称 |
| roleID | String | 玩家角色ID |
| roleName | String | 玩家角色ID |
| roleLevel | String | 玩家角色等级 |
| moneyNum | String | 当前角色身上拥有的游戏币 |
| roleCreateTime | Long | 角色创建时间 |
| roleLevelUpTime | Long | 角色等级变化 |
| Vip | String | 玩家VIP等级 |

1. **退出接口(必接)**

当玩家点击手机上的返回键或者游戏中[退出游戏]按钮,部分渠道SDK会弹出一个”退出对话框”,让玩家确认退出,或者取消退出游戏层需要在EagleExitListener的onGameExit中,来调用游戏自己的退出游戏确认弹窗(对于没有提供退出确认弹窗接口的SDK,会触发该回调)

EaglePlatform.getInstance().exitSDK(**new** EagleExitListener() {

@Override

**public** **void** onGameExit() {

//TODO游戏自己的退出确认框

}

});

1. **支付充值(必接)**

调用充值接口,打开SDK充值界面.充值成功或者失败,会触发上面初始化监听器中的onPayResult接口

PayParams params = **new** PayParams();

params.setBuyNum(1); //购买数量，固定1

params.setCoinNum(100); //当前玩家身上拥有的游戏币数量

//游戏自定义数 据，充值成功，回调游戏服的时候，会原封不动返回

params.setExtension(System.currentTimeMillis()+""); params.setPrice(1); //单位 整数 元

params.setProductId("1"); //产品ID

params.setProductName("元宝"); //产品名称

params.setProductDesc("购买100元宝"); //产品描述

params.setRoleId("1"); //角色ID

params.setRoleLevel(1); //角色等级

params.setRoleName("测试角色名"); //角色名称

params.setServerId("10"); //服务器ID

params.setServerName("测试"); //服务器名称

params.setVip("vip1"); //角色VIP等级

params.setPayNotifyUrl("http://www.game.com/pay/callback"); //支付成功，EagleServer异步通知该地址，告诉游戏服务器发货

EaglePlatform.getInstance().pay(**this**, params);

关于PayParams对象

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 参数说明 |
| productId | String | 充值商品ID |
| productName | String | 商品名称 |
| productDesc | String | 商品描述 |
| Price | int | 充值金额 |
| buyNum | int | 购买数量 |
| coinNum | Int | 玩家当前身上剩余的游戏币 |
| serverID | String | 玩家所在的服务器ID |
| ServerName | String | 玩家所在服务器名称 |
| roleID | String | 玩家角色ID |
| roleName | String | 玩家角色名称 |
| roleLevel | String | 玩家角色等级 |
| vip | String | 玩家VIP等级 |
| payNotifyUrl | String | 游戏服务器支付回调地址 |
| extension | String | 额外扩展参数 |

1. **生命周期接口(必接)**

在游戏启动Activity的如下生命周期函数中，调用对应的方法

**public** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data){

EaglePlatform.getInstance().onActivityResult(requestCode, resultCode, data);

**super**.onActivityResult(requestCode, resultCode, data);

}

**public** **void** onStart(){

EaglePlatform.getInstance().onStart();

**super**.onStart();

}

**public** **void** onPause(){

EaglePlatform.getInstance().onPause();

**super**.onPause();

}

**public** **void** onResume(){

EaglePlatform.getInstance().onResume();

**super**.onResume();

}

**public** **void** onNewIntent(Intent newIntent){

EaglePlatform.getInstance().onNewIntent(newIntent);

**super**.onNewIntent(newIntent);

}

**public** **void** onStop(){

EaglePlatform.getInstance().onStop();

**super**.onStop();

}

**public** **void** onDestroy(){

EaglePlatform.getInstance().onDestroy();

**super**.onDestroy();

}

**public** **void** onRestart(){

EaglePlatform.getInstance().onRestart();

**super**.onRestart();

}

**public** **void** onConfigurationChanged(Configuration newConfig){

EaglePlatform.getInstance().onConfigurationChanged(newConfig);

**super**.onConfigurationChanged(newConfig);

}

**public** **void** onRequestPermissionsResult(**int** requestCode, String[] permissions, **int**[] grantResults){

EaglePlatform.getInstance().onRequestPermissionResult(requestCode, permissions, grantResults);

}

**3.3.8 切换账号(可选)**

切换账号功能主要用于调用渠道的切换账号接口，游戏技术人员在游戏内点击游戏的切换账号按钮，调用以下代码即可完成对渠道切换账号接口的对接。

注：由于目前市面上大部分渠道的切换账号功能都不是必接，而且部分渠道并不存在直接的切换账号功能接口，所以建议游戏可不对接EagleSDK切换账号接口，建议处理方式为：点击游戏内切换账号按钮即可直接注销游戏状态回到游戏登录界面。

**if**(EaglePlatform.*getInstance*().isSupportMethod("switchLogin")){

EagleUser.*getInstance*().switchLogin();

}

1. **出包注意事项**
2. 游戏按照以上文档接入完EagleSDK后，应当仔细检查是否存在接口漏接情况
3. 出母包前请仔细认真审查是否存在必要文件尚未修改或者添加
4. 游戏出给平台的母包必须一定不能带有任何第三方渠道插件或者文件和资源，否则导致重新排期出母包延期提审日期，后果自负。