# 客户端接入指导手册

SDK version：5.0

# 说明

SDK接入框架，能让手游开发商以最短的时间接入渠道SDK，让游戏快速上线。手游开发商只需要接此抽象层框架，然后通过本地打包工具打包，即可打出最终的各个渠道SDK的渠道包整个过程，方便快捷，安全可靠。

# 1.用户相关接口的调用

注：以下方法都必须在(UI)主线程中调用 如下：

U8SDK.getInstance().runOnMainThread(new Runnable() {

@Override

public void run() {

调用登录U8User.getInstance().login();

}});

|  |  |  |  |
| --- | --- | --- | --- |
| **方法名称** | **必须** | **说明** | **方法** |
| **登录** | 是 | 为了打开SDK登录界面，我们需要在UI线程中调用 | **U8User.getInstance().login();** |
| **退出** | 是 | 退出游戏 | **U8User.getInstance().exit();** |
| **切换账号** | 是 | 部分渠道要求游戏内必须要有切换账号 | **U8User.getInstance().switchLogin();** |
| **登出账号** | 是 | 一般游戏中只需要实现切换账号功能，无需实现登出功能。该功能，是登出SDK账号，但是仅仅部分渠道支持，对于不支持登出的渠道，该方法默认不做任何逻辑 | **U8User.getInstance().logout();** |
| **提交游戏数据** | 是 | 部分渠道要求在 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。[【详细5】](#_5. 【提交扩展数据】) | **U8User.getInstance().submitExtraData (extraData);** |
| **支付** | 是 | 支付接口比较简单，就是一个pay方法。游戏中仅仅需要将需要的参数传进去即可。[【详细6】](#_5.6 【支付方法】 游戏中仅仅需要将需要的参数传进去即可) | **U8Pay.getInstance().pay(params);** |

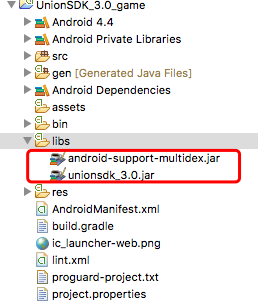
# 

# 2.接入配置

## 2.1 添加依赖库

**SDK提供了两个依赖库【union4.0.jar android-support-multidex.jar】导入对应的目录即可**

EXE：



# 3.开始接入(可参照demo)

## 3.1初始化SDK接入框架

第一步：

你需要将游戏母工程的AndroidManifest.xml的application设置为com.u8.sdk.U8Application

注意：把启动页的android:label="@string/app\_name"删除

第二步：

在游戏的主Activity的onCreate方法中调用init方法：

U8SDK.getInstance().init(Activity context);

U8SDK.getInstance().onCreate();

第三步：

在主Activity的生命周期方法中，也需要对应调用U8SDK对应的方法：

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

U8SDK.getInstance().onActivityResult(requestCode, resultCode, data);

super.onActivityResult(requestCode, resultCode, data);

}

public void onStart(){

U8SDK.getInstance().onStart();

super.onStart();

}

public void onPause(){

U8SDK.getInstance().onPause();

super.onPause();

}

public void onResume(){

U8SDK.getInstance().onResume();

super.onResume();

}

public void onNewIntent(Intent newIntent){

U8SDK.getInstance().onNewIntent(newIntent);

super.onNewIntent(newIntent);

}

public void onStop(){

U8SDK.getInstance().onStop();

super.onStop();

}

public void onDestroy(){

U8SDK.getInstance().onDestroy();

super.onDestroy();

}

public void onRestart(){

U8SDK.getInstance().onRestart();

super.onRestart();

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

U8SDK.getInstance().onConfigurationChanged(newConfig);

super.onConfigurationChanged(newConfig);

}

@Override

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

U8SDK.getInstance().onRequestPermissionsResult(requestCode, permissions, grantResults);

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

}

@Override

public void onBackPressed() {

U8SDK.getInstance().onBackPressed();

if (U8User.getInstance().isSupport("exit")) {

U8User.getInstance().exit();

} else {

this.finish();

System.exit(0);

}

}

# 4.设置回调监听

登录，登出，支付以及所有失败等情况，SDK接入层都会通过对应的事件，通知到抽象层中。所以，我们需要在调用初始化方法之前先设置一个监听类来监听这些事件：

### 4.1设置监听(以下回调结果如果需要在界面进行操作，请在UI线程中进行。)

U8SDK.getInstance().setSDKListener(new IU8SDKListener(){

### 4.2.1 初始化回调 :

@Override

public void onInitResult(int code, String string) {

switch (code) {

case U8Code.CODE\_INIT\_SUCCESS:// 初始化成功

break;

case U8Code.CODE\_INIT\_FAIL:// 初始化失败

break;

}

}

### 4.2.2 登录成功的回调:

这里是sdk登陆回调

@Override

public void onLoginResult(int code,String result) {

switch (code) {

case U8Code.CODE\_LOGIN\_SUCCESS:// 登录成功

break;

case U8Code.CODE\_LOGIN\_FAIL://// 登录失败

break;

}

}

### 4.2.3 登录认证的回调:

登录认证成功将UToken里面的参数发送游戏服，让游戏服进行处理

@Override

public void onAuthResult(UToken authResult) {

this.runOnUiThread(new Runnable() {

@Override

public void run() {

if(authResult.isSuc()){//登陆认证成功，游戏客户端进行操作

}else{ //登陆认证失败

}

}});}

|  |  |  |
| --- | --- | --- |
| **UToken说明** | **类型** | **说明** |
| **suc** | Boolean | 登录认证是否成功 true：成功 false：失败 |
| **userID** | Int | Union-server生成唯一用户ID（游戏服务器登录认证使用） |
| **sdkUserID** | String | SDK-server 返回用户的ID |
| **username** | String | Union-server返回统一格式的用户名 |
| **sdkUserName** | String | SDK-server返回的用户名 |
| **token** | String | Union-server生成的token（游戏服务器登录认证使用） |
| **extension** | String | 扩展数据字段。有特殊需求的数据可以在这里添加 |

### 4.2.4切换帐号成功事件：

调用onSwitchLogin()接口。游戏中监听对应的事件时，收到onSwithLogin时，需要返回 游戏的登录界面，并显示SDK的登录界面(大部分渠道)

@Override

public void onSwitchLogin(int code,String message) {

switch (code) {

case U8Code.CODE\_SWITCH\_ACCOUNT\_SUCCESS:// 切换账号成功

break;

case U8Code.CODE\_SWITCH\_ACCOUNT\_FAIL:// 切换账号失败

break;

}

}

### 4.2.5 登出或者SDK登录会话失效的回调:

如果玩家正在游戏中玩，需要退出到登录界面，重新登录

@Override

public void onLogoutResult(int code,String message) {

switch (code) {

case U8Code.CODE\_LOGOUT\_SUCCESS:// 登出账号成功

break;

case U8Code.CODE\_LOGOUT\_FAIL:// 登出账号账号失败

break;

}

}

### 4.2.6 支付回调：

该接口给单机游戏使用，网游不用实现。单机游戏通过这个接口，表示支付成功，给玩家 发道具

@Override

public void onPayResult(int code,String message) {

switch (code) {

case U8Code.CODE\_PAY\_SUCCESS:// 支付成功

break;

case U8Code.CODE\_PAY\_FAIL:// 支付失败

break;

}

}

### 4.2.6 特殊回调：（例如防沉迷,实名认证）

@Override

public void onResult(int code,String message) {

}

# 5. 【提交扩展数据】

**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**

**此方法调用的时机分为几种类型：（datatype = 对应的值即1，2，3，4，5为必须调用）**

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 6：进入副本 7：退出副本 8：VIP等级提升

U8User.getInstance().submitExtraData(UserExtraData extraData);

UserExtraData参数说明：

|  |  |  |
| --- | --- | --- |
| **参数（UserExtraData）** | **类型** | **说明** |
| dataType | Int | 调用时机 |
| serverID | Int | 玩家所在服务器的ID |
| serverName | String | 玩家所在服务器的名称 |
| roleID | String | 角色ID |
| roleName | String | 玩家角色名称 |
| roleLevel | String | 玩家角色等级 |
| moneyNum | Int | 角色身上拥有的游戏币数量 |
| roleCreateTime | long | 角色创建时间，从1970年到现在的时间，单位秒 |
| roleLevelUpTime | long | 角色等级变化时间（同上） |
| Vip | String | 玩家VIP等级 |

## 

# 6.【支付方法】

## 游戏中仅仅需要将需要的参数传进去即可

U8Pay.getInstance().pay(params);

params 支付参数说明：

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| **productId** | String | 充值商品ID，游戏内的商品ID |
| **productName** | String | 商品名称，比如100元宝，500钻石... |
| **productDesc** | String | 商品描述，比如 充值100元宝，赠送20元宝 |
| **price** | int | 充值金额(单位：元) |
| **ratio** | int | 兑换比例，暂时无用 |
| **buyNum** | int | 购买数量 1 |
| **coinNum** | int | 玩家当前身上剩余的游戏币数量 |
| **serverId** | String | 服务器id |
| **serverName** | String | 服务器名 |
| **roleId** | String | 角色id |
| **roleName** | String | 角色名 |
| **roleLevel** | int | 角色等级 |
| **payNotifyUrl** | String | 支付回调地址 |
| **vip** | String | 玩家Vip等级 |
| **extension** | String | 透传参数 |

例：

U8SDK.getInstance().runOnMainThread(new Runnable() {

@Override

public void run() {

PayParams params = new PayParams();

params.setProductId("商品id");// 商品id

params.setProductName("元宝");

params.setProductDesc("购买100元宝");

params.setPrice(1);// 单价:元

params.setRatio(10);// 对话比例

params.setBuyNum(1);// 数量

params.setCoinNum(1);// 金额

params.setServerId("10");

params.setServerName("测试");

params.setRoleId("1");

params.setRoleName("测试角色名");

params.setRoleLevel(10);

params.setPayNotifyUrl("http://");

params.setVip("vip1");

params.setExtension(null);// 透传字段

U8Pay.getInstance().pay(params);

}});