# Facebook 接入文档

[Facebook 接入文档 1](#_Toc63263402)

[一 添加Facebook Package 1](#_Toc63263403)

[1 下载package 1](#_Toc63263404)

[2 导出package 1](#_Toc63263405)

[3 添加Facebook设置 3](#_Toc63263406)

[二代码部分添加： 4](#_Toc63263407)

[1 FB初始化 4](#_Toc63263408)

[2 FB.ActivateApp 5](#_Toc63263409)

[3.FB登录 5](#_Toc63263410)

[4 FB获取用户名和用户id 7](#_Toc63263411)

[5 FB获取头像 8](#_Toc63263412)

[7 FB获取玩同一个游戏的好友列表 8](#_Toc63263413)

[7.1 确保所开发的项目在FB后台时开发中的状态 8](#_Toc63263414)

[7.2 给FB测试账号添加权限和好友 8](#_Toc63263415)

[7.3 添加代码获取好友信息 9](#_Toc63263416)

## 一 添加Facebook Package

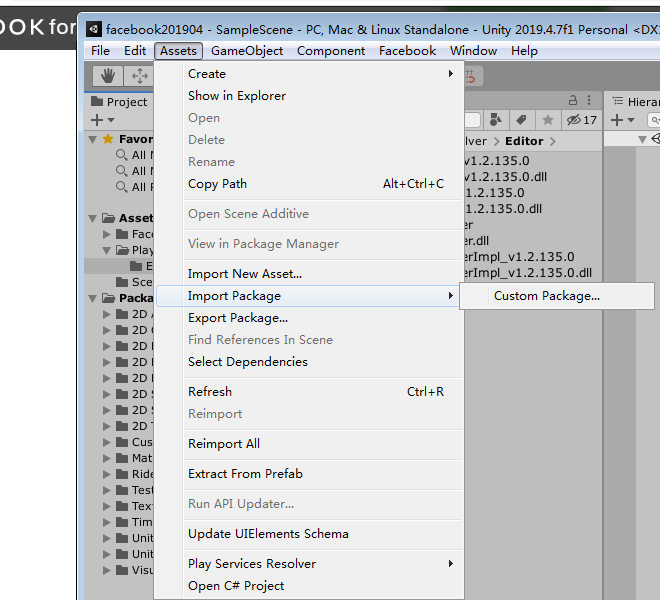
1 下载package

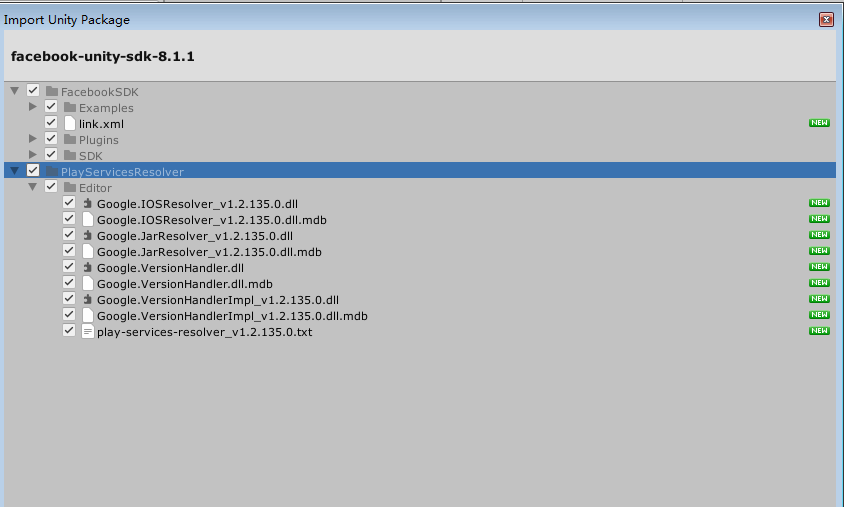
facebook unity sdk下载链接，当前使用版本8.1.1

<https://developers.facebook.com/docs/unity/>

2 导出package

unity->assets->import package->选择下载解压好的facebook unitypackage



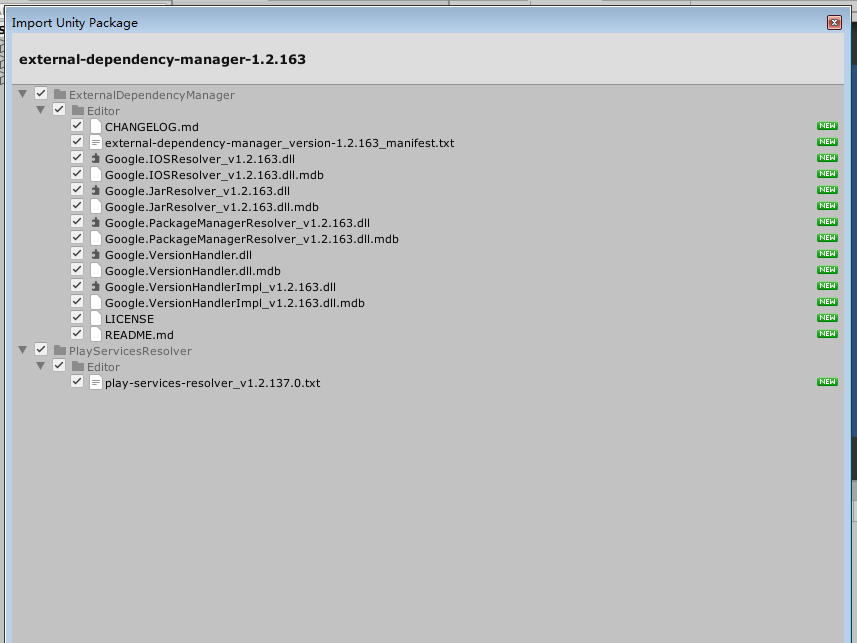


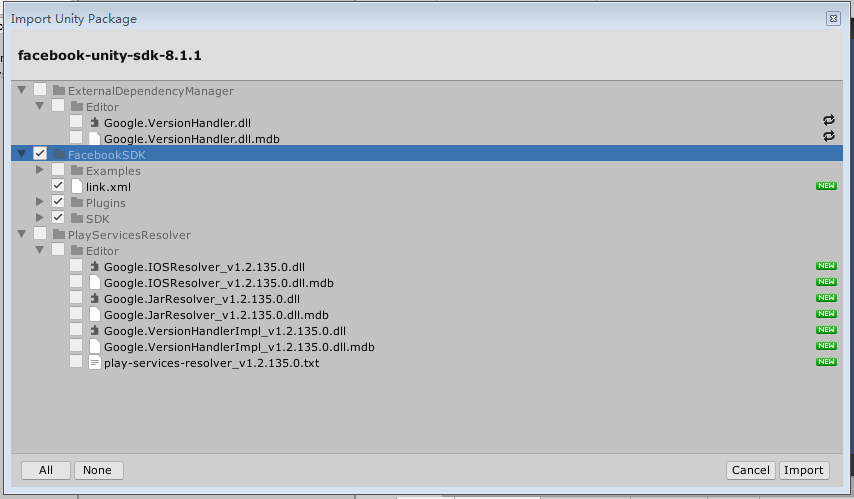
注意事项： 8.1.1版本中包含facebook sdk和PlayerServicesResolver插件，如果unity项目工程中已经包含PlayerServicesResolver插件并且版本比较高，可以不用勾选fbsdk中携带的PlayerServicesResolver插件。

External Dependency Manager（以前叫Play Services Resolver）也可以单独安装，下载链接

https://github.com/googlesamples/unity-jar-resolver

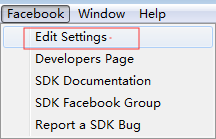
安装包的名字是external-dependency-manager

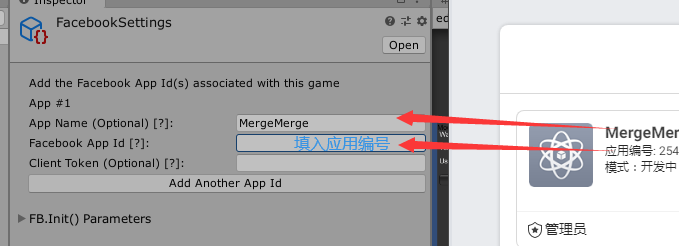




3 添加Facebook设置

导入package后，unity编辑器上面的菜单栏将会出现facebook选项，选择facebook->editsetting







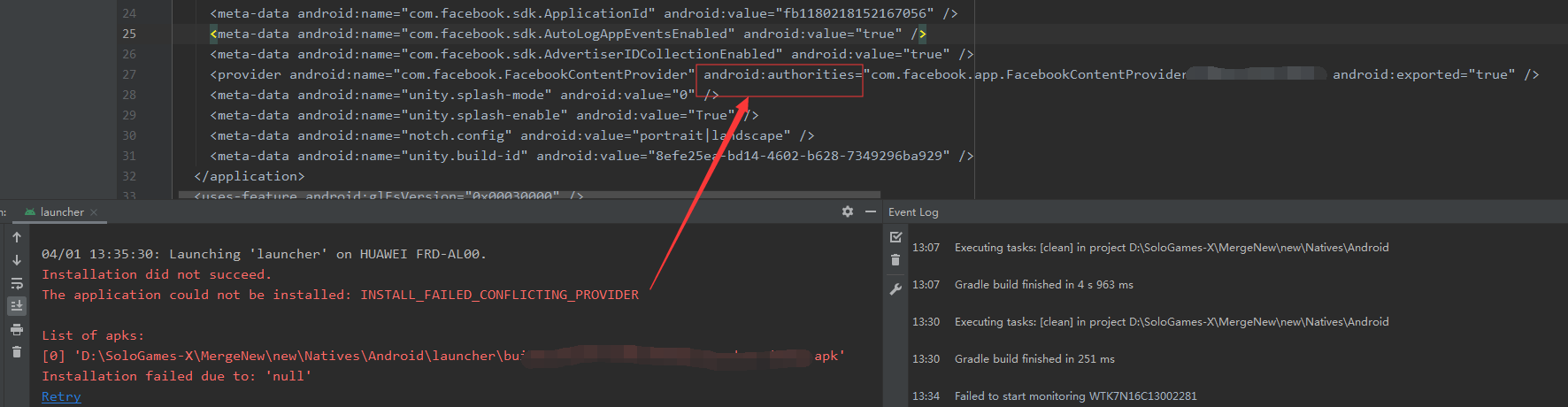
打开配置文件，填入项目名称和项目id，项目id在facebook管理后台获取，然后点击Regenerate Android Manifest按钮，会在项目的AndroidManifest文件中添加facebook的id信息。

在fb开发者后台，android需要额外配置android key hash, unity中列出的key hash是debug

Hash，线上版本需要一个release key hash

最快速获取release key hash的方式是，在手机上面运行android release版本，调用FB.Init()接口后在android studio的运行log里面，fb会提示正确的release key hash，把key hash填入fb开发者后台的android信息中，否则无法正常初始化FB。

关于facebook的id，在android上面，两个不同的apk不可以使用相同的id，否则在往手机上面安装时，会出现无法安装的错误，报错信息如下



注意事项：

FB在调试登录和获取头像，好友等接口时，可以在unity上面直接调试，但是必须是FB开发者账号,并且这个FB账号在我们应用的FB管理后台是管理员或者开发者的身份。

FB的所有接口调用必须使用Graph API的方式，不可以直接使用链接请求数据，否则游戏可能被FB下架。

FB申请到的权限必须在游戏中显式使用，否则游戏可能被FB下架。

调用FB.API之前必须先登录，如果在未登录状态下调用API接口，FB会直接返回错误信息，而不是弹出登录窗口。

FB中的接口除了初始化和登录之外，其它都可以使用FB.API来获取，API的使用说明链接：

<https://developers.facebook.com/docs/unity/reference/current/FB.API>，

调用API的方式被称之为图谱API，使用参数和返回值说明链接：

<https://developers.facebook.com/docs/graph-api/using-graph-api>

FB调试工具界面

https://developers.facebook.com/tools/

## 二 代码部分添加：

1 FB初始化

游戏启动时首先要检查FB的初始化状态，如果FB.IsInitialized为false，则需要调用FB.Init().

FB.Init()是一个异步的过程，可以设置两个回调函数，一个是init完成之后的回调，一个是fb打开html登录界面时游戏进入暂停状态的回调。

FB.Init()在一次游戏运行期间只可以成功调用一次。

if (!FB.IsInitialized)

{

FB.Init(() =>

{

// 初始化成功的回调

Debug.Log("FBInitComplete fb初始化完成= " + FB.IsInitialized);

},

(isGameShow) =>

{

// FB打开html登录界面时游戏进入暂停状态的回调

});

}

else

{

// 如果FB是初始化成功的状态，不需要重复初始化

}

官网关于init接口的说明链接：

https://developers.facebook.com/docs/unity/reference/current/FB.Init

2 FB.ActivateApp

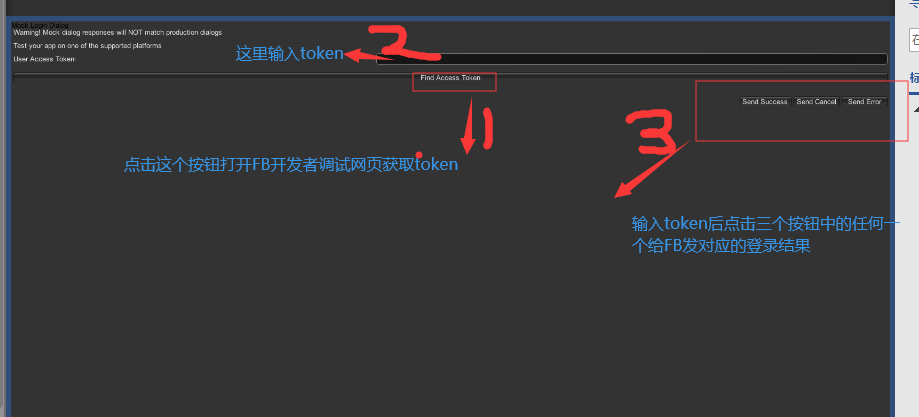
这个接口主要是用来统计玩家激活游戏的次数和游戏时长，可以根据项目需要选用

3.FB登录

Fb登录的前置条件

在项目的FB开发者后台，在设置->基本->右侧页面最下端添加Android和ios平台，ios平台没有上线前可以只添加BundleID，上线后再添加Store编号。如果没有对应的平台，FB登录时采用APP之间跳转会有问题。

在unity 编辑器模式下，调用FB.LogInWithReadPermissions接口可以打开如下的调试界面



这里必须先准备一个FB开发者账号，点击上图的Find Access Token后，会在浏览器中打开FB开发者工具，如果没有登录，则需要先登录账号，登录后打开下面的界面



点击调试打开一个新的界面，会展示完整的token，把token复制下来填入unity调试窗口User Asset Token里面点击Send Success按钮，FB.LogInWithReadPermissions接口设置的回调函数会接口到FB返回的登录成功的状态。点击Send Cancel按钮会返回登录失败，失败原因是用户取消，点击Send Error会返回登录失败的错误信息。

FB初始化成功后才可以调用登录接口。

if (FB.IsInitialized)

{

var perms = new List<string>() { "public\_profile" };

FB.LogInWithReadPermissions(perms, (loginResult) =>

{

bool loginSuccess = true;

// 分析登录返回的数据，判断FB是否成功登录

if (FB.IsLoggedIn)

{

loginSuccess = true;

}

else if (null == loginResult)

{

loginSuccess = false;

}

else if (!string.IsNullOrEmpty(loginResult.Error))

{

Debug.LogError("fb login error details: " + loginResult.Error);

loginSuccess = false;

}

else if (loginResult.Cancelled)

{

loginSuccess = false;

}

else if (!string.IsNullOrEmpty(loginResult.RawResult))

{

loginSuccess = false;

}

else

{

loginSuccess = false;

}

});

}

else

{

// 需要先初始化成功才可以调用登录接口

}

调用FB.LogInWithReadPermissions时需要注意传入的权限参数，"public\_profile"权限不需要申请就可以获取。FB的基本权限中不包含”email”，当FB登录失败时需要留意是否获取了过多的权限。

登录接口的详细说明链接：

<https://developers.facebook.com/docs/unity/reference/current/FB.LogInWithReadPermissions>

FB登录，授权成功后，FB.IsInitialized和token展示的状态都是正确的，目前发现有一种情况例外。

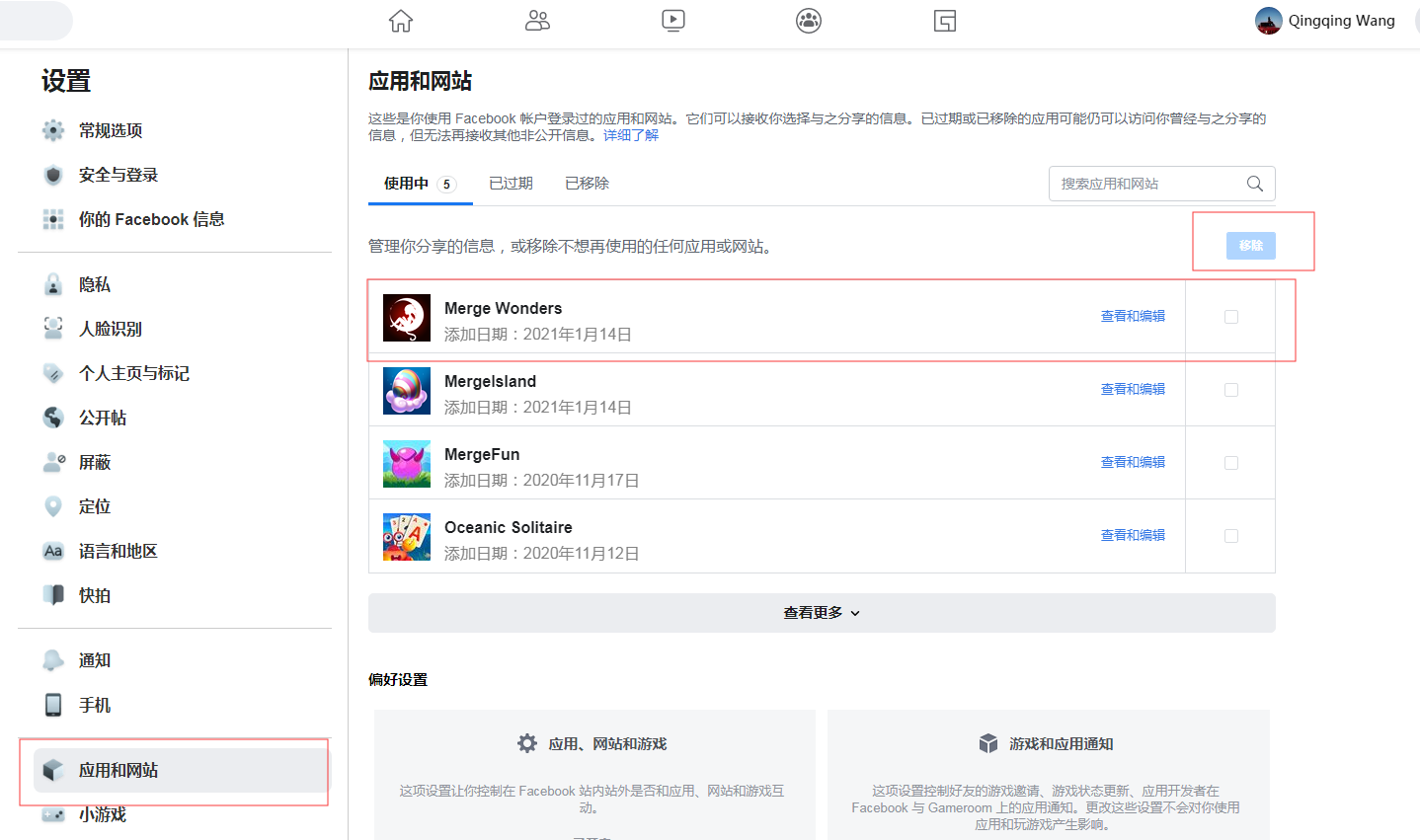
用户在FB的网页端和APP里面可以移除对我们游戏的授权，这个移除授权的操作不会影响我们游戏里面FB的登录状态。如果用户在FB里面移除授权之前 FB.IsInitialized为true的话，用户移除授权之后FB.IsInitialized依然为true，但是此时我们应用已经没有权限，直接调用FB.API的话会返回错误信息。可以在任何的API调用之前先调用登录接口，或者是处理API返回的错误信息。

没有授权的错误信息是：

{"error":{"message":"Error validating access token: The user has not authorized application {id}.","type":"OAuthException","code":190,"error\_subcode":458,"fbtrace\_id":"AnobwmSHJjkt-67Lg\_WZtYx"}}

可以对type和code进行解析。Code的定义链接：

https://developers.facebook.com/docs/marketing-api/error-reference/



4 FB获取用户名和用户id

FB.API("me?fields=id,name", HttpMethod.GET, (result) =>

{

if (result != null && !string.IsNullOrEmpty(result.RawResult))

{

Debug.Log(result.RawResult);

JObject userData = JObject.Parse(result.RawResult);

string id = (string)userData["id"];

string name = (string)userData["name"];

}

else

{

}

});

5 FB获取头像

FB.API("/me/picture?type=large", HttpMethod.GET, (result) =>

{

if (null != result && string.IsNullOrEmpty(result.Error) && result.Texture != null)

{

//保存在本地

byte[] pngData = result.Texture.EncodeToPNG();

string pngPath = AccessToken.CurrentAccessToken.UserId + ".png";

if (File.Exists(pngPath))

{

File.Delete(pngPath);

}

File.WriteAllBytes(pngPath, pngData);

Sprite pngSprite = Sprite.Create(result.Texture, new Rect(0, 0, result.Texture.width, result.Texture.height), new Vector2(0, 0));

}

});

7 FB获取玩同一个游戏的好友列表

在开发阶段获取好友信息的方式

7.1 确保所开发的项目在FB后台时开发中的状态

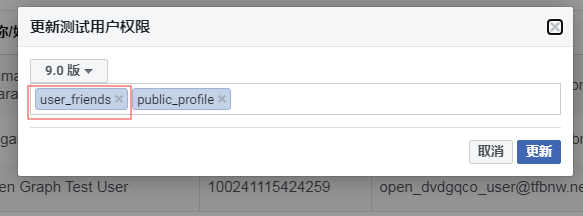


7.2 给FB测试账号添加权限和好友

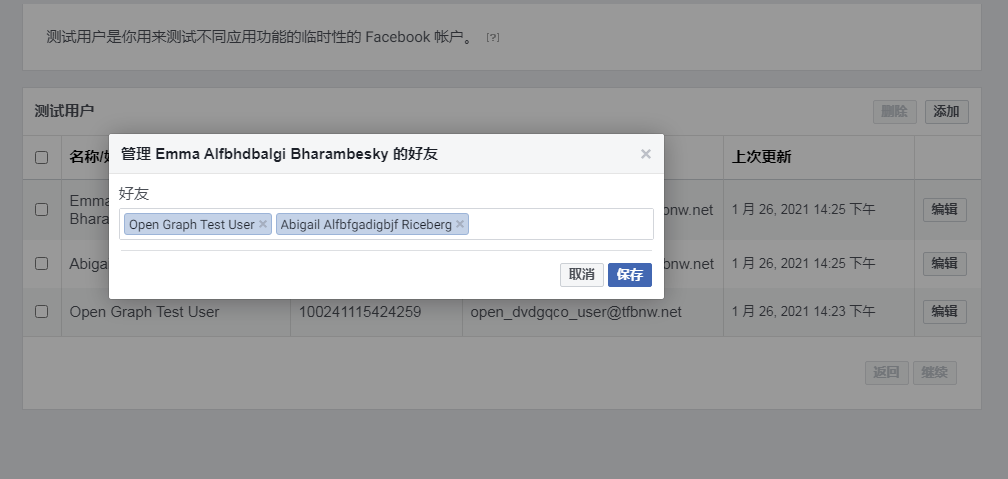
在开发项目中的FB好友权限没有申请下来之前，是不可以直接用自己的账号来获取好友的，这是需要借助FB账户内的测试账号



下图是添加User\_friends权限，这个账号添加的所有测试好友账号都需要加上user\_friends权限，否则无法获取对应的账号内的详细信息。



下图把另外两个测试账号添加为好友



7.3 添加代码获取好友信息

调用好友接口先需要刚刚设置好的测试账号登录，不能使用自己的账号

FB.API("me/friends?fields=id,name,picture", HttpMethod.GET, (result) =>

{

if (result.Error != null)

{

Debug.Log("friends " + result.Error);

}

if (result != null && !string.IsNullOrEmpty(result.RawResult))

{

Debug.Log(result.RawResult);

JObject dict = JObject.Parse(result.RawResult);

if (null != dict["data"])

{

JArray friendsArray = (JArray)dict["data"];

for (int i = 0; i < friendsArray.Count; i++)

{

string fbID = (string)friendsArray[i]["id"];

string fbName = (string)friendsArray[i]["name"];

JObject picData = (JObject)friendsArray[i]["picture"];

JObject dataInfo = (JObject)picData["data"];

string fbPicPath = (string)dataInfo["url"];

Debug.Log("friend Name " + fbName);

}

}

}

else

{

Debug.Log("get friend null");

}

});