appsflyer接入文档

# 

目录

1.[sdk下载](#sdk下载)**2**

2. [sdk初始化](#sdk初始化)**3**

3. [归因数据回调](#归因数据回调)**4**

4. [打点上传](#打点上传)**4**

5. [订单校验](#订单校验)**4**

6.[onelink深层连接和延迟深层连接](#onelink深层连接和延迟深层连接)**5**

6.1[运营后台运营配置](#运营后台运营配置)5

6.2[客户端代码修改](#客户端代码修改)8

6.2.1[链接概念](#链接概念)8

6.2.2[安卓修改](#安卓修改)8

6.2.3 [IOS修改](#ios修改)9

7.卸载回传**11**

7.1[卸载回传运营配置](#运营配置)11

7.1.1[安卓运营后台配置](#安卓运营后台配置) 11

7.1.2 [ios运营后台配置](#ios运营后台配置) 11

7.2[卸载回传客户端配置](#客户端配置)15

DemoCs文件



unity版本：2019.4.7

appsflyer unity：firebase\_unity\_sdk\_7.0.2

git：https://github.com/DarkYuanJs/SDKTemplate.git

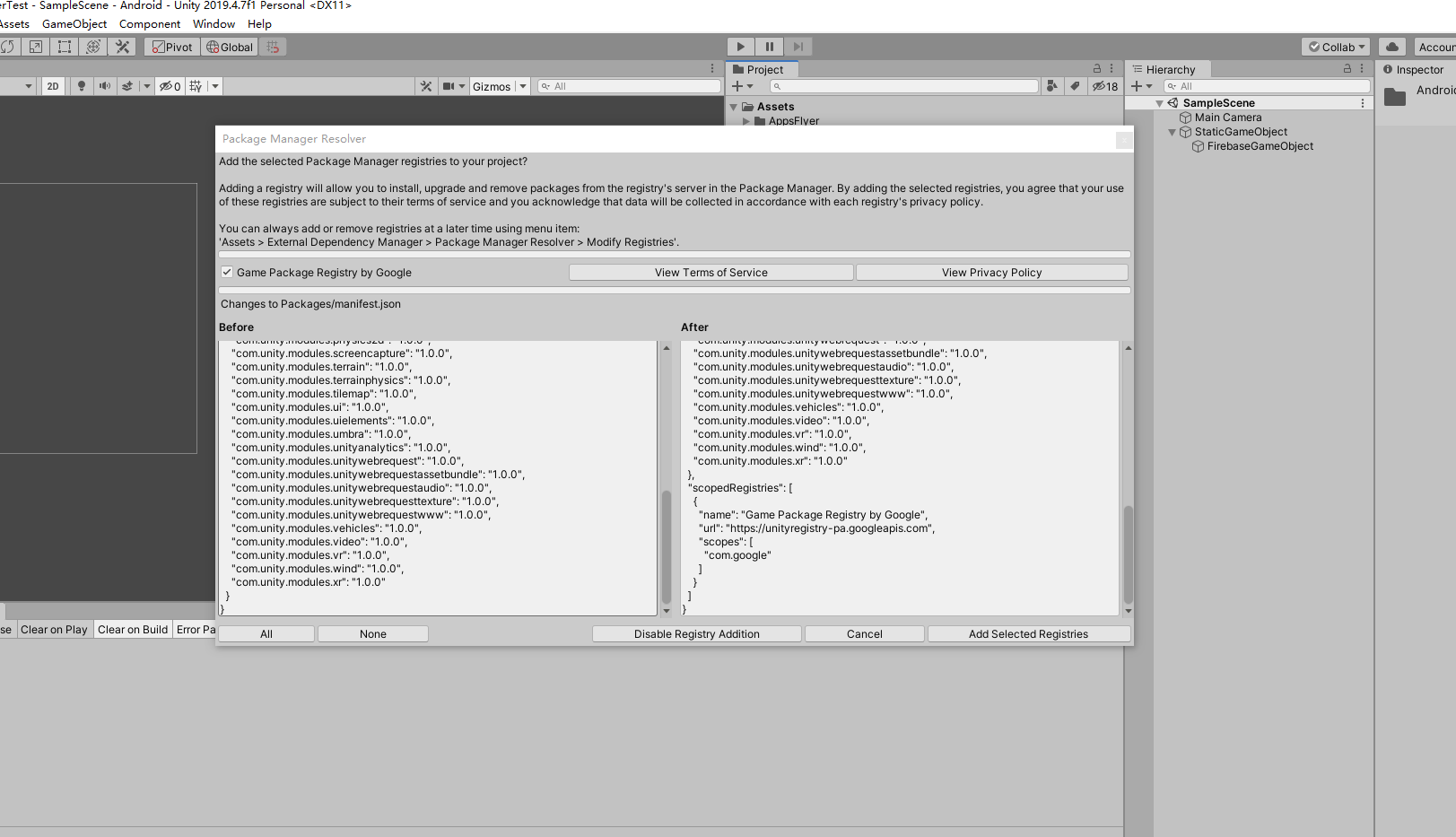
1.sdk下载

1.下载appsflyer的sdk，当前我下载的是appsflyer-unity-plugin-6.1.3

<https://github.com/AppsFlyerSDK/appsflyer-unity-plugin>

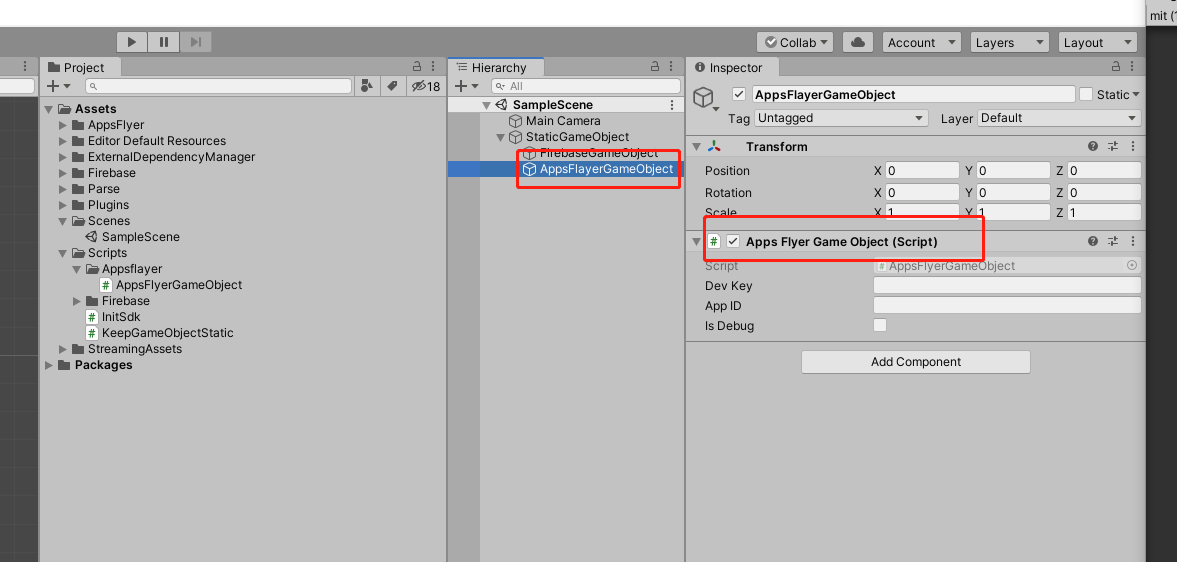
导入appsflyer-unity-plugin-6.1.3.unitypackage 到工程

点击all就不会在出现了，这个是一个google的一个服务



2. sdk初始化

直接复制一份儿appsflyer目录下的AppsFlyerObjectScript文件内容改下名字挂载到一个空的gameobject上，这样的目的是如果直接在官方里面修改，升级版本就会冲掉。这样也方便自己管理

代码初始化的方法(this代表的是继承IAppsFlyerConversionData的回调函数所在的MonoBehaviour，也就是上面我们copy下来的AppsFlyerObjectScript文件）

AppsFlyer.setIsDebug(isDebug);

AppsFlyer.initSDK(devKey, appID, this);

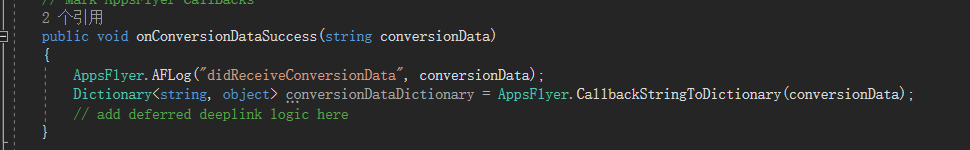
AppsFlyer.startSDK();

AppsFlyer.setCustomerUserId(UID); // 根据自己需要设置af的UID（非必须）

3.归因数据回调

游戏初始化成功后会自动回调onConversionDataSuccess

这个地方着重注意下这个object里面会有null，所以使用的时候需要进行下非空判断



4.打点上传

System.Collections.Generic.Dictionary<string, string> purchaseEvent = new

System.Collections.Generic.Dictionary<string, string> ();

purchaseEvent.Add(AFInAppEvents.CURRENCY, "EUR");

purchaseEvent.Add(AFInAppEvents.REVENUE, "200.12");

purchaseEvent.Add(AFInAppEvents.QUANTITY, "1");

purchaseEvent.Add(AFInAppEvents.CONTENT\_TYPE, "category\_a",);

AppsFlyer.sendEvent ("af\_purchase", purchaseEvent);

5．订单校验

下面的初始化函数中的(this代表的是继承IAppsFlyerConversionData的回调函数所在的MonoBehaviour）

#if UNITY\_IOS

AppsFlyeriOS.validateAndSendInAppPurchase(productID, price.ToString(), priceUnitType, transactionId, extraParams, this);

#elif UNITY\_ANDROID

AppsFlyerAndroid.validateAndSendInAppPurchase(PurchaseManager.Instance.\_androidPublicTransactionKey, signatureOnAndroid, purchaseData, price.ToString(), priceUnitType, extraParams, this);

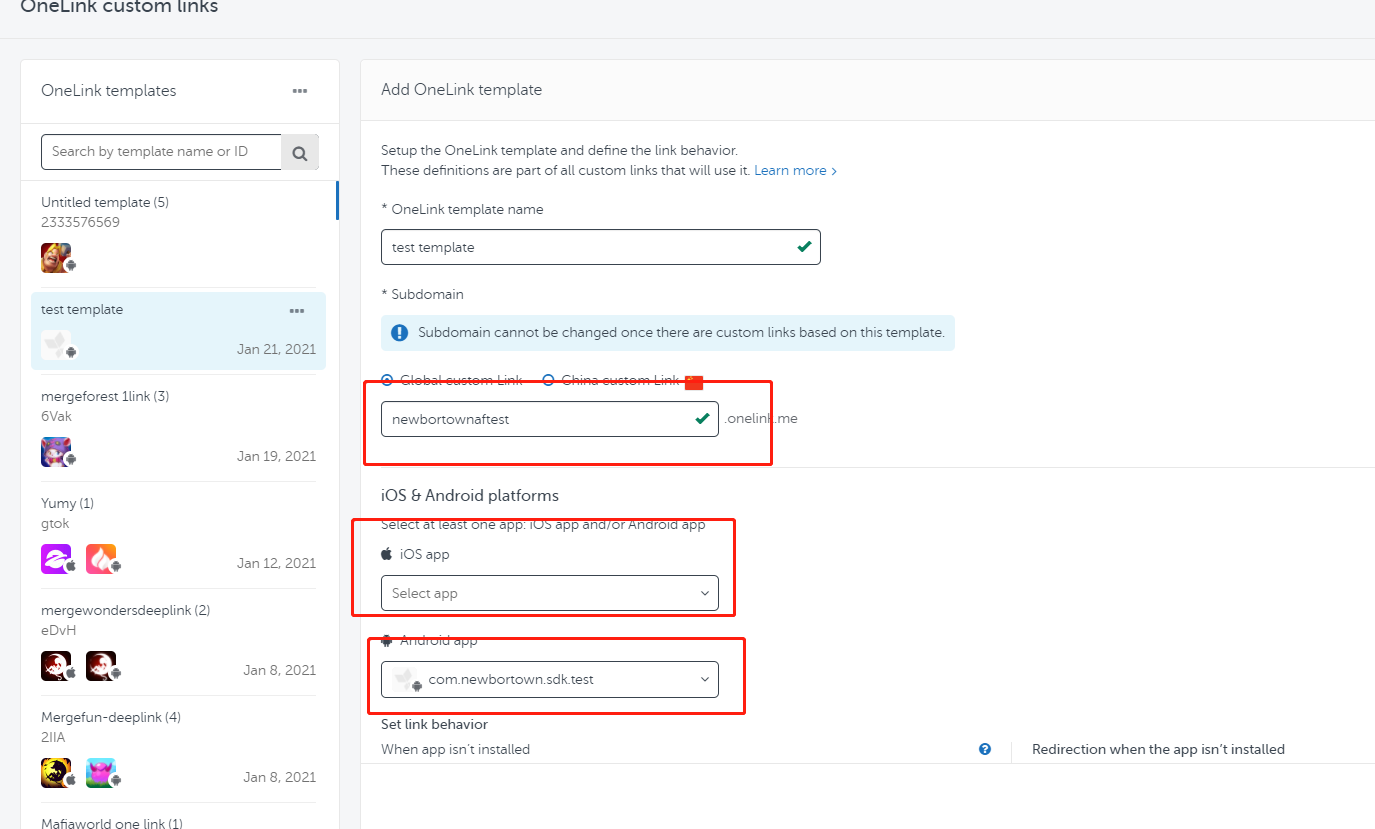
#endif

6.onelink深层连接和延迟深层连接

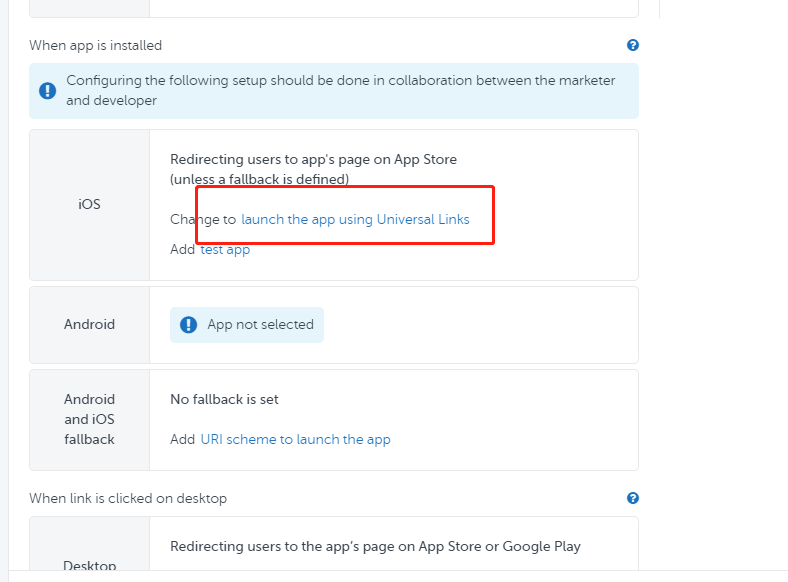
* 6.1运营后台运营配置（此步骤是运营配置，但是需要客户端来配合）

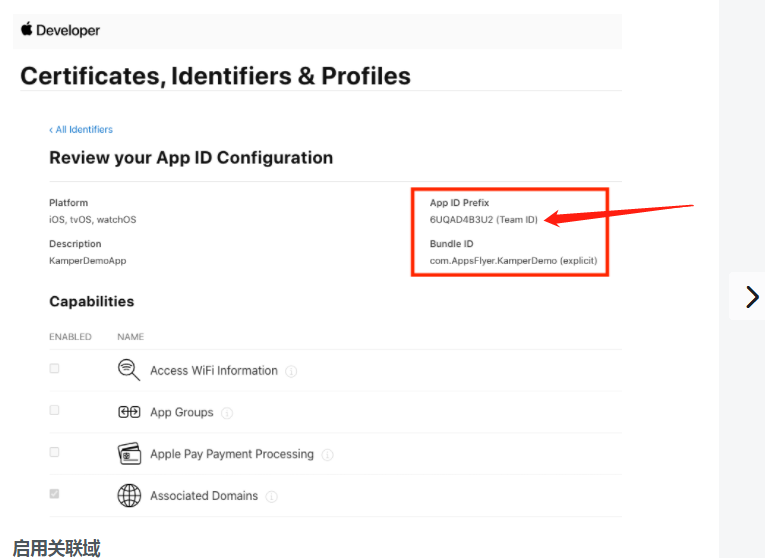
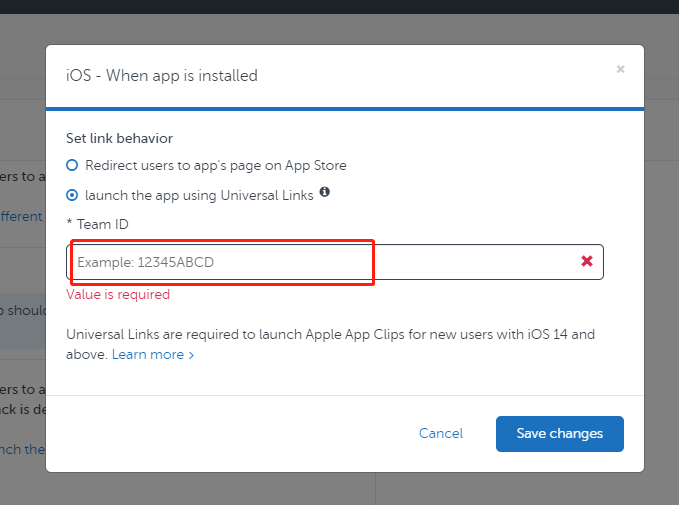
AF后台->onelink custom Links->Add OneLink template

同学让填写Subdomain的名字，并选好要跳转的包名



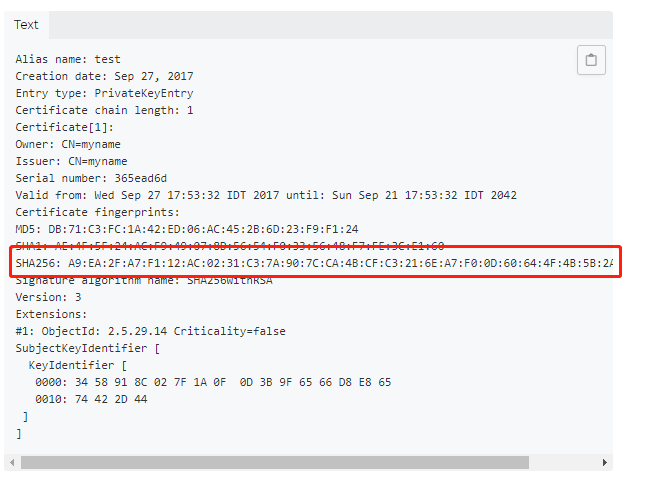
ios配置 Universal Links 这个地方填写的是苹果后台的TeamId

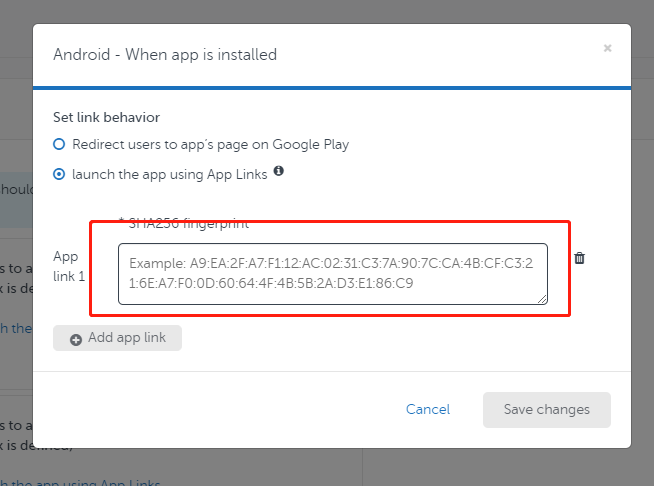
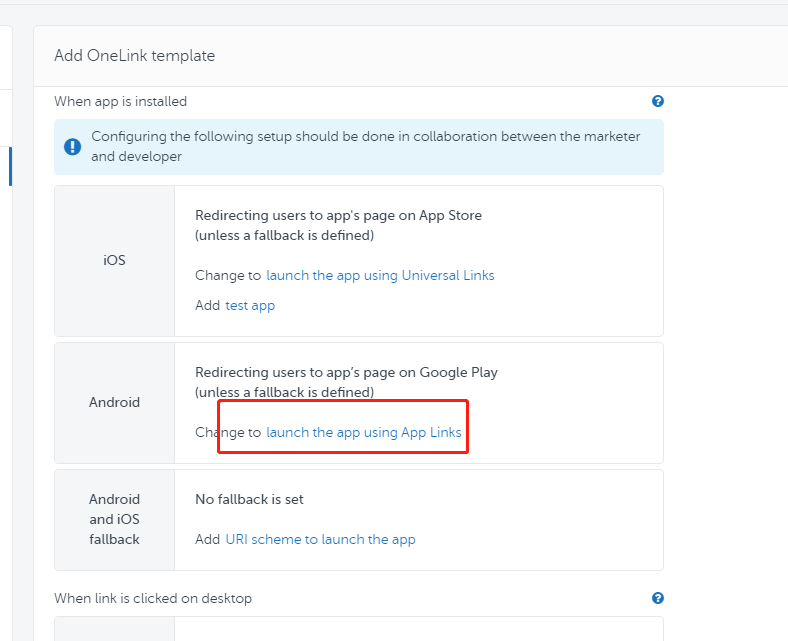




android配置SHA256 找到自己的打包证书

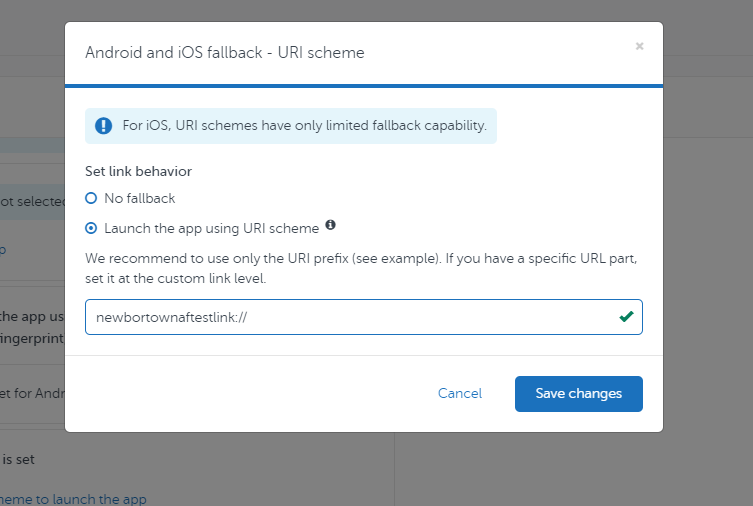
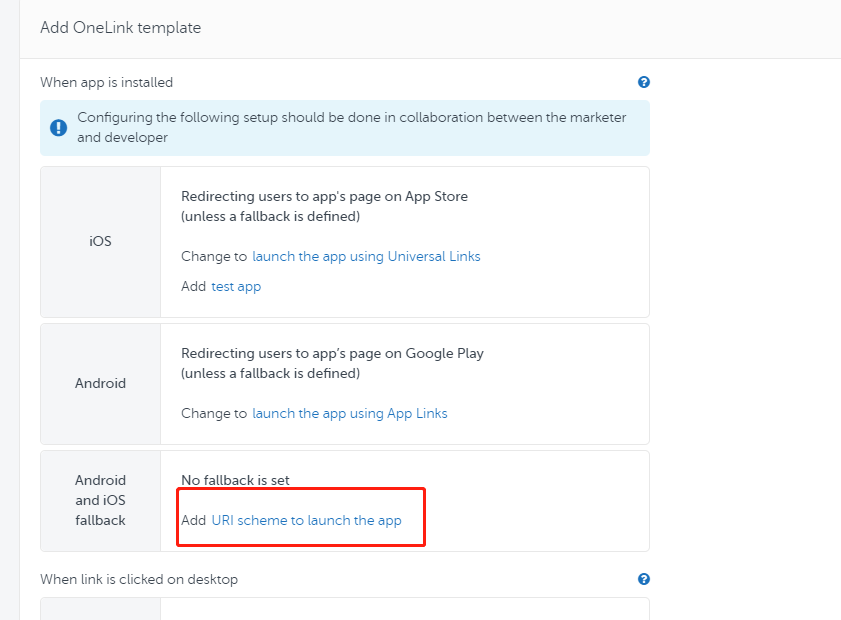
keytool -list -v -keystore [APK-KEY].keystore



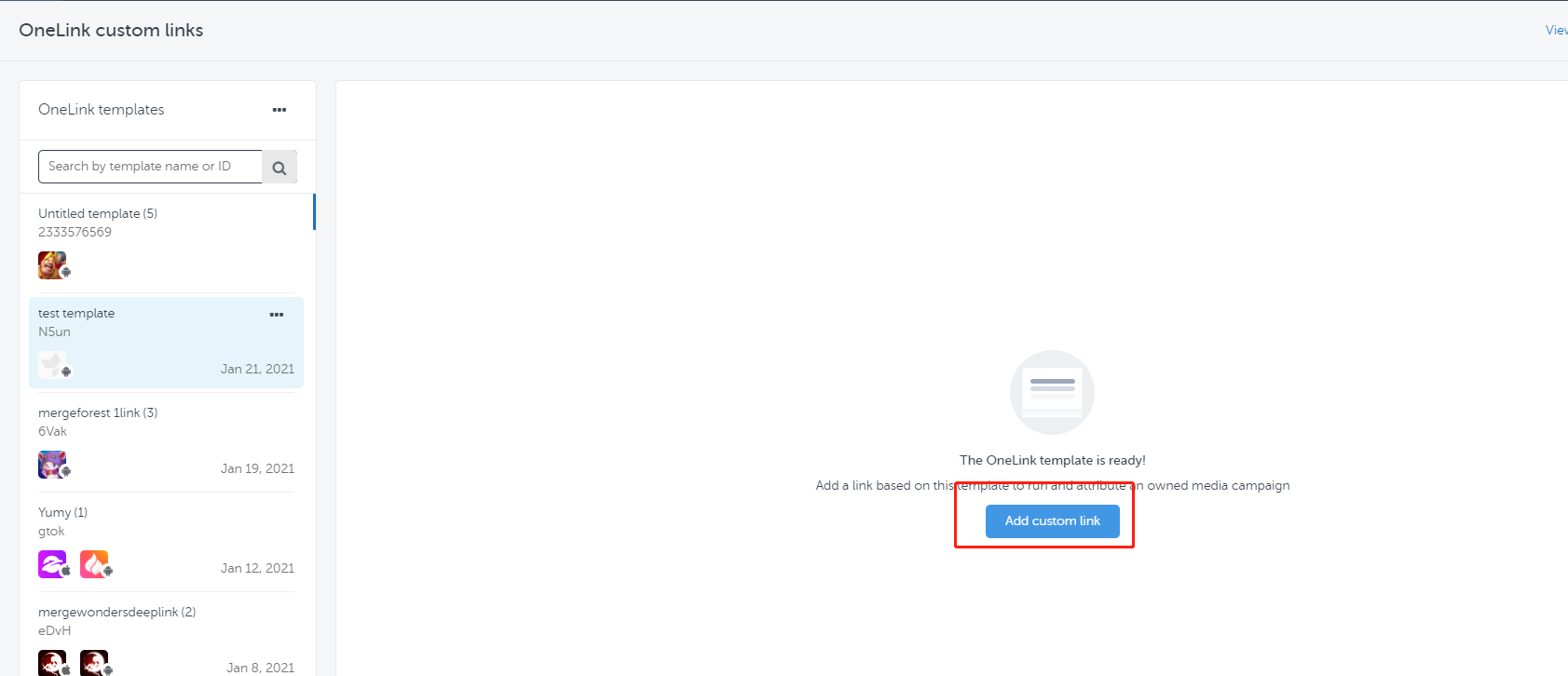


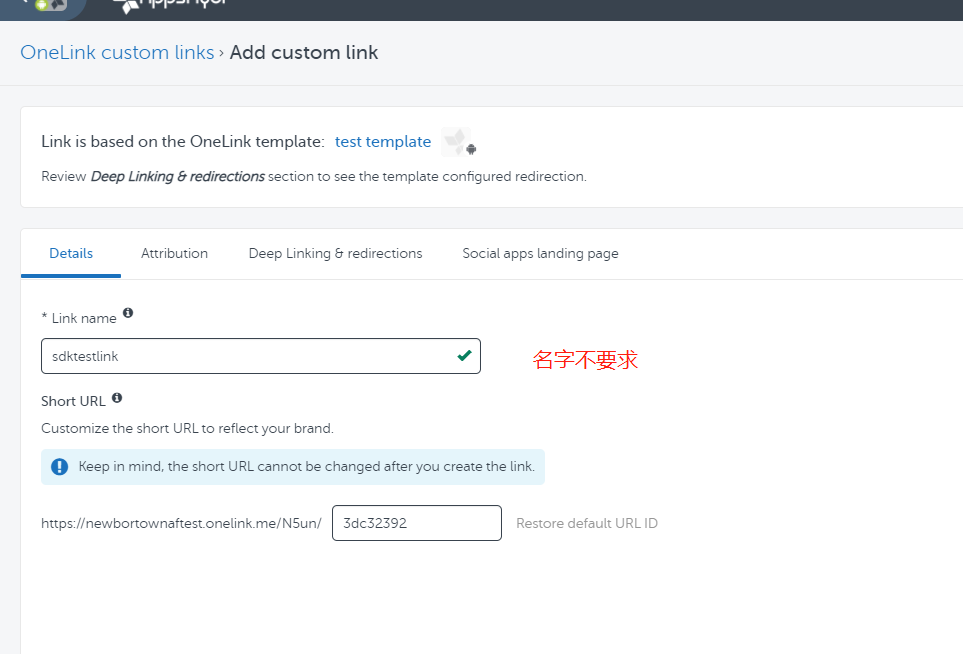
最后填写scheme名字 ，shceme相当于一个唯一的域名一般用包名+link+://(例mergefunlink://)

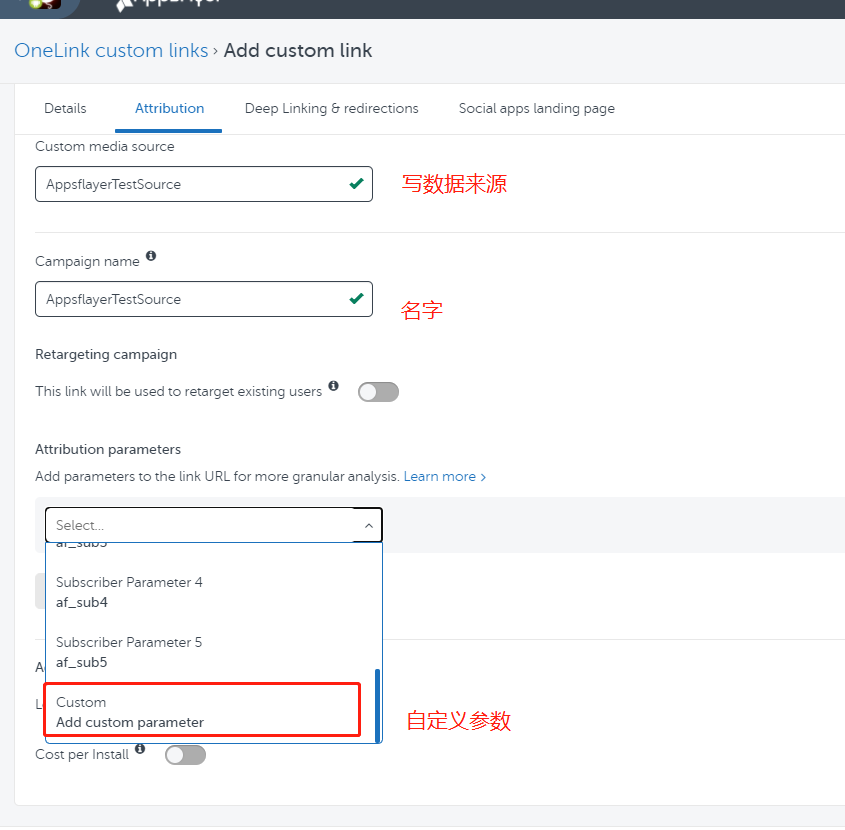
这个域名的作用就是当上面的安卓和ios配置的那种唤起引用方式不管用的时候可以使用通用的scheme唤起，起保险作用

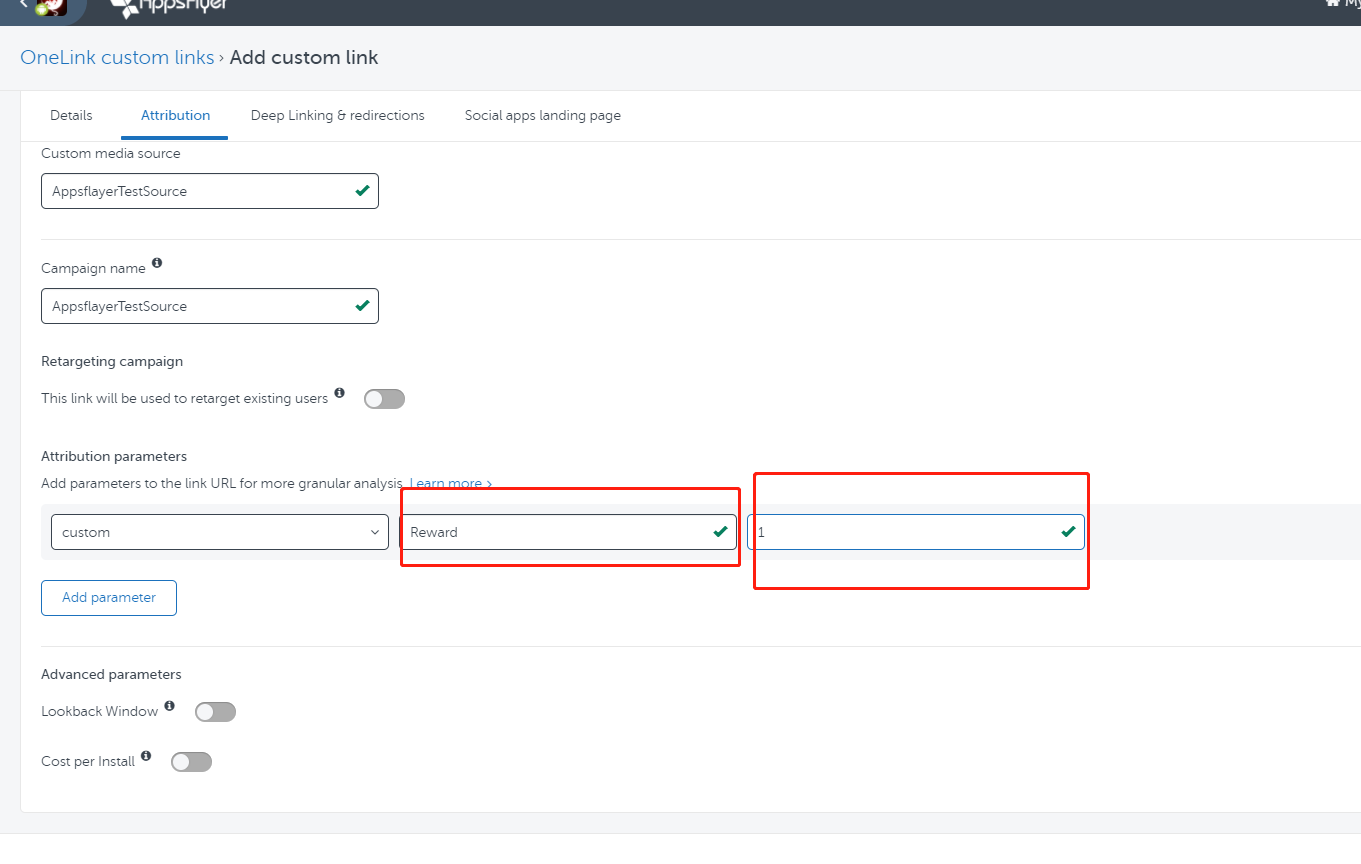


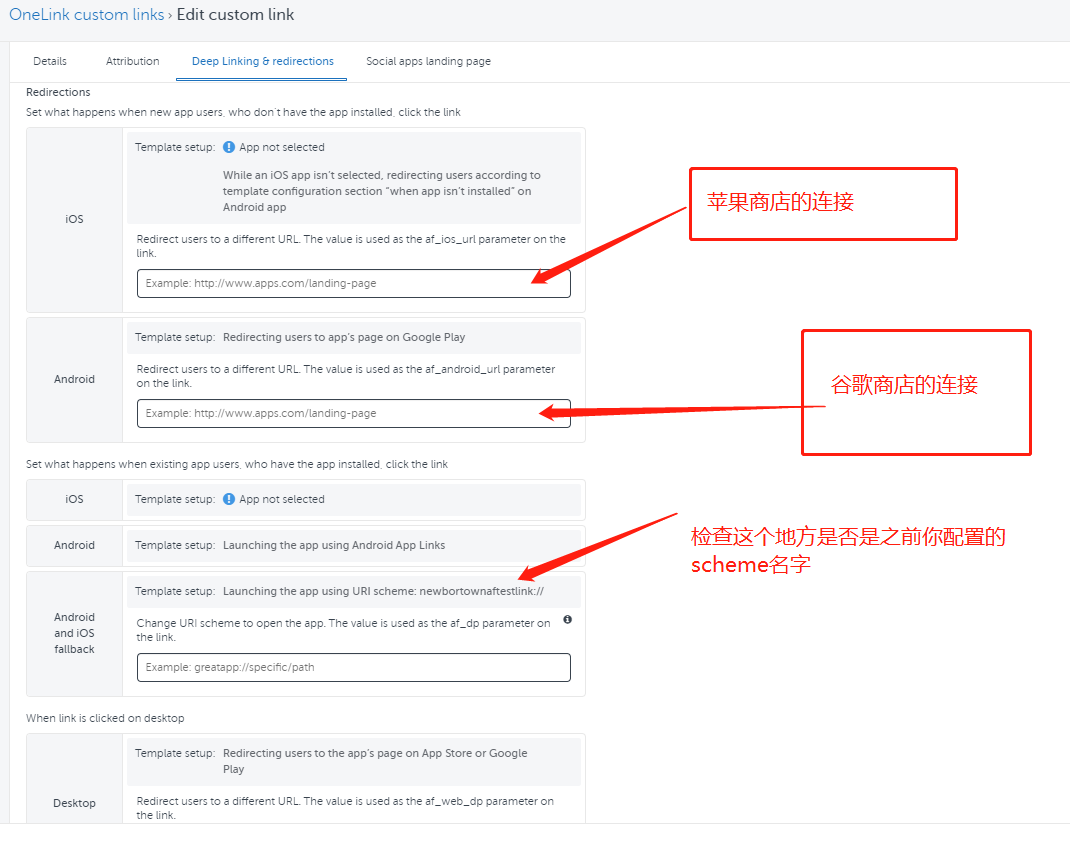
点击刚刚生成好的模板来创建custom link

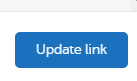










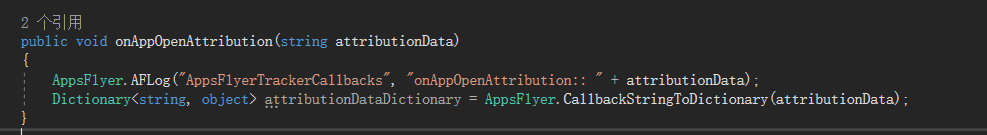
最后点击

最后会出现一个短链一个长链（包含刚刚我填写的自定义字段Reward）,默认我们使用短链，发布到推广平台上使用

短链：https://newbortownaftest.onelink.me/N5un/3dc32392

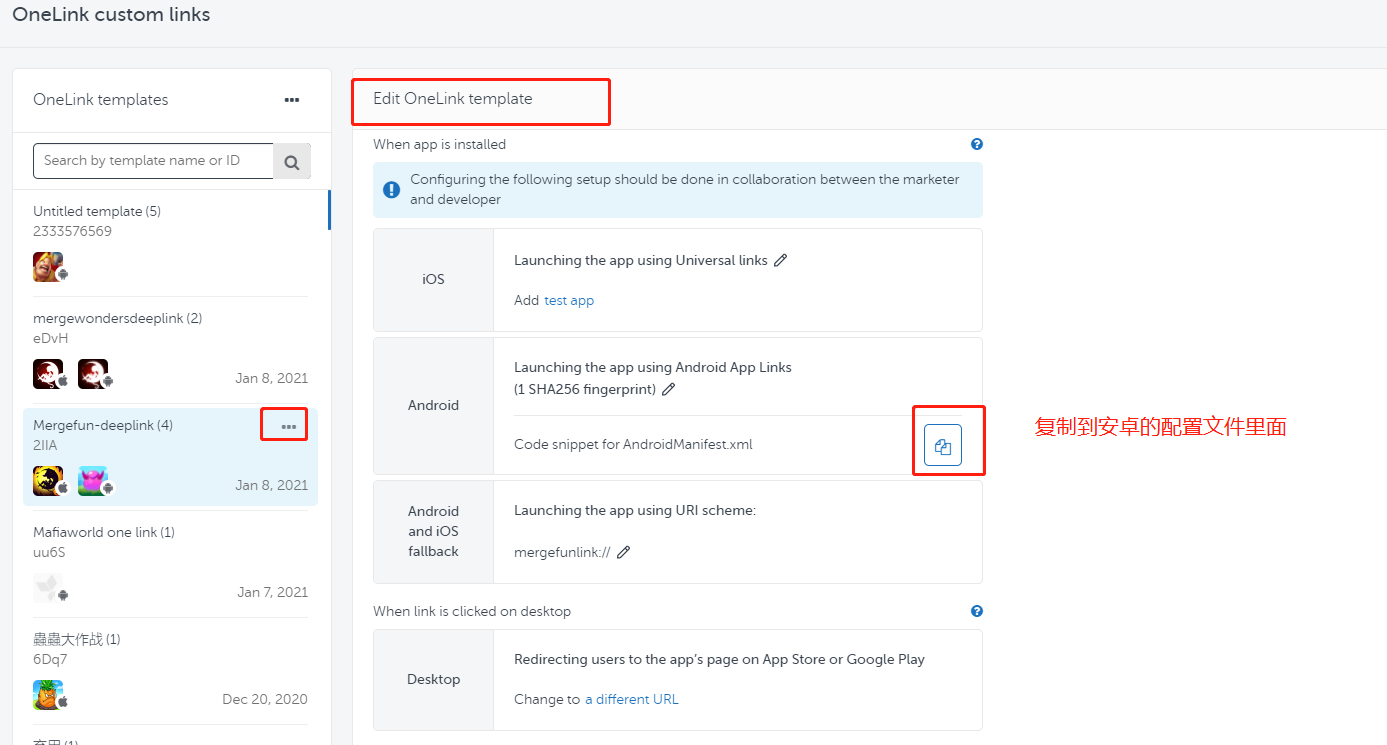
长链：https://newbortownaftest.onelink.me/N5un?pid=AppsflayerTestSource&c=AppsflayerTestSource&Reward=1&af\_dp=newbortownaftestlink%3A%2F%2F

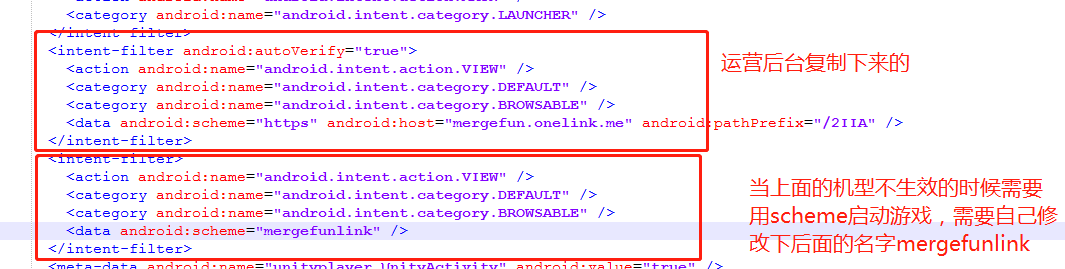
* 6.2客户端代码修改
* 6.2.1 链接概念
* 深层连接在继承IAppsFlyerConversionData 的文件中的onAppOpenAttribution方法回调里面用户是在已经安装过游戏的用户可以直接回调到该方法里面，自定义参数里面会包含在这个数据里面



* 延迟深度连接是用户没有安装游戏，在点击过af的推广链接后下载游戏在回调函数的onConversionDataSuccess中会有数据展现自定义参数里面会包含在这个数据里面
* 6.2.2 安卓修改

AndroidManifest.xml 修改 运营提供一个唯一的模板信息

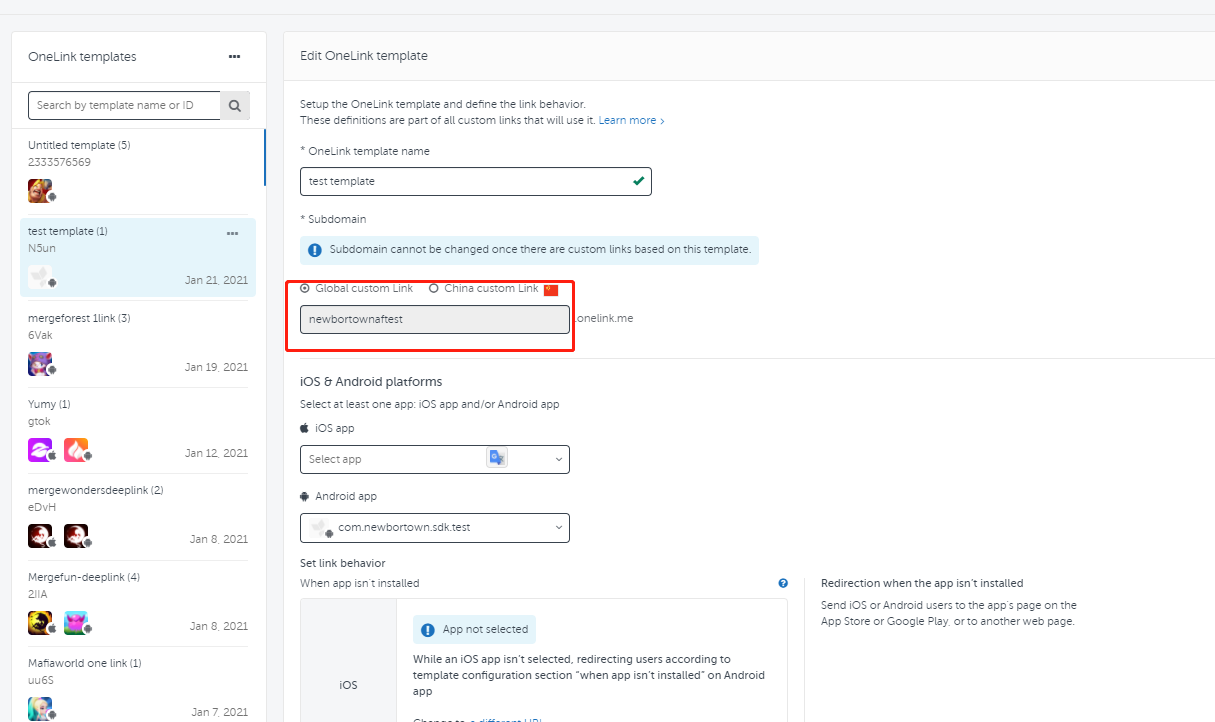


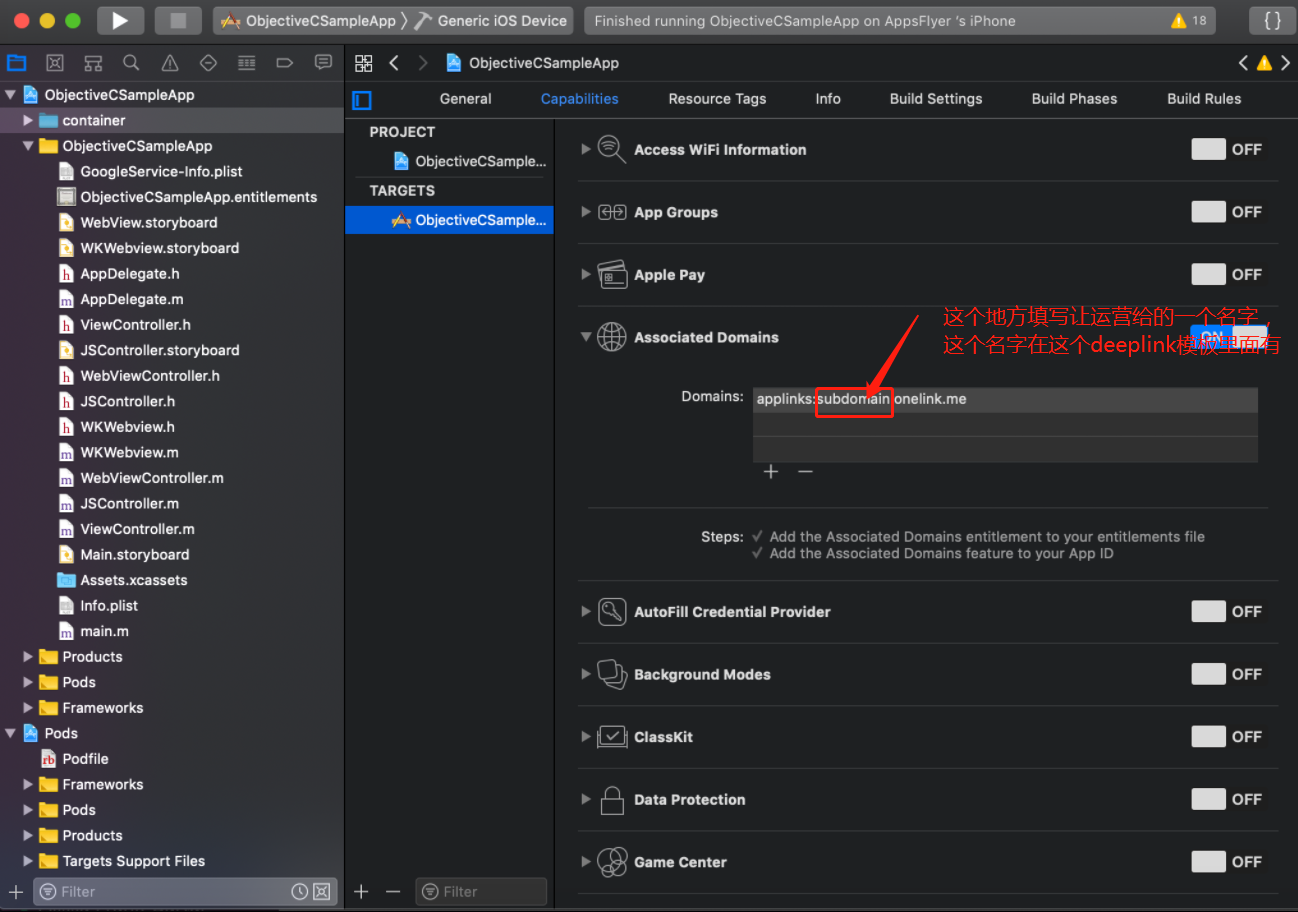


<**intent-filter android:autoVerify="true"**>  
 <**action android:name="android.intent.action.VIEW"** />  
 <**category android:name="android.intent.category.DEFAULT"** />  
 <**category android:name="android.intent.category.BROWSABLE"** />  
 <**data android:scheme="https"  
 android:host="newbortownaftest.onelink.me"  
 android:pathPrefix="/N5un"** />  
</**intent-filter**>  
<**intent-filter**>  
 <**action android:name="android.intent.action.VIEW"** />  
 <**category android:name="android.intent.category.DEFAULT"** />  
 <**category android:name="android.intent.category.BROWSABLE"** />  
 <**data android:scheme="newbortownaftestlink"** />  
</**intent-filter**>

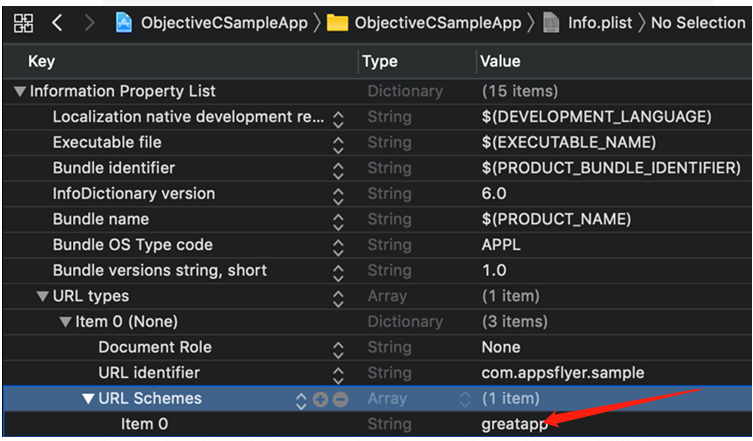
* 6.2.3 ios修改

让运营把subdomain名字给到客户端配置下





把scheme填写到URL types是item0的schemes里最后面添加一个上面定义好的scheme名字

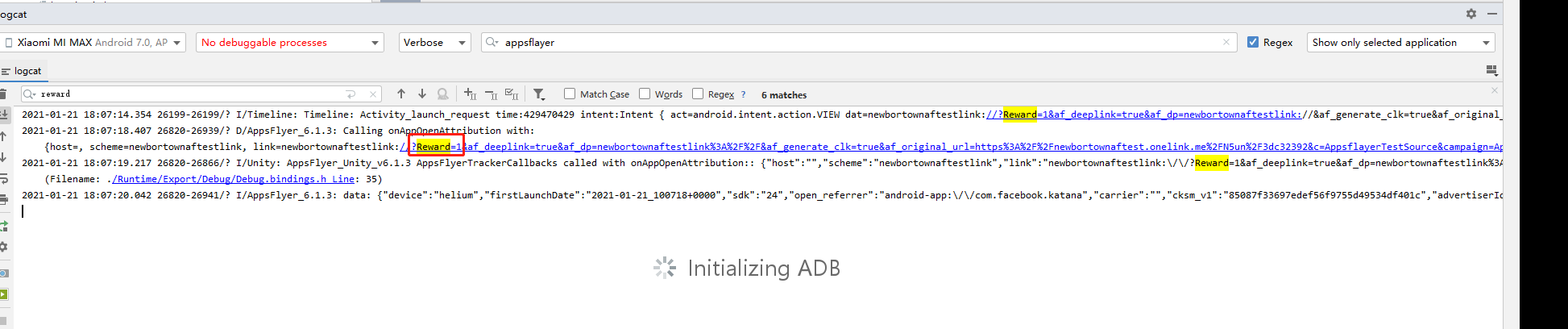


接入完毕，开始测试

url不支持浏览器 qq 微信等引用可以用af的官方<http://pusubj.com>测试



最后点击连接会直接启动游戏，Reward参数就可以看到了



7.卸载回传

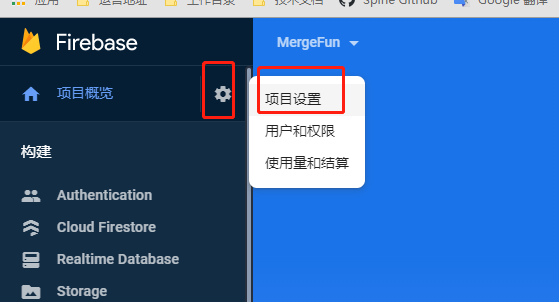
卸载回传是当安装包在平台上卸载后可以在后台检测到该用户已经卸载了使用的是firebase的FCM

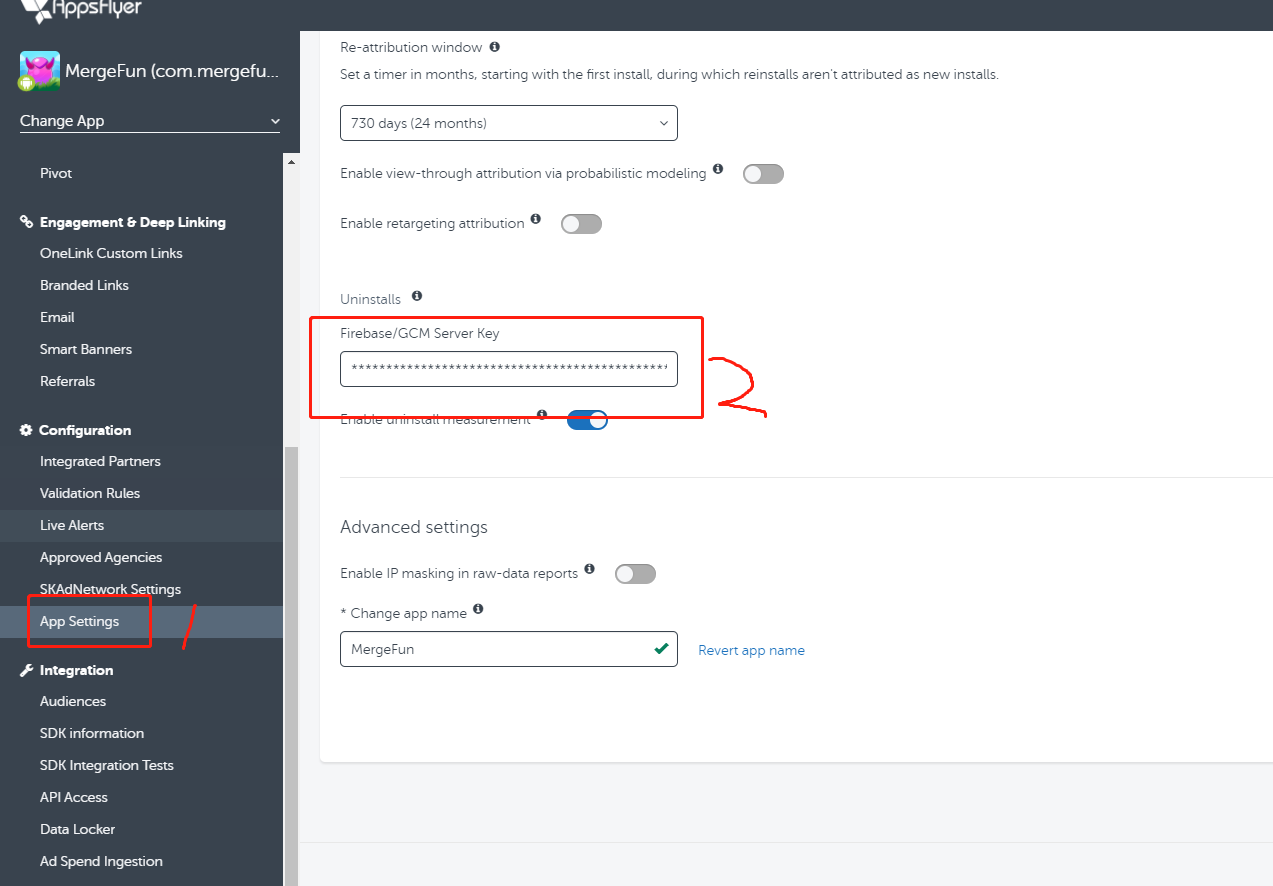
Af的卸载回传需要依赖于firebase的FCM，最新版本直接安装firebase的FirebaseMessaging.unitypackage

7.1卸载回传运营配置

7.1.1 安卓运营后台配置

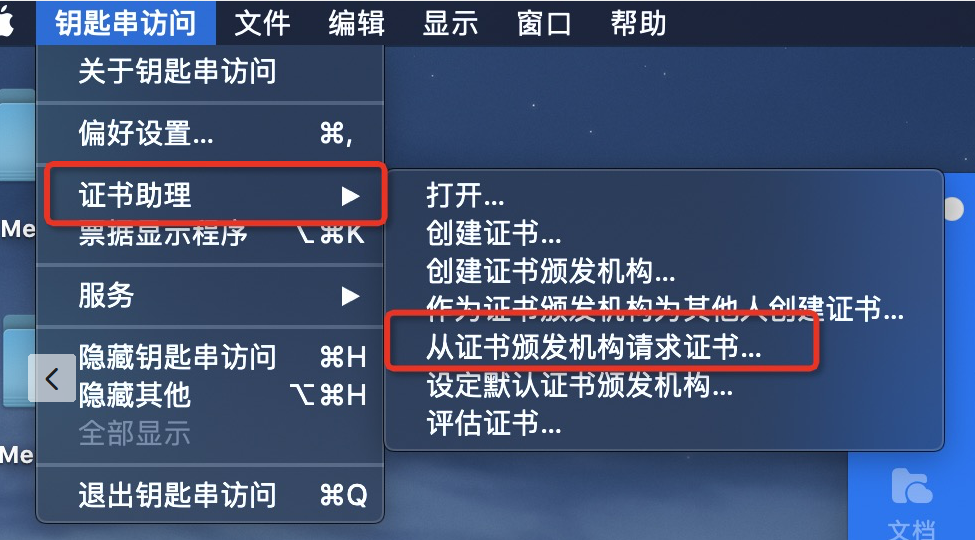
打开[Firebase console](https://console.firebase.google.com/" \t "_blank)控制台后台，把firebase的服务器秘钥填写到af的安卓包后台

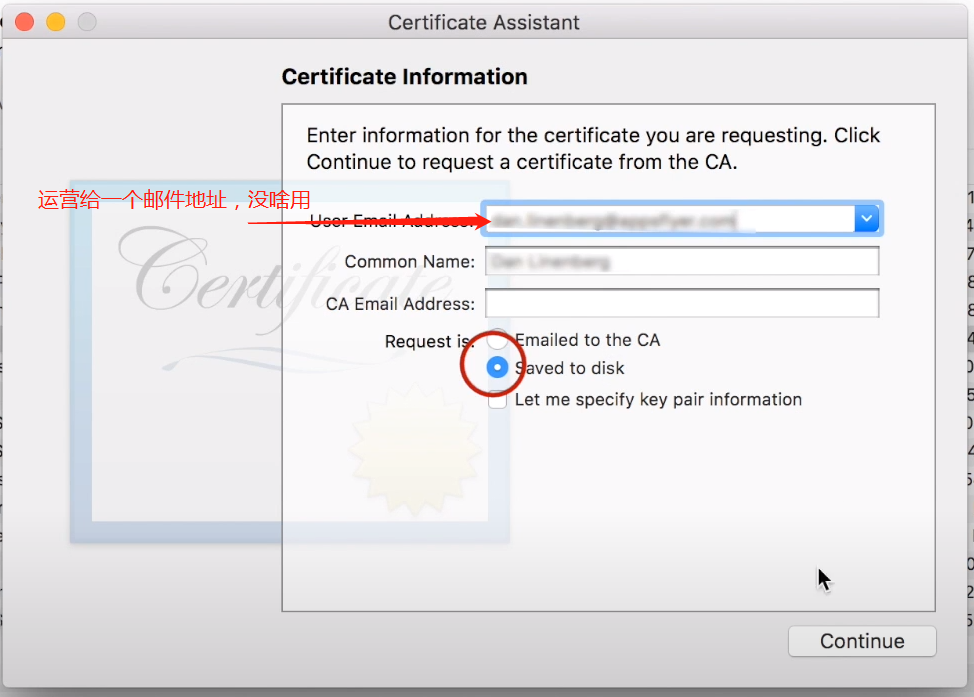


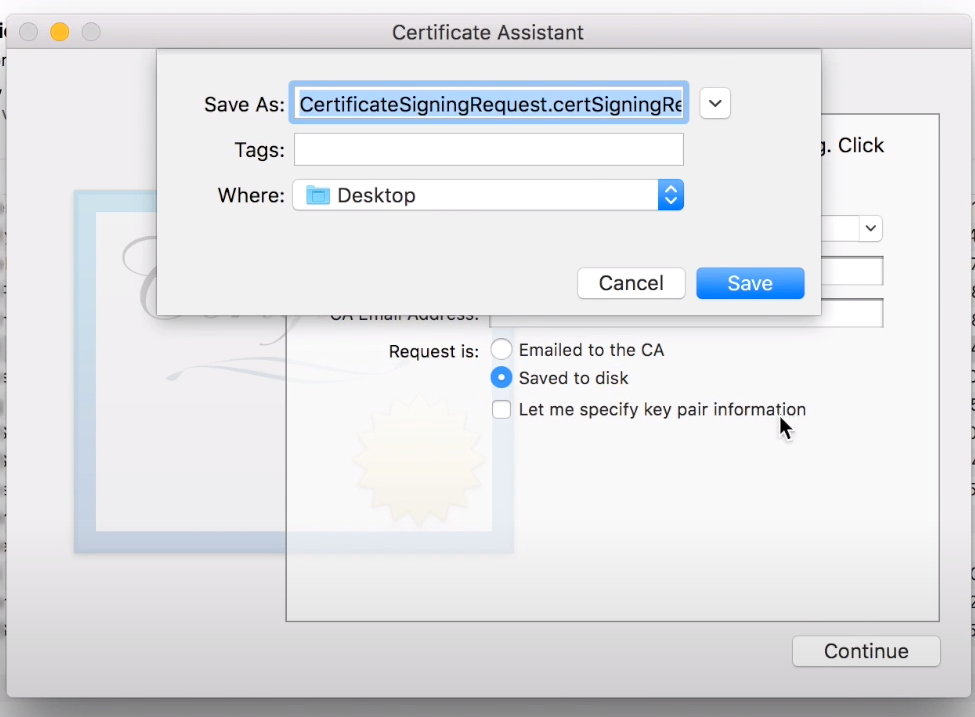


7.1.2 ios运营后台配置

客户端先在xcode上生成一个类似于证书的东西给到运营同学，后台配置要用

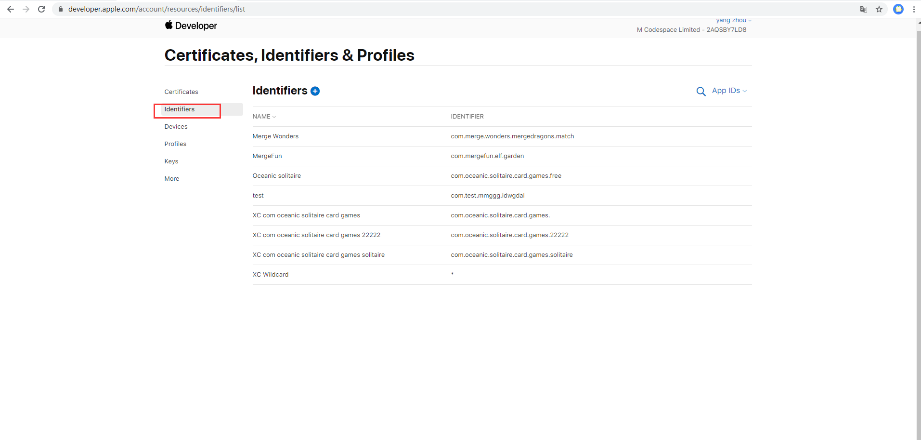


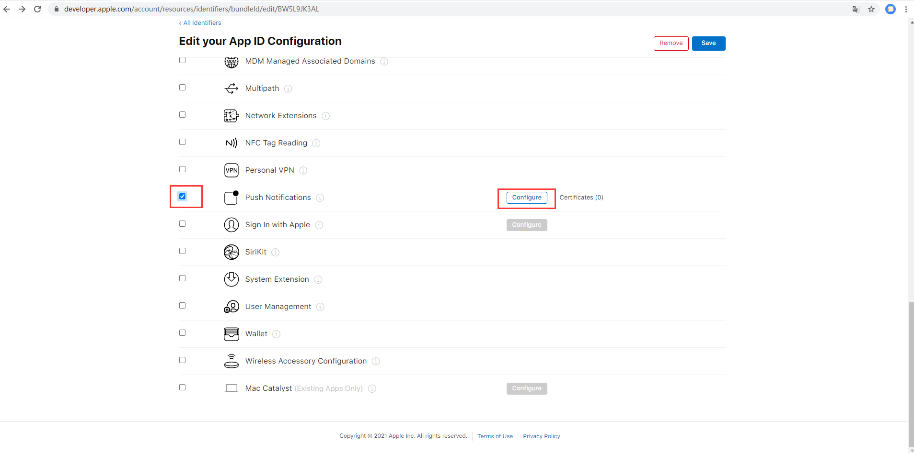




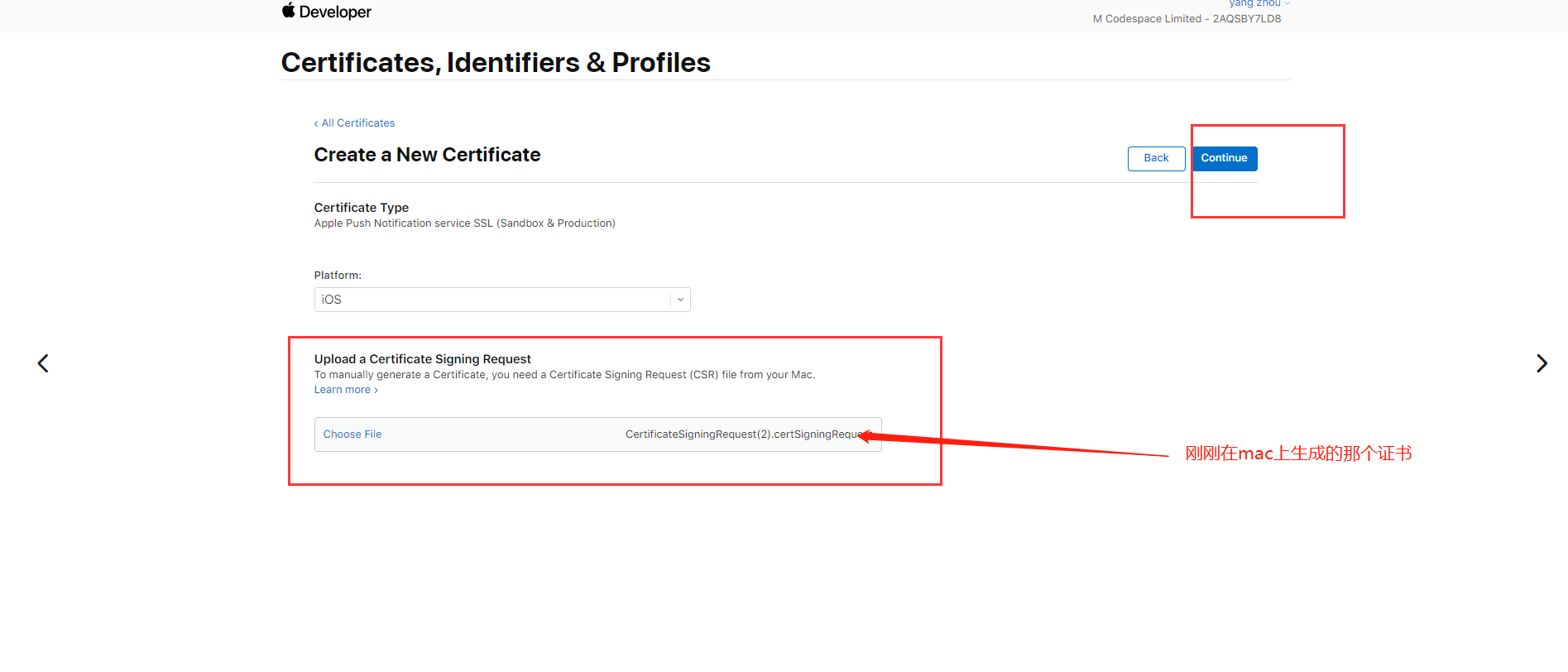
先打开苹果官网后台

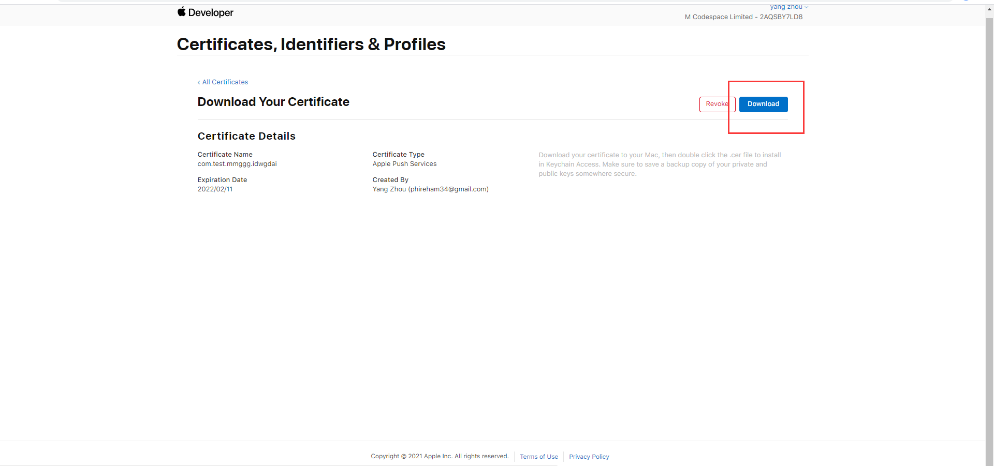
点击要增加的应用或者游戏



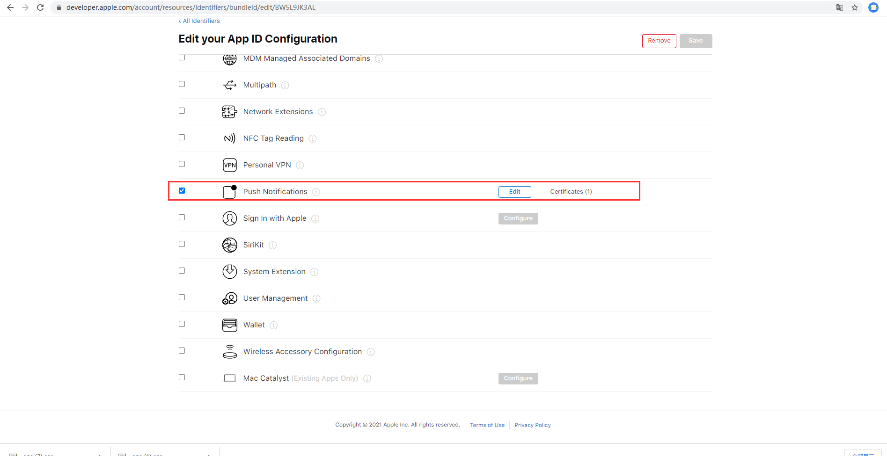




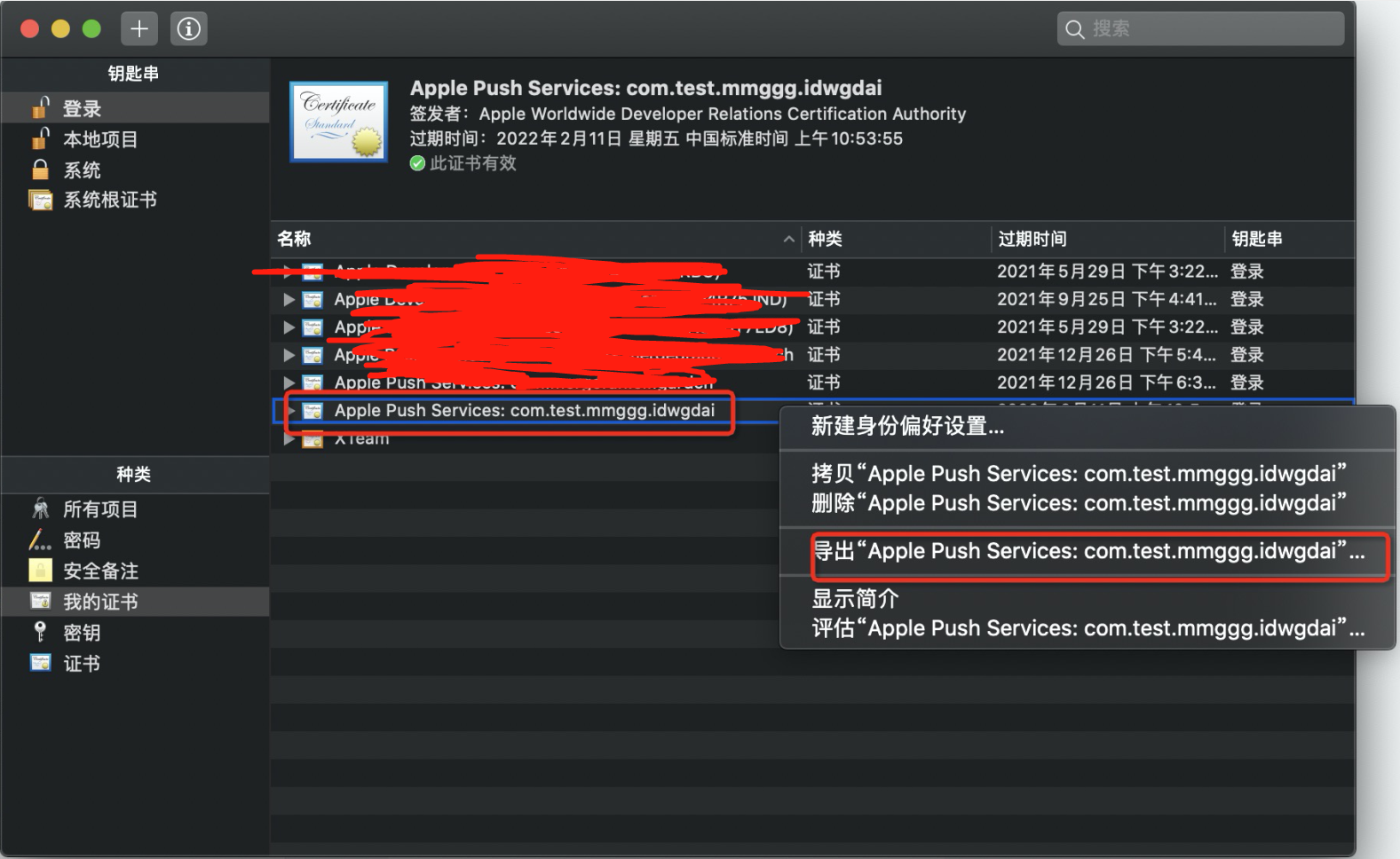


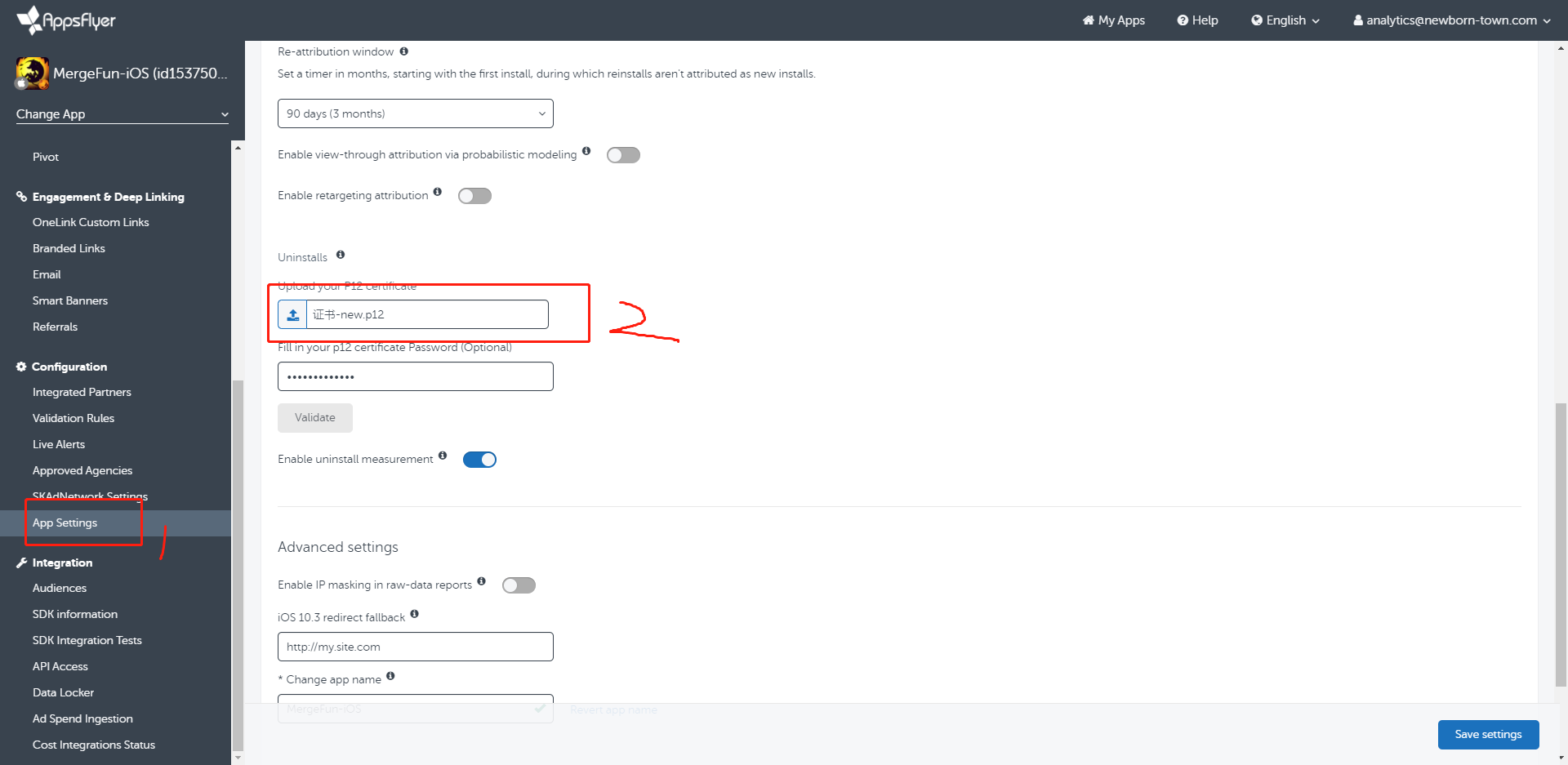


下图是有通知证书的样子检查下



把证书下载到本地，然后双击安装证书，导出p12上传到后台，





7.2 卸载回传客户端配置

导入FirebaseMessaging.unitypackage的package到项目中,firebasemessage接入请看firebase的接入文档，在firebase的初始化文件中添加

Firebase.Messaging.FirebaseMessaging.TokenReceived += OnTokenReceived;

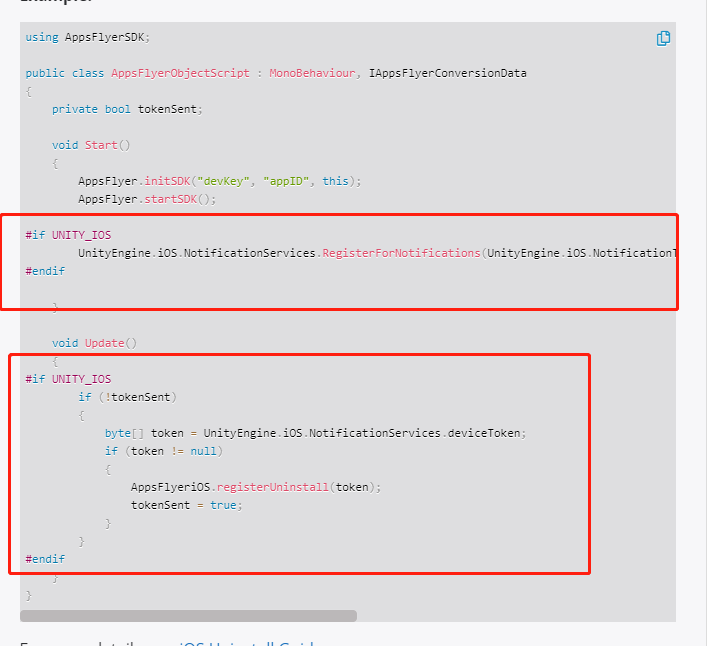
public void OnTokenReceived(object sender, Firebase.Messaging.TokenReceivedEventArgs token) {

#if UNITY\_ANDROID

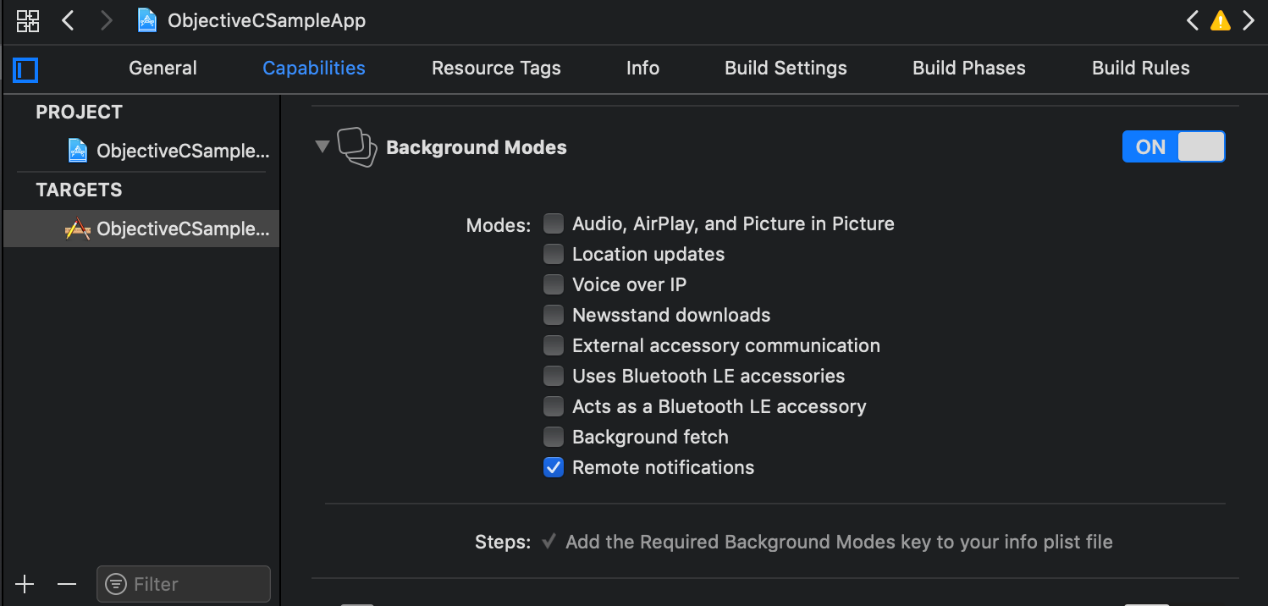
AppsFlyerAndroid.updateServerUninstallToken(token.Token);

#endif }

Af的回调文件里面添加,也可以抄作业[DemoCS文件](#DemoCS文件)



Xoce中打开通知



接入成功后如果用户卸载了游戏，af后台最多需要48小时心才能看到记录

