Unity-IAP接入流程

[Unity-IAP接入流程 1](#_Toc63263039)

[1 添加Unity IAP Package 1](#_Toc63263040)

[2 购买功能Codeless实现 4](#_Toc63263041)

[3 购买功能代码实现 5](#_Toc63263042)

[4 校验订单 6](#_Toc63263043)

[5 完成订单 9](#_Toc63263044)

## 1 添加Unity IAP Package

添加Unity IAP必须先启动Unity Services，启动Services需要Project ID, 官方文档说明链接：

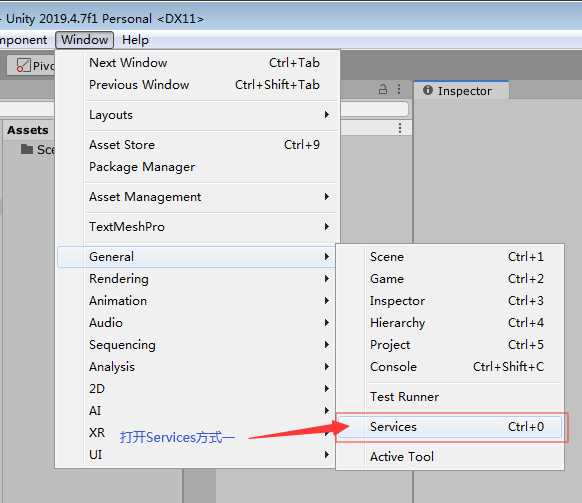
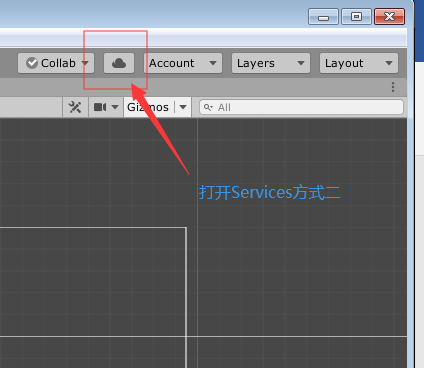
<https://docs.unity3d.com/cn/2019.4/Manual/SettingUpProjectServices.html>

添加unity iap 的官方完档说明链接：

<https://docs.unity3d.com/cn/2019.4/Manual/UnityIAP.html>

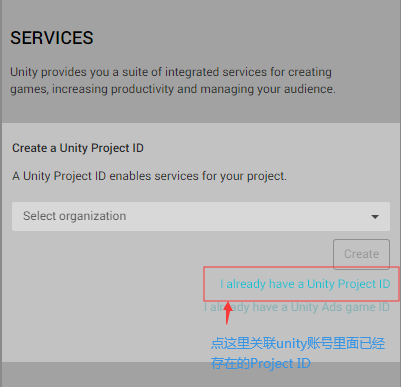
添加unity iap package之前必须先在unity service里面把unity Analytics 和 unity In-App Purchasing设为On的状态。

Step1:打开Unity Services面板，下面的截图展示了两种打开方式

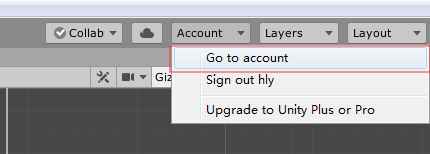
Step2: 给Services服务添加Project ID。

这个Project ID是在自己项目使用的unity账号中添加。如果之前已经有Project ID，可以在services里面直接进行关联。



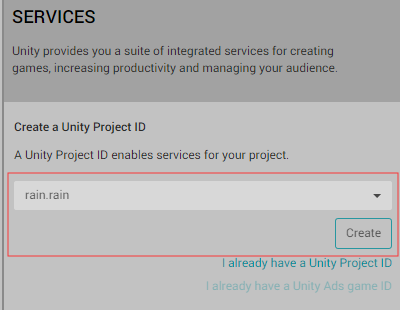
在网页端可以浏览unity账号下面所有的Project ID。

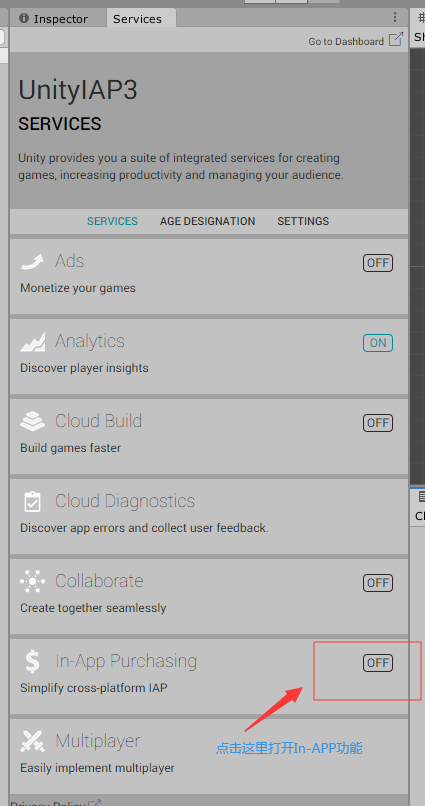
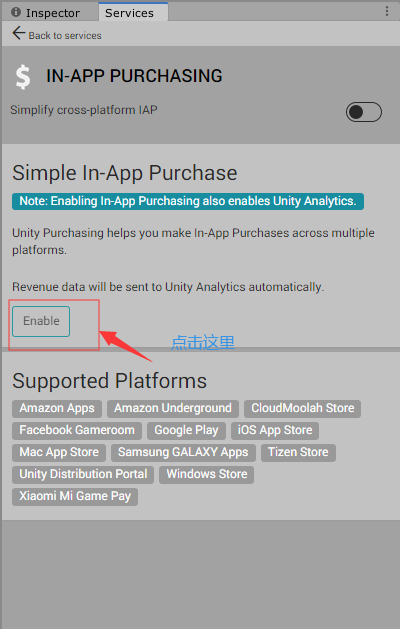
可以直接在浏览器打开unity界面或者从unity中跳转过去，下面的截图展示了如何从unity中跳转。



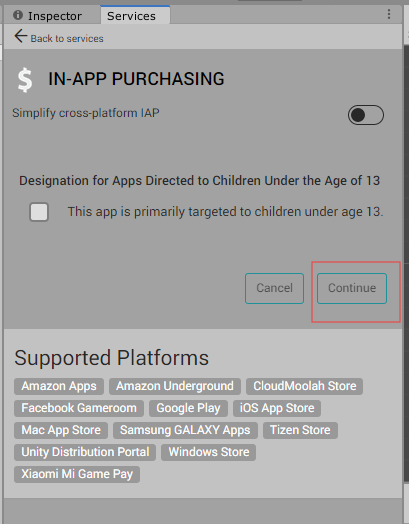
打开网页中，点击左侧的项目管理，可以打开Projects界面，里面列出了unity账号下面所有的Project，点击任意一个Project，可以看到Project Name和Project ID。

如果unity账号下面没有Project，或者想使用新的Project ID，可以通过Services面板直接创建一个，通过下拉列表选中一个organization,点击create按钮。



因为是新创建的Project ID，需要设置年龄限制，按照自己项目需求勾选就可以了。然后点击contunue



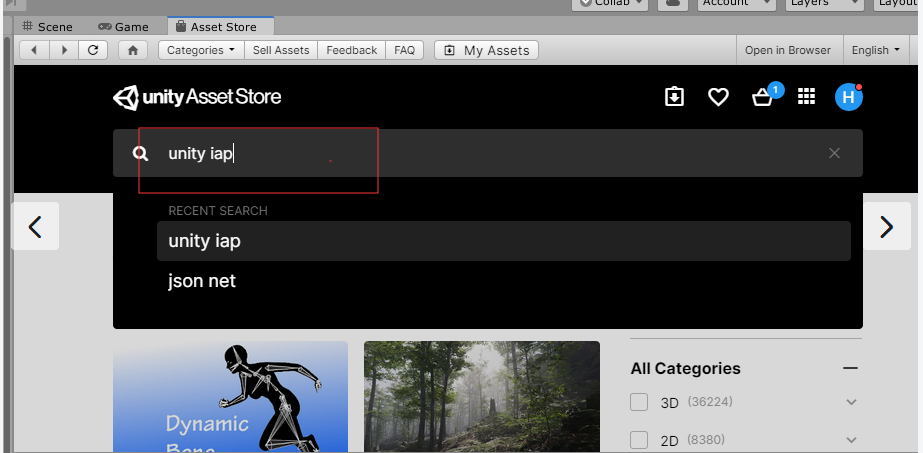
点击continue之后进入下一个界面。

Step3:导入Unity IAP Package

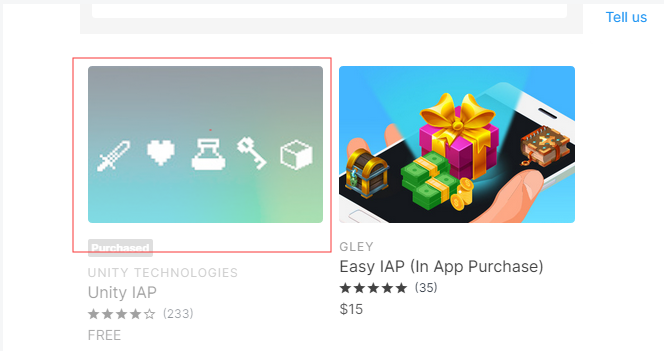


在这个界面直接点击import安装的package不是最新版本，最新版本可以从asset store中直接安装。

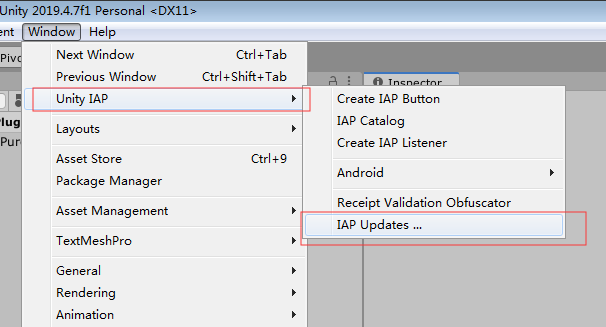
在unity编辑器中点击window,从下拉列表中选择Asset Store，打开store窗口。在搜索栏中查找unity iap



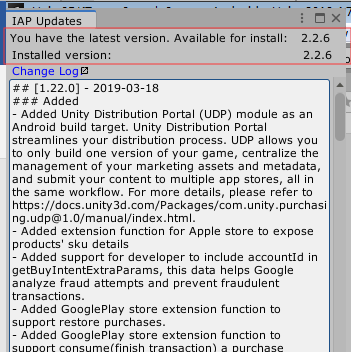
从搜索到的插件中找到下图红框圈起来的插件，这个是unity 官方的iap插件，点击打开后选择Import。

Unity IAP插件导入后，Unity编辑器的Window的下拉列表中会多一个Unity IAP的选项



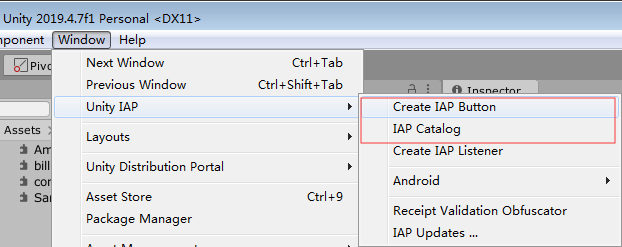
点击IAP Updates，可以查看当前已经安装好的Unity IAP版本。



## 2 购买功能Codeless实现

Unity提供了Codeless的功能，直接提供了购买按钮和购买逻辑，以及商品列表的添加，一般来说，这个功能无法满足我们游戏或者应用里面的界面需求，所以我们基本都采用下面代码实现的方式。

Unity也提供了添加商品列表，然后导出给Google Store和IOS Store使用的功能。



具体操作链接：

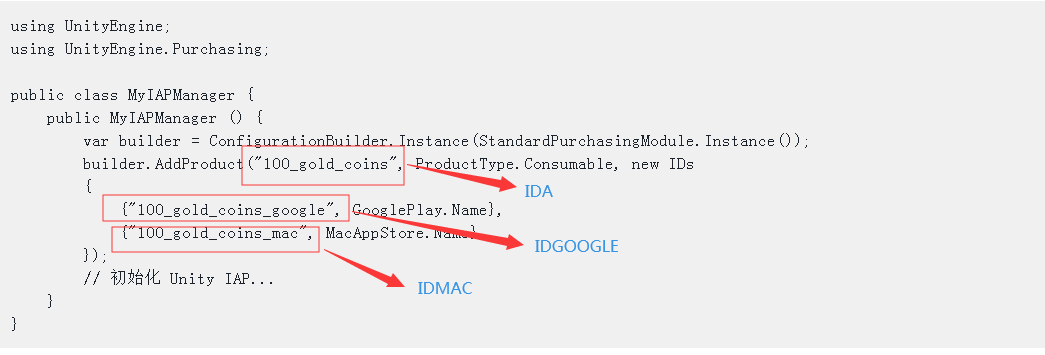
<https://docs.unity3d.com/cn/2019.4/Manual/UnityIAPCodelessIAP.html>

## 3 购买功能代码实现

Step1: 向Unity IAP注册所有商品ID

每个商品在Unity中有一个跨平台的ID，通过AddProduct接口可以把跨平台ID和各个平台的商店的商品ID对应起来。跨平台ID可以和真正的商品ID相同。

注册ID示例代码如下：

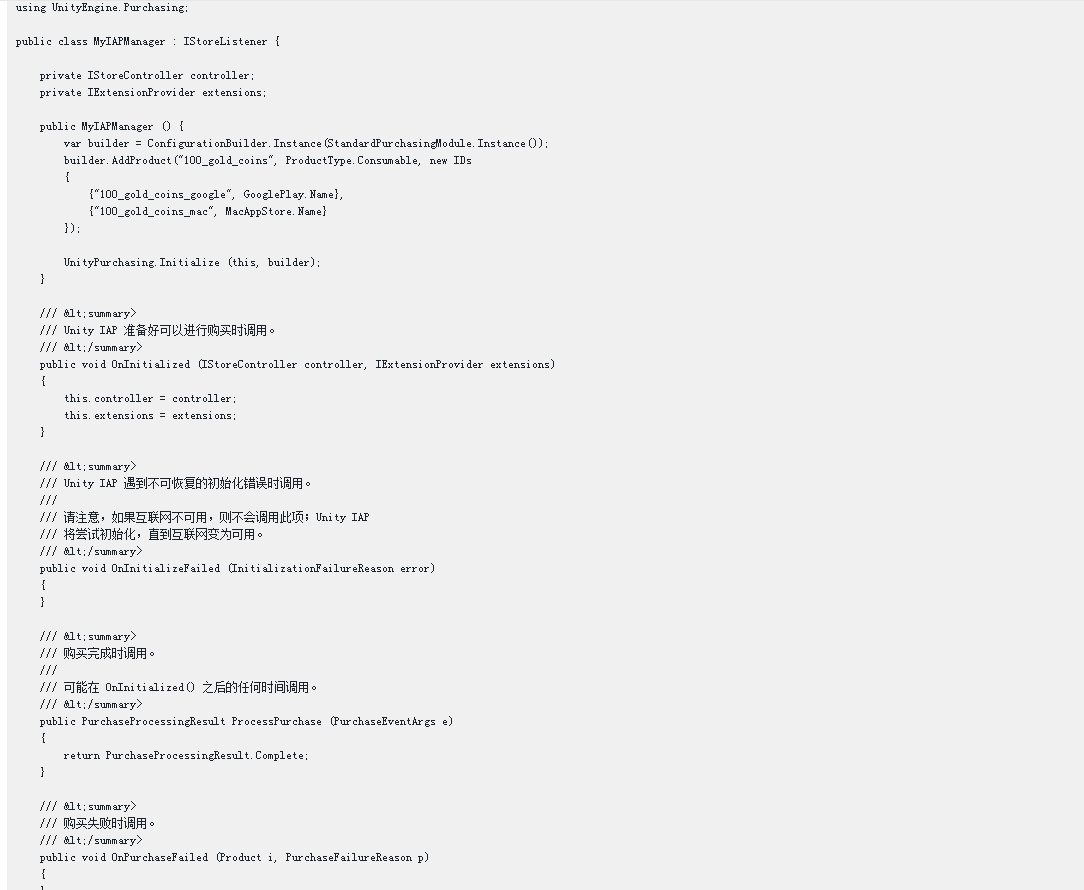


在Google Store， IDA被自动替换为IDGOOGLE， 在Mac Store， IDA被自动替换为IDMAC

如果一个商品在每个平台上面的ID都相同，可以不设置第三个参数。

Step2 初始化IAP模块，实现购买成功和失败的回调

处理IAP初始化和购买功能的类必须继承自IStoreListener，并且实现对OnInitialized， OnInitializeFailed，OnPurchaseFailed，ProcessPurchase的重写。



这里需要注意几个问题

1. 当调用 UnityPurchasing.Initialize时如果没有网络，unity 会持续在后台初始化，并不会调用OnInitializeFailed，一直到设备连上网络，初始化成功，才会调用OnInitialized接口。

如果android手机没有google框架，Unity IAP将无法初始化成功，这时Unity IAP会直接调用OnInitializeFailed。

1. 对于消耗性商品， Google Store 和IOS Store都需要将对应的消耗品订单置为完成状态，这笔订单才会从Google Store 和IOS Store的库存中删除，如果不删除，在游戏或者应用里面就不能继续购买同个ID的商品。这种机制可以防止在购买过程中断网，程序崩溃等原因造成的漏单。Unity iap综合了这种处理，当用户付费成功后，unity会调用ProcessPurchase 接口通知游戏购买成功，游戏里面可以进行商品发放逻辑处理，当发放逻辑处理完成后，需要把对应的订单置为Complete状态，如果没有置为complete状态，那么下次游戏启动，Unity IAP依然会调用ProcessPurchase 接口通知游戏这个ID的商品购买成功。

关于这个机制的unity 官网说明链接：

<https://docs.unity3d.com/cn/2019.4/Manual/UnityIAPProcessingPurchases.html>

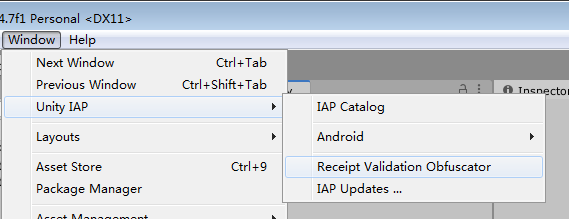
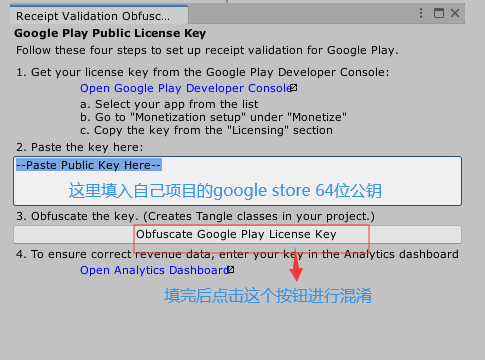
1. 在Unity IAP中，每笔订单都由 productid和transactionid来唯一确定。

## 4 校验订单

订单校验不是商品购买中必须的一步，即使不校验订单，也可以完成整个购买操作。但是不校验订单，不管是在android平台还是在ios平台，都会存在大量的虚假订单。

Step1:本地校验，利用google提供的公钥和ios的根证书进行校验

Google公钥和IOS根证书是用来做本地校验的重要数据，为了不容易被破解，unity提供了工具对着两个数据进行混淆，混淆工具的打开方式如下

在Unity IAP中，使用的是混淆后的数据，混淆前的原始数据通过接口GooglePlayTangle.Data()， AppleTangle.Data()来获取。

本地校验直接通过接口CrossPlatformValidator来实现，代码参考如下：



具体校验代码和原理的说明链接：

<https://docs.unity3d.com/cn/2019.4/Manual/UnityIAPValidatingReceipts.html>

这里重点说明下apple收据，ios7.0之后，从客户端获取到的ios收据是无法直接解析出订单详情的，之前的项目是通过把获取到的订单数据整个返回给apple，然后apple解析数据后再返回给客户端。但是unity 提供了AppleReceipt，可以对Apple 的ASN1收据格式进行建模，可以解析出订单里面的详细信息。（这里需要进一步确认）

Step2:通过服务器校验

如果有自己的服务器，可以通过自己的服务器和Google Store，IOS Store校验订单，如果没有，可以通过第三方SDK AppFlyer来实现校验功能。

AF校验订单数据，在同一时刻只能进行一笔订单的校验

Android平台校验说明链接：

<https://support.appsflyer.com/hc/zh-cn/articles/207032126-%E9%80%82%E7%94%A8%E4%BA%8E%E5%BC%80%E5%8F%91%E4%BA%BA%E5%91%98%E7%9A%84Android-SDK%E9%9B%86%E6%88%90#%E6%A0%B8%E5%BF%83api>

IOS平台购买校验说明链接：

<https://support.appsflyer.com/hc/zh-cn/articles/207032066-%E9%80%82%E7%94%A8%E4%BA%8E%E5%BC%80%E5%8F%91%E8%80%85%E7%9A%84iOS-SDK-V6-%E5%AF%B9%E6%8E%A5%E6%8C%87%E5%8D%97#%E6%A0%B8%E5%BF%83api>

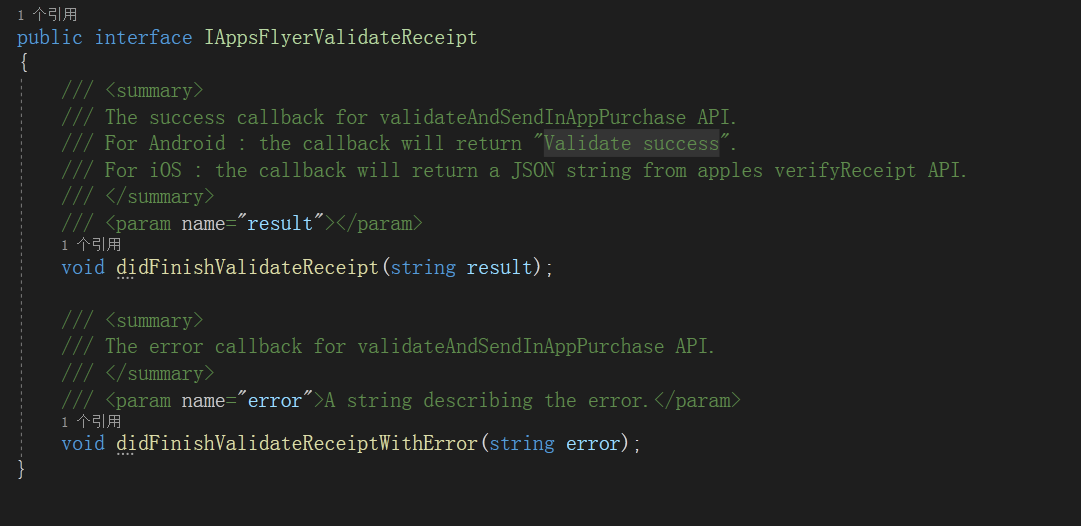
下面是步骤说明：

(a)导入SDK Package

向AF申请AppsFlyer-DEV-KEY，在untiy中导入 AppFlyer Package.

(b)自定义用于和AF交互的Class (AppsFlyerTrackerCallbacks)

AppsFlyerTrackerCallbacks需要继承自MonoBehaviour， IAppsFlyerValidateReceipt，通过AppsFlyer.initSDK接口把这个脚本设置给AF，并且在AppsFlyerTrackerCallbacks中实现下面截图中的两个接口

在Android AF返回的接口参数中没有明确传回校验的商品ID，所以在自己的代码中需要自行记录当前正在进行的商品信息

(c)Ios平台需要为沙盒模式设置开关，

当使用沙盒模式购买时，

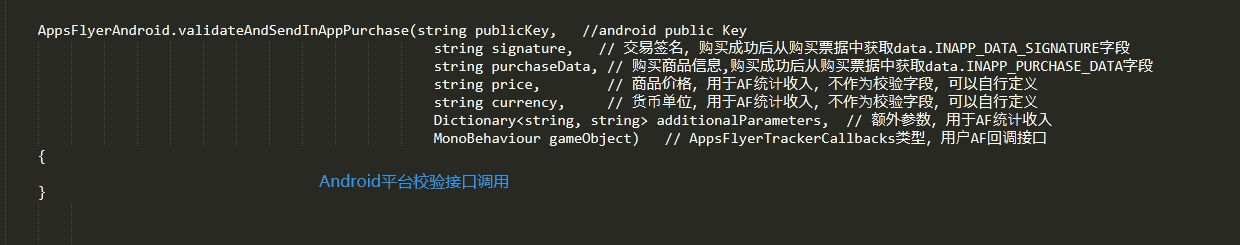
AppsFlyeriOS.setUseReceiptValidationSandbox(true).

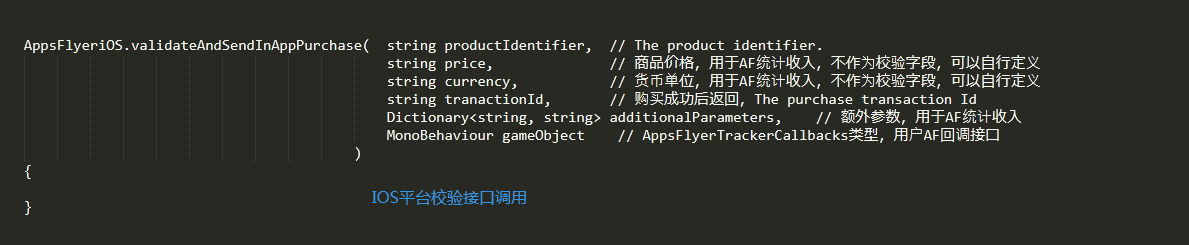
当使用正式购买方式时：

AppsFlyeriOS.setUseReceiptValidationSandbox(false).

线上版本一定要把沙盒测试模式的开关关闭。但是，当app和app内的商品第一次提交审核，app内的商品id处于待审核状态时，审核员使用的购买模式也属于沙盒模式，这时线上版本的沙盒测试模式一定要打开，否则可能因为审核员无法成功购买导致被拒绝上架。

(d)调用接口开始校验





调用校验接口后，AF会向Google Store 和IOS Store发起校验请求，接受到store的校验结果后，调用已经注册的didFinishValidateReceipt和didFinishValidateReceiptWithError接口把结果返回给客户端。

## 5 完成订单

根据校验结果发放商品后，调用把m\_StoreController(unity iap返回的实例).ConfirmPendingPurchase(product)把购买的商品订单设置为complete状态。

