**Unity iOS端友盟推送集成方案**

**一、初始化**

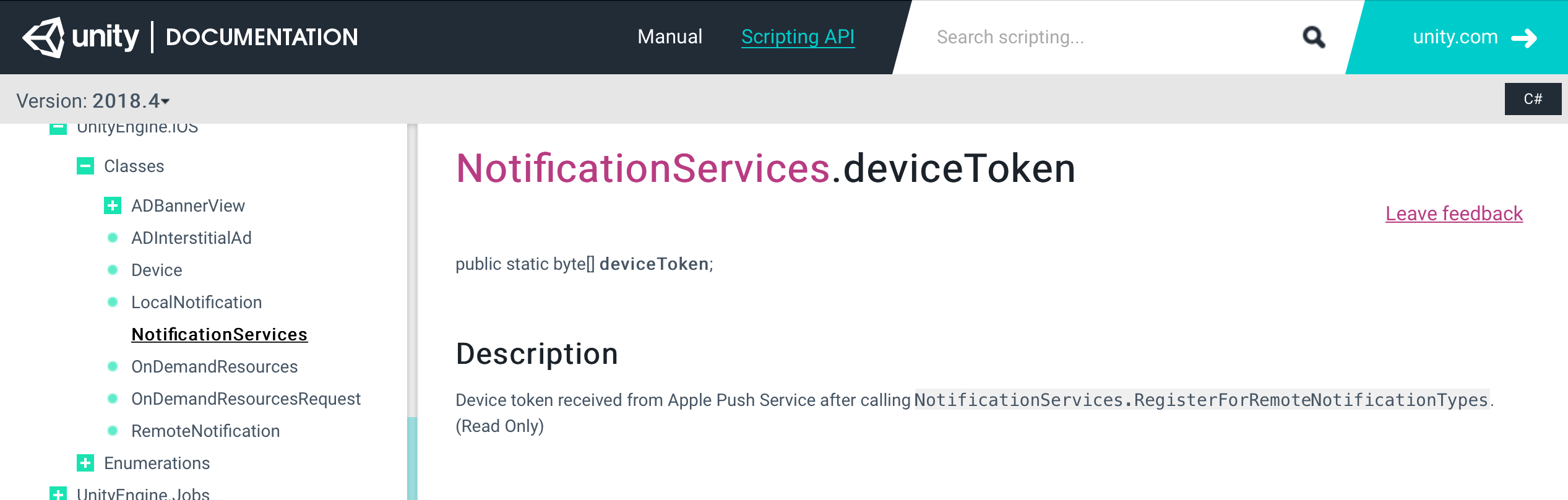
初始化友盟SDK，注册远程推送

|  |
| --- |
| C#  registerIOSRemoteNotification() |

Unity注册推送：

|  |
| --- |
| C#  NotificationServices.RegisterForRemoteNotificationTypes() |

方法参考自Unity官方文档：https://docs.unity3d.com/2018.4/Documentation/ScriptReference/iOS.NotificationServices-deviceToken.html



**二、Unity端收到DeviceToken后，发送给iOS原生注册**

**1.Unity参考在以下代码：**

|  |
| --- |
| Objective-C  AppController\_SendNotificationWithArg(kUnityDidRegisterForRemoteNotificationsWithDeviceToken, deviceToken);  UnitySendDeviceToken(deviceToken);    // 以下是两个方法的定义，传参时可以参考  **void** AppController\_SendNotificationWithArg(NSString\* name, **id** arg)  {  [[NSNotificationCenter defaultCenter] postNotificationName: name object: GetAppController() userInfo: arg];  }    **void** UnitySendDeviceToken(NSData\* deviceToken); |

**2.将收到的DeviceToken传给iOS**

调用iOS的方法：

⚠️参数是传string，还是传byte数组，待商榷，看Unity那边收到的原始数据是什么，再确定

|  |
| --- |
| C#  didRegisterForRemoteNotificationsWithDeviceToken(byte[] deviceToken) |

iOS收到deviceToken后，用友盟进行注册、设置别名（如果有需要）：

|  |
| --- |
| Swift  //系统获取Token  **func** application(**\_** application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {  **var** deviceId = String()  **if** **#available**(**iOS** 13.0, \*) {  **let** bytes = [UInt8](deviceToken)  **for** item **in** bytes {  deviceId += String(format:"%02x", item&0x000000FF)  }  print("我的deviceToken：\(deviceId)")  } **else** {  **let** device = NSData(data: deviceToken)  deviceId = device.description.replacingOccurrences(of:"<", with:"").replacingOccurrences(of:">", with:"").replacingOccurrences(of:" ", with:"")  print("我的deviceToken：\(deviceId)")  }  UMessage.registerDeviceToken(deviceToken)  **if** userManager.isLogin {  UMessage.setAlias(userManager.userID ?? "", type: "uid") { (responseObject, error) **in**  print("设置别名-\(responseObject ?? "")")  }  }  } |

**三、Unity自行对收到的推送信息进行处理**

经过查看Unity打包出来的Xcode项目代码，发现iOS端对收到的推送转发给了Unity

|  |
| --- |
| Swift  AppController\_SendNotificationWithArg(kUnityDidReceiveRemoteNotification, userInfo);  UnitySendRemoteNotification(userInfo); |

Unity可以解析推送的内容，根据自己的业务逻辑去处理

**四、其他**

**1.推送证书的创建**

参考友盟的证书配置文档：https://developer.umeng.com/docs/67966/detail/66748

建议：可以的话，使用P8证书，配置相比p12简单很多

**2.集成测试，iOS端想要本机测试的话**

除了友盟SDK的集成，还需要：

**（1）解决推送证书问题**

可以先自行配置其他app的证书，用来测试

**（2）了解Unity游戏开发的生命周期，和iOS进行交互，形成闭环**