TestFlight——完美的iOS app测试方案

TestFlight Logo

转载本文请保留以下原作者信息:

原作：onevcat http://www.onevcat.com/2012/01/testflight/

2014.5.3补充

TestFlight 现在已经修成正果，被 Apple 高价收购。虽然很遗憾不能再支持 Android 版本，但是有理由相信在 Apple 旗下的 TestFlight 将被深度整合进 Apple 开发的生态体系，并且承担更加重要的作用。不妨期待一下今年的 WWDC 上 Apple 在 CI 方面的进一步动作，预测应该会有 OSX Server 和 TestFlight 的协作方面内容。对于 CI 方面的一些其他介绍，可以参看 objc 中国的这篇帖子。

2013.3.31补充

在整理以前写的内容，想不到还有机会再对这篇帖子进行一些更新。当时写这篇帖子的时候，app内部测试以及对应的crash报告类的服务相对很少，而且并不成熟。TestFlight算是在这一领域的先行者，而随着app市场的不断膨胀，相应的类似服务也逐渐增多，比较常用的有：

崩溃报告类：

- Crittercism 个人用了一段时间，表现很稳定，但是版本更新时设置比较麻烦
- Crashlytics 相当优雅方便，最近被Twitter收购。十分推荐

用户行为统计类：

- Flurry 这个太有名了，不多说了
- Countly 好处是轻量开源，数据可以自己掌控

但是在“发布前”测试分发这个环节上，基本还没有出现能与TestFlight相匹敌的服务出现，因此如果有这方面的测试需求的话，TF依然是开发人员的首选。

当然，这一年多来，TF也进步了很多。从整个队伍建立和开发者添加开始，到桌面客户端的出现以及打包上传的简化，可以说TF也逐渐向着一个更成熟易用的方向发展。本文虽然写的时间比较早，但是整个TF的基本流程并没有发生变化，依然可以作为入门的参考。

前言

iOS开发的测试一直是令人头疼的问题。app开发的短周期和高效率的要求注定了一款app，特别是小公司的app，不会留给开发人员很多测试的时间。而在测试时往往又遇到crash报告提交困难，测试人员与开发人员沟通不便等等问题，极大延缓了测试进度。TestFlight即是为了解决iOS开发时测试的种种困难而生的服务，使用TestFlight可以十分便利地完成版本部署，测试用户Log提交，收集Carsh Log和收集用户反馈等工作，而这一切居然连一个iDP账号都不需要！

基本使用

注册

TestFlight界面友好，文档齐全，开发者在使用上不会遇到很多问题。到TestFlight官网注册账号即可开始使用。

注册账号

注册时记得勾选I am a developer，之后便可以以开发者身份管理开发和测试团队，提交测试版本和查看报告等，若没有勾选则是以测试者身份注册。若在注册时没有选上，之后在帐号设置中也可以进行更改。

勾选开发者

确认

注册完成以后会在注册邮箱中收到确认邮件。使用你的iDevice用邮件内的帐号登陆，并且完成设备注册，加入TestFlight的描述文件。关于设备注册和可能遇到的问题，可以参看这篇帖子。

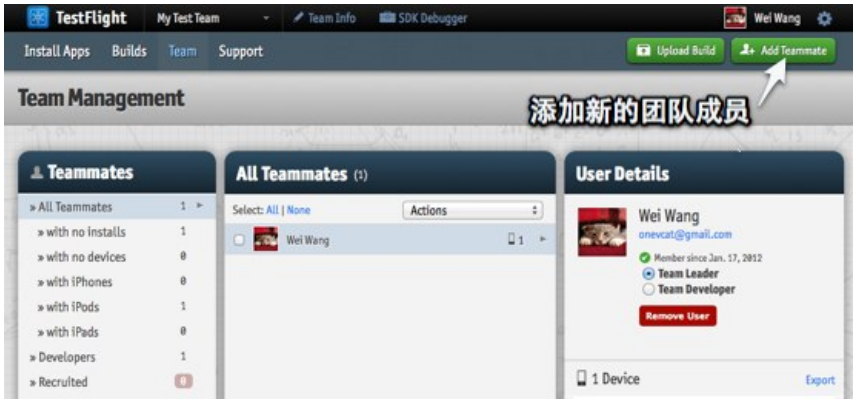
创建团队

登陆TestFlight后在自己的Dashboard可以新建一个团队。团队包括了开发者、测试者和相应的测试版本。创建团队后可以通过选择团队来查看团队的信息等情况。

建立团队

添加测试者

在团队管理界面可以为团队添加成员。填写受邀者的邮件和简单的说明，一封包含注册链接的邮件将被发送到指定邮箱。受邀者通过类似的注册和确认流程即可加入团队，参与共同开发和测试。



添加测试者

上传测试版本

上传的版本必须是包含签名的ipa，成功上传版本后即可选择给团队内的成员发邮件或推送邀请他们进行新版本的安装和测试。之后在版本管理中即可看到关于该版本的测试信息。该部分具体内容参看本文最后。

收集测试信息

在build界面中选择需要查看的版本的对应按钮即可看到收集到的测试信息，包括一般的session信息，设备使用TFLog进行的输出(需要TestFlight SDK)，crash报告，是否通过了预先设定的检查点，测试人员的安装情况等信息。

结合SDK来使用，一切测试机仿佛都变成了你自己的终端，所有的Log和设备的状态尽在掌握，而这样的便利仅仅需要点击下鼠标和写几行代码，这便是TestFlight的强大之处。

收集信息

TestFlight SDK使用

下载

不使用TestFlight的SDK的话，可以说就连这个强大的平台的一成功力都发挥不出来。点击这里从官方网站下载SDK，官方文档提供了关于SDK的很全面的说明，在支持页面也能找到很多有用的信息。

之后将以Xcode4为例，简介SDK的使用，更多信息可以参考TestFlight官网。

配置

- 将头文件加入工程：File->Add Files to
    - 找到包含SDK的文件夹
    - 勾选"Copy items into destination folder (if needed)"
    - 选择"Create groups for any added folders"

    - 勾上想要使用TestFlight SDK的Target

- 验证libTestFlight.a是否被加到link部件中
    - 在Project Navigation里选中工程文件
    - 选中想要应用SDK的Target
    - 选择Build Phase栏
    - 打开Link Binary With Libraries Phase
    - 如果libTestFlight.a不在栏内，从Project Navigation里将其拖到栏内

- 开始使用
    - 在需要用到TestFlight SDK的文件中引入头文件：*#import "TestFlight.h"*，方便起见，您也可以在工程的预编译文件中的*#ifdef* ***OBJC***块中引入
    - 获取团队token：在这个页面中对应的团队下选取TeamInfo，获取团队的token。
    - 在AppDelegate中启动TestFlight

```
-(BOOL)application:(UIApplication *_)application didFinishLaunchingWithOptions:(NSDictionary
_*)launchOptions {      // start of your application:didFinishLaunchingWithOptions // …     [TestFlight
takeOff:@"团队Token"];      // The rest of your application:didFinishLaunchingWithOptions method // … }
```

为了能得到有用的crash log(挂载过的)，必须在生成ipa的时候不剥离.dSYM文件。在Project Navigation里选中工程文件，选中需要使用TestFlight SDK的Target，在Building Setting的Deployment块下，将以下三项设为NO

- Deployment Post Processing
- Strip Debug Symbols During Copy
- Strip Linked Product

检查点

开发者可以在代码的任意位置设置检查点，当测试者通过检查点时，session里将会对此记录。比如测试者通过了某个关卡，或者提交了某个分数，或者向数据库加入了某条信息等。通过验证检查点，一方面可以检测代码是否正确，另一方面也可以作为游戏的平衡性调整和测试，用来检测用户的普遍水平。

加入检查点

```
[TestFlight passCheckpoint:@"CHECKPOINT_NAME"];
```

检查点问题

配合检查点可以向测试者提出问题，比如“是否正确地通过了演示界面？”或者“分数榜的提交正常吗？”这样的问题。在build management下可以找到Question选项，为检查点添加问题。问题的回答分为多选，是/否以及任意回答，如果选择多选的话，还需要指出问题的可能的选项。
当测试者通过问题所对应的检查点时，一个modalViewController形式的问题和选项列表会出现供测试者选择。开发者可以在build的Question选项卡中看到反馈。

反馈

TestFlight提供了一个默认的反馈界面，测试者可以填写他们想写的任何内容并将这个反馈发送给你。调用一个反馈：

```
-(IBAction)launchFeedback {      [TestFlight openFeedbackView];  }
```

一般来说可以在主界面或者最常见的界面上设置一个“反馈”按钮，这样测试者可以很方便地将他们的感受和意见发送给你。

远程Log

太棒了…配合TestFlight，现在开发者可以拿到远程客户端的Log。使用**TFLog**代替NSLog即可，任何TFLog的输出将被上传到TestFlight的服务器。如果需要详细一些的输出信息，可以用内建的**参数**的方式输出信息，比如：

```
#define NSLog(__FORMAT__, ...) TFLog((@"%s [Line %d] " __FORMAT__), __PRETTY_FUNCTION__, __LINE__, ##__VA_ARGS__)
```

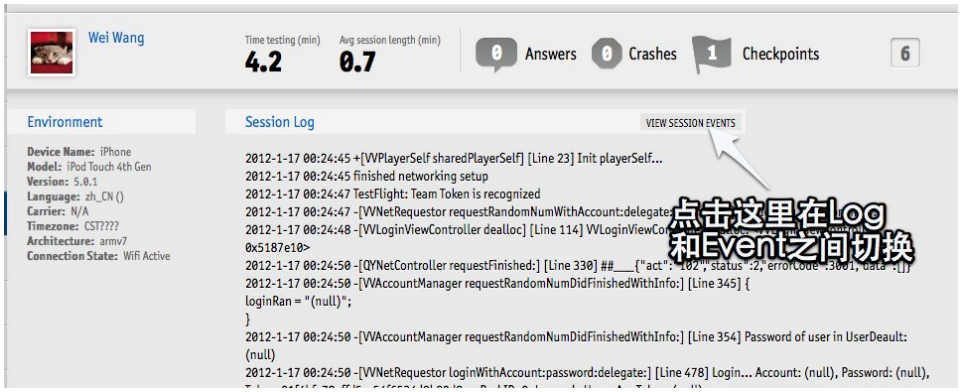
将会得到类似这样的输出

```
-[HTFCheckpointsController showYesNoQuestion:] [Line 45] Pressed YES/NO
```

所有的TFLog都不会阻塞主线程，所有的TFLog都将完成以下三种Logger工作

- TestFlight logger
- Apple System Log logger
- STDERR logger

配合以前介绍过的NSLogger(参见这篇文章)，将Log发挥到极致，让你永远掌控自己的代码吧～
Log将会在客户端进入后台或者被退出的时候上传到服务器，如果没有看到应该有的Log的话，很有可能是还在上传中。视Log文件大小而定，一般这个过程需要若干分钟。当然，巨量上几M甚至10+M的Log可能会被TestFlight拒绝哦..毕竟没有那么多存储空间..
当然，客户端必须有可用的网络环境这个功能才会启用。得到的Log会存储在Session下。



Session页面

生成和上传测试版本

打包ipa

..做过部署的童鞋对这个应该很熟了，官方也有一个详细的guide，总之照着做就没错了

- XCode3如何生成ipa
- Xcode4如何生成ipa

上传测试版本

打包好ipa后就到版本上传界面，把做好的ipa拖过去就万事大吉了。

上传ipa包

最后一步是邀请团队内的测试者进行测试。把你想邀请的测试者打上勾然后OK，包含链接的邀请邮件将会发到他们的邮箱。然后～等待测试结果和大家的反馈，并且根据反馈完善app吧～

写在最后

TestFlight是一个很棒的工具，而且关键，它现在还是免费的～

虽然有趋势以后将会收费，但是这套方案确实是方便易用..希望多支持吧～