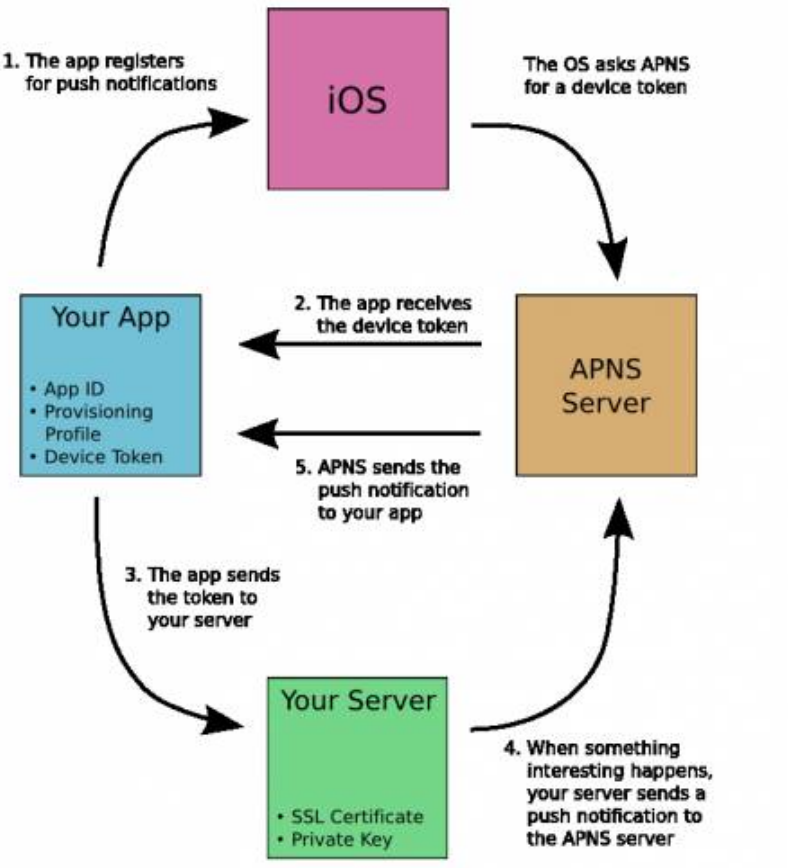
苹果信息推送服务（Apple Push Notification Service）使用总结

苹果信息推送服务（Apple Push Notification Service），是由苹果官方提供的消息推送服务。推送形式包括顶部消息条、声音以及badge number()有了APNS，应用程序可在任意状态接收到与程序有关的消息（包括运行状态not running，foreground以及background），由于在大多数情况下，iOS中最多只有一个应用能处于active状态，所以，APNS为应用的交互提供了极大的便利。

一：简介

在使用APNS之前，有这么几点需要了解：
1：APNS是免费的。只要有开发者账号便可以申请APNS证书。
2：APNS又是不可靠的，苹果对信息推送的可靠性不做任何保证。
3：APNS对消息的大小是有限制的，总容量不能超过256字节。

清楚了以上三条，各位应该对APNS适用的应用情景有所了解了。
APNS的工作流程如下图所示：



1 & 2：用户第一次安装应用并第一次启动时，会弹出对话框提示应用需要开通推送，是否允许，如果允许，应用会得到一个硬件token。
有三点需要注意：
第一，此token唯一与设备相关，同一设备上不同应用获取的token是一样的；
第二，当应用被卸载，然后重新安装时，确认对话框不会再出现，自动继承前一次安装的设置信息；
第三，推送设置可以在设置-通知中进行更改。可以选择开启消息框、声音以及badge number中的一种或多种。

3：应用将受到的token发送到服务端，也就是APNS消息的源头。
4：应用服务器通过token及证书向苹果的消息服务器发送消息。
5：苹果将接收到的消息发送到对应设备上的对应应用。
6：如果应用未处于Active状态（未启动或backgroud），默认设置下，屏幕顶部会弹出消息框，同时有声音提示，点击改消息框会进入应用，如不点击则应用图标上会有badge number出现。

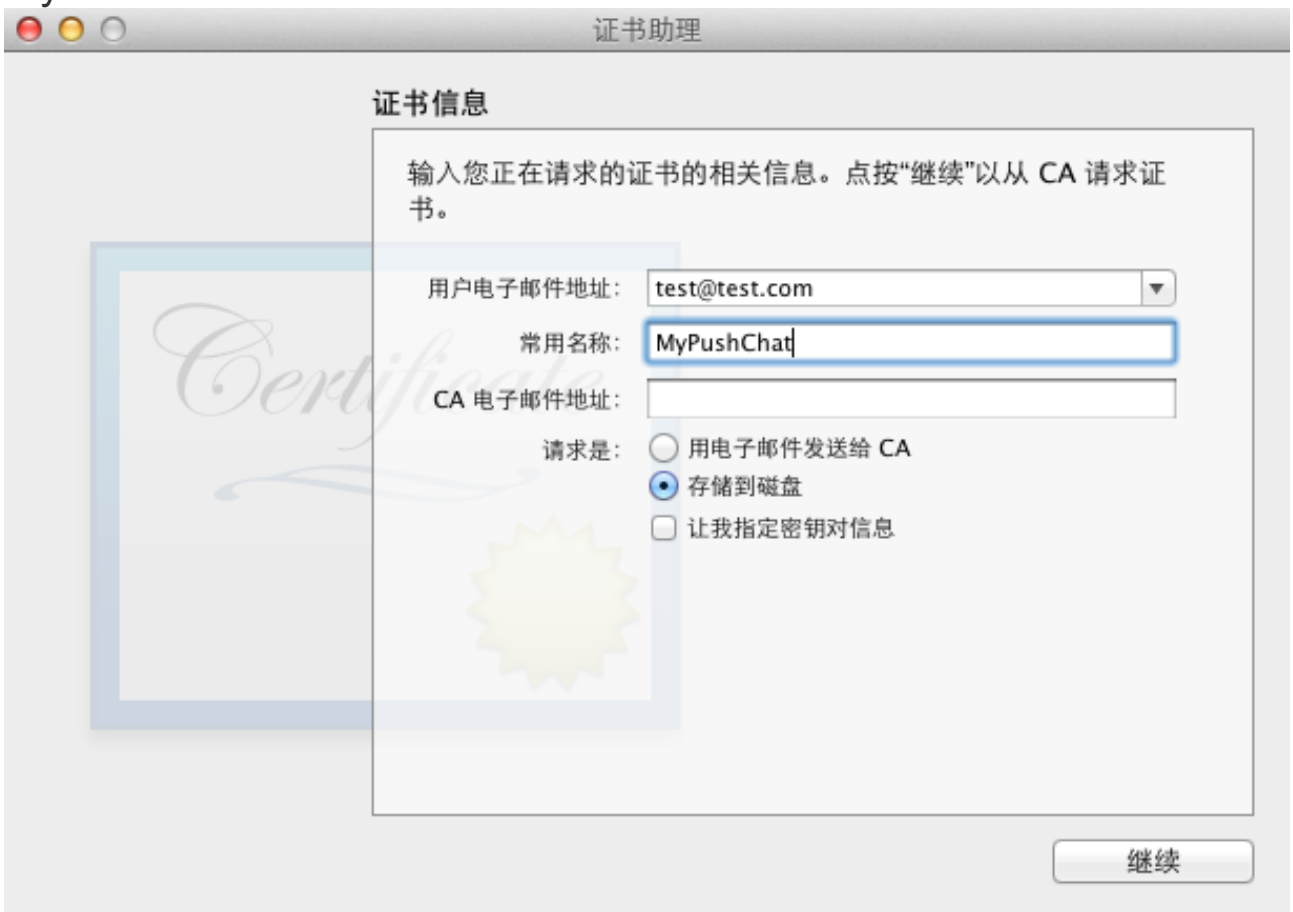
二：使用步骤

APNS的使用并不复杂，但容易出错的环节比较多，特别是证书申请的部分，要特别的注意。
下面根据我按教程实际操作的步骤进行阐述：
准备工作：
A: 一个Xcode工程，我们将其命名为MyPushChat，以及一个对应的App ID.
B:一台能用于调试的iOS设备（APNS只能在实体设备上工作，模拟器无法运行）

step1:

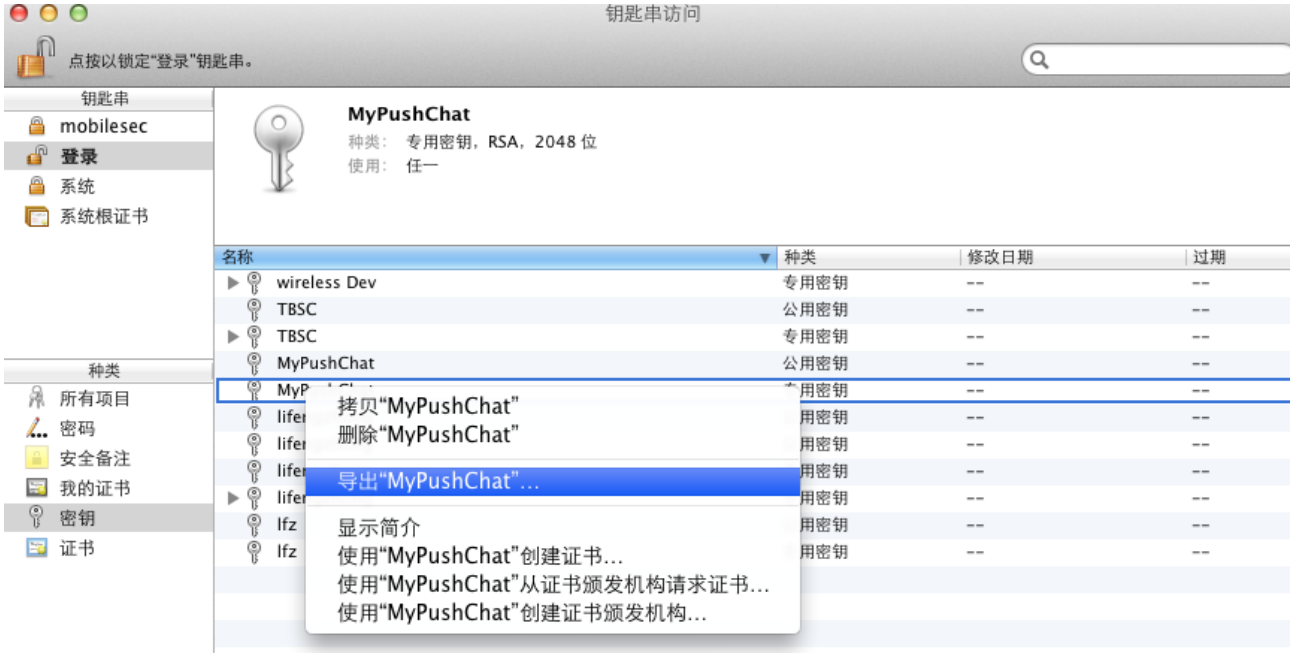
在"应用程序-使用工具"中打开"钥匙串访问"（Keychain Access），如下图所示：

在接下来的对话框中选择存储到磁盘，邮件可随意填写，名称命名为MyPushChat



点击“继续”，将文件名设为"MyPushChat"，点击存储。这样，会得到一个名为"MyPushChat.certSigningRequest"的文件，此文件要妥善保管。

从刚刚创建的csr文件中到处私钥，具体操作如下图所示：

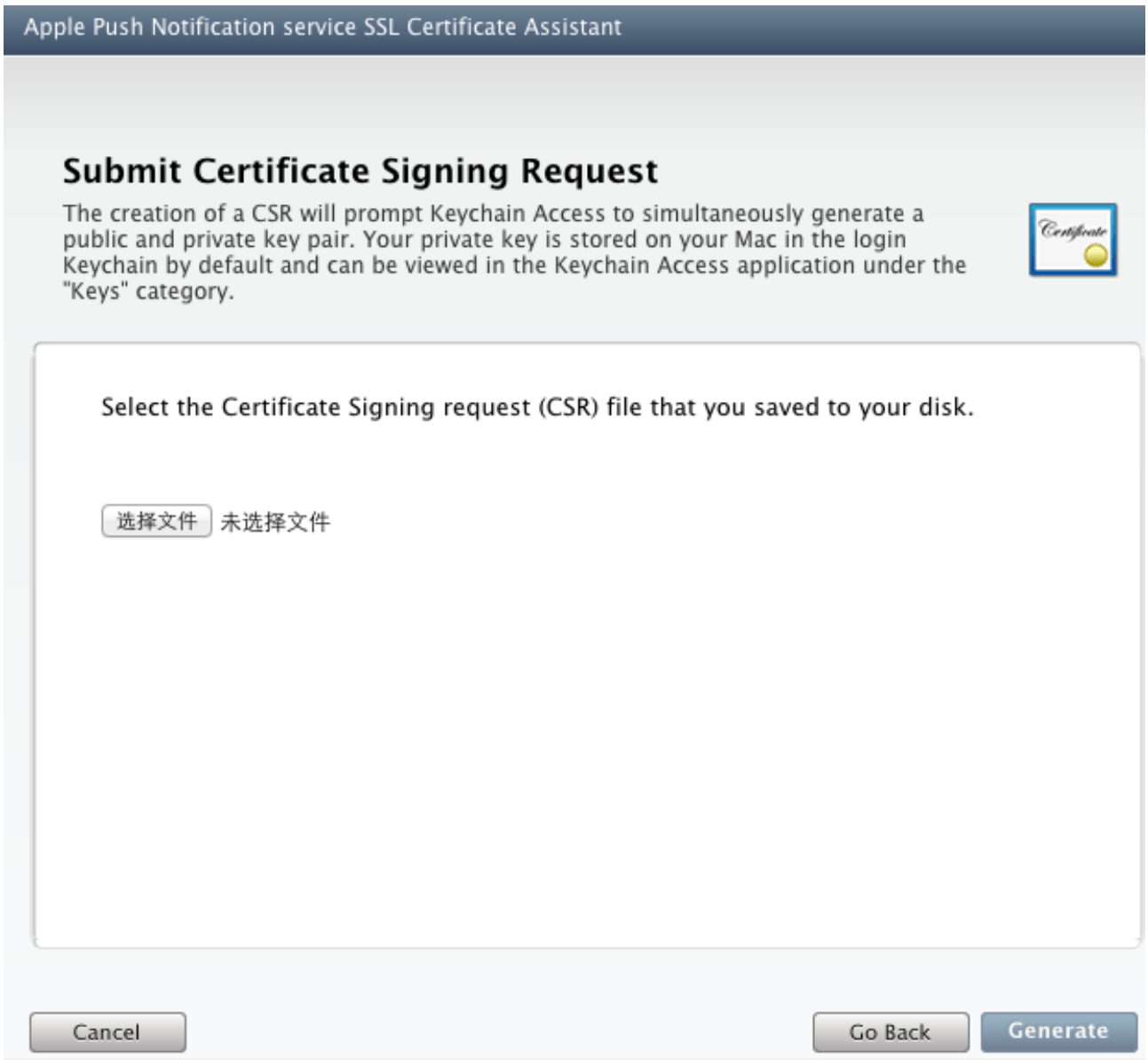


将导出的文件命名为MyPushChatKey.p12，并输入密码，请牢记此密码，这里姑且设为123456abc。
此时，我们已有文件MyPushChat.certSigningRequest，以及MyPushChatKey.p12

step2:

在App IDs中找到与MyPushChat对应的AppID, 点击右侧"Configure"按钮，勾选下图所示选择框：

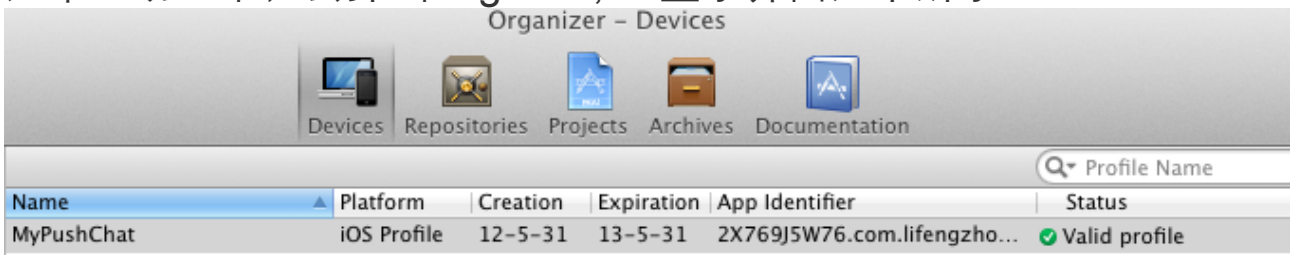
点击”Development Push SSL Certificate“右侧的configure按钮，development版本的应用于测试，有效期只有一年，且只能使用苹果的APNS测试服务器，应用发布时需要申请Distributions版本的证书。Development与Distribution版本的证书获得的Token是不一样的。弹出框如下所示：



上传"MyPushChat.certSigningRequest"并点击Generate，片刻后证书生成完毕，下载，命名为“aps_developer_identity.cer”。

step3:

打开Provision Portal，点击New Provision，将Provision File命名为"MyPushChat"，选择对应的App ID 以及Device并下载。得到文件MyPushChat.provision。双击导入此MyPushChat.Provision文件，如果一切正常，会弹出Orgnizer, 且显示界面如下所示：



step4:

将上面得到的文件都保存到桌面。打开Console,切换到桌面。
首先将aps_developer_identity.cer转换成MyPushChat.cert
命令：openssl x509 -in aps_developer_identity.cer -inform der -out MyPushChatCert.pem

然后将私钥文件转换为MyPushChatKey.pem
命令:
openssl pkcs12 -nocerts -out MyPushChatKey.pem -in MyPushChatKey.p12
Enter Import Password:
此处密码输入为前面为私钥设置的密码: 123456abc
MAC verified OK
Enter PEM pass phrase:
这里一定要输入新密码,我们设为123456abc
Verifying - Enter PEM pass phrase:

如果需要对 key不进行加密，执行下边语句
openssl rsa -in MyPushChatKey.pem -out nopassMyPushChatKey.pem

Enter pass phrase for MyPushChatKey.pem:

此处密码输入为前面为私钥设置的密码: 123456abc
writing RSA key

下一步，将MyPushChatKey.pem及MyPushChatCert.pem合成一个pem文件:
命令：cat PushChatCert.pem PushChatKey.pem > ck.pem

最后，测试一下得到的ck.pem文件
首先运行:
命令：telnet gateway.sandbox.push.apple.com 2195
如果网络正常，会出现如下所示，ctrl + C终止连接。
Trying 17.172.232.226...

Connected to gateway.sandbox.push-apple.com.akadns.net.
Escape character is '^]'.

然后使用ssl测试连接
命令：openssl s_client -connect gateway.sandbox.push.apple.com:2195 -cert MyPushChatCert.pem -key MyPushChatKey.pem
输入密码123456abc后，如果一切正常，会出现很多的输出，你将可以输入若干字符，回车后，连接将中断。

到此，最繁琐与易错的过程已经完成，证书相关工作到此为止了，进入编码阶段~

step5:

1：
在项目MyPushChat中AppDelegate.m的didFinishLaunchingWithOptions中加入如下代码

```
1.  [[UIApplication sharedApplication] registerForRemoteNotificationTypes:
2.           (UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound | UIRemoteNotificationTypeAlert)];
```

本句代码的作用为在应用第一次启动时弹出对话框让用户确认是否开启消息推送，本句注册的消息类型有BadgeNumber, 声音, 顶部消息框. 可以选择其中的一种或多种。

2：
在AppDelegate中加入如下代码：

```
1.  - (void)application:(UIApplication*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
2.  {
3.      NSLog(@"My token is: %@", deviceToken);
4.  }
```

```
5.
6.  - (void)application:
    (UIApplication*)application didFailToRegisterForRemoteNotifications
    WithError:(NSError*)error
7.  {
8.      NSLog(@"Failed to get token, error: %@", error);
9.  }
```

如果获取token成功，运行后控制台中会有如下格式的输出：

My token is:<740f4707 bebcf74f 9b7c25d4 8e335894 5f6aa01d a5ddb387 462c7eaf 61bb78ad>

将尖括号内容保存，稍后使用

同样，在AppDelegate中加入如下代码

```
1.  - (void) application:
    (UIApplication *)application didReceiveRemoteNotification:
    (NSDictionary *)userInfo
2.  {
3.
        if ( application.applicationState == UIApplicationStateActive )
     {
4.          // 程序在运行过程中受到推送通知
5.
            NSLog("%@", [[userInfo objectForKey: @"aps"] objectForKey:
    @"alert"]);
6.      } else {
7.          //程序为在运行状态受到推送通知
8.      }
9.  }
```

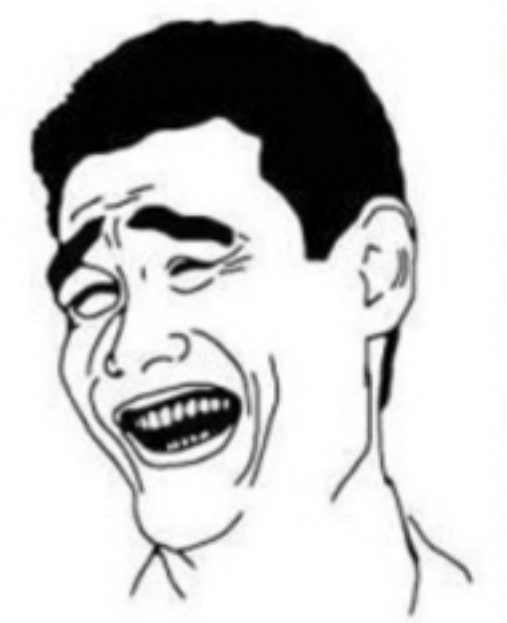
上面这段代码处理了应用分别在运行和非active状态下接收推送通知的处理方式。

3：

下载php样例程序，将其中的devicetoken字段设为刚才保存的token，注意，去掉空格。

将password设为123456abc，将message设为你想设置的内容，保存，然后命令行下进入php源码路径，运行php simplepush.php

如果人品够好，你的设备上马上会咚咚的响一下~



三：其他注意事项

1：可以使用如下代码判断开启了那些类型的消息通知：

```
1.
    UIRemoteNotificationType enabledTypes = [[UIApplication sharedAppli
    cation] enabledRemoteNotificationTypes];
2.  if (enabledTypes & UIRemoteNotificationTypeBadge) {
3.  //开启badge number
4.  }
5.  if (enabledTypes & UIRemoteNotificationTypeSound) {
6.  //开启声音
7.  }
8.  if (enabledTypes & UIRemoteNotificationTypeAlert) {
9.  //开启alert
10. }
```

2: 推送服务端推荐使用Javapns, 使用很简便，注意其使用的证书文件不是pem，而是p12格式，具体生成方法为：

一：生成csr文件（同上）
二：通过csr在苹果网站上生成cert文件(同上)
三：双击导入生成的cert文件，在keychain中同时选中csr的专用秘钥及刚刚导入的ssl证书，右键->导出，保存为p12
其他过程相同

3: 如果有把握，可以直接使用distribution版的证书和provision文件，但线上服务器有一定的限制，如果使用不当，会被苹果当ddos ban掉。

4：苹果的推送服务器会向应用服务器返回一个发送结果，对于一直失败的目标，应用服务端需要进行处理。

5：传送的message为json格式，可以在其中加入自己的字段，但同样，总大小不能超过256字节。