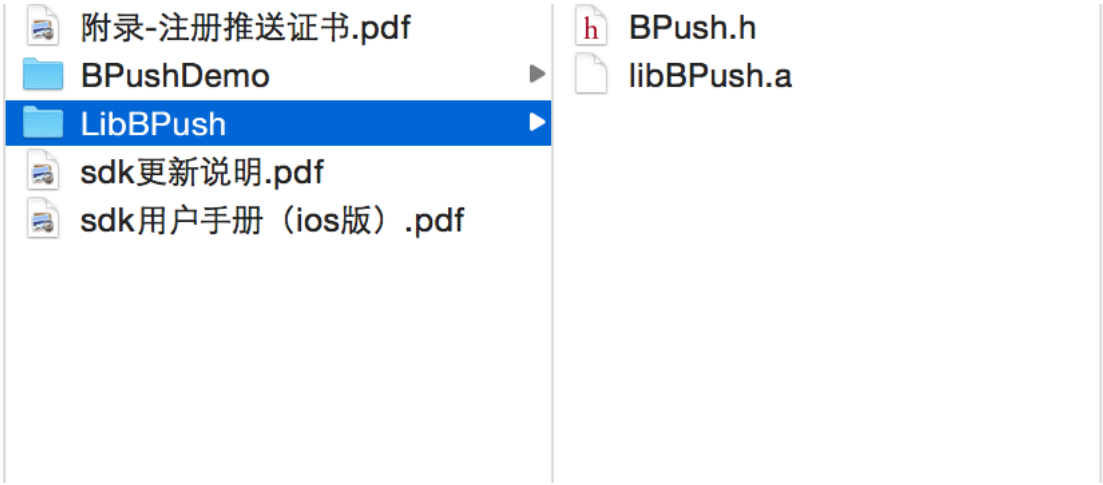
百度Push服务SDK用户手册（iOS版）

第一章 简介

百度云推送(以下简称百度 Push 服务或 Push 服务) iOS SDK 是百度官方推出的 Push 服务 iOS 开发平台软件开发工具包，包中含有供 iOS 开发者使用的百度 Push 服务的接口，开发者可通过阅读本文档从而简单快捷地集成百度 Push 服务。Push iOS SDK 的集成压缩包名为 BPush-SDK-iOS-XXX.zip(XXX为版本号)，解压后的目录结构如下图所示：



• 版本说明：记录Push iOS SDK所有的版本信息以及版本更新信息的文档。

• 用户手册：介绍 Push iOS SDK 接口及如何在项目中集成 Push 服务的文档。

• PushDemo：一项已经集成百度 Push 服务的 iOS 平台示例工程，可供用户参照，快速了解如何使用 SDK。

• LibBPush：包括头文件 BPush.h 、静态库文件libBPush.a。

第二章 SDK 功能说明

2.1 框架设计

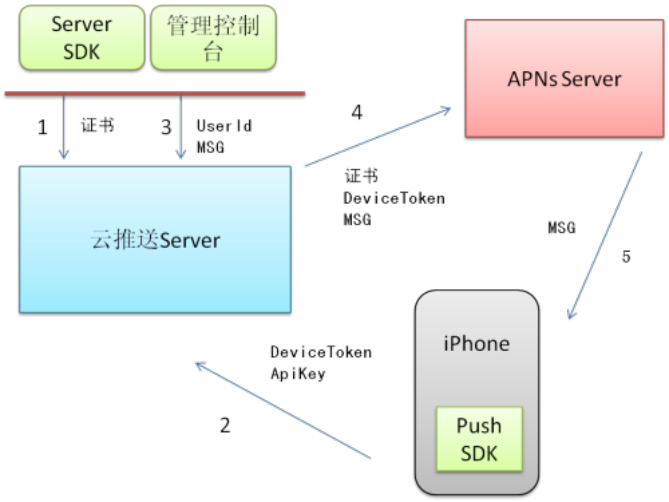
在iOS App 中加入消息推送功能的基本步骤分为如下两步

第一步，开发者在 Apple 开发者中心官网上注册推送证书(详见附录)，然后在 App 工程中添加证书，并且实现一系列推送有关的方法。

第二步，向 Apple 的推送服务器 APNS (Apple Push Notification Service) 发送需要推送的消息，APNS 在收到消息后，会将消息发到设备上。

以上整个过程较为复杂，而且功能比较单一，在集成百度的Push SDK 后，可以越过这些复杂的操作，使用百度 Push SDK 提供的 API接口，可以更加简便、简捷的在 App 中使用 Push 功能。

由于苹果 iOS 系统限制，推送到搭载 iOS 系统设备上的消息都需要经过 APNS 再下发到设备，百度 Push 服务相当于开发者与 APNS 之间的桥梁，帮助开发者完成 Push 服务。具体如下图所示：



对应上图的流程标号，推送服务的各个流程解释如下：

• 1、初始化应用推送证书

• 2、应用运行在 iOS 设备上时，向百度云推送服务器做绑定操作

• 3、向云推送服务发送请求，向指定iOS设备推送消息（广播或组播不需要user id）

• 4、百度云推送在收到开发者的推送请求后，向 APNS 转发推送消息

• 5、APNS 收到推送消息的命令，向 iOS 设备推送消息，开发者想要推送的消息成功到达指定设备

2.2 主要功能

百度云推送 SDK 主要提供以下功能接口：

1.Push 服务

- Push 服务初始化及绑定
- Push 服务解除绑定

2.Tag 管理

- 创建 tag
- 删除 tag
- 列出 tag

3.通知推送

4.推送效果反馈

第三章 快速Demo 体验

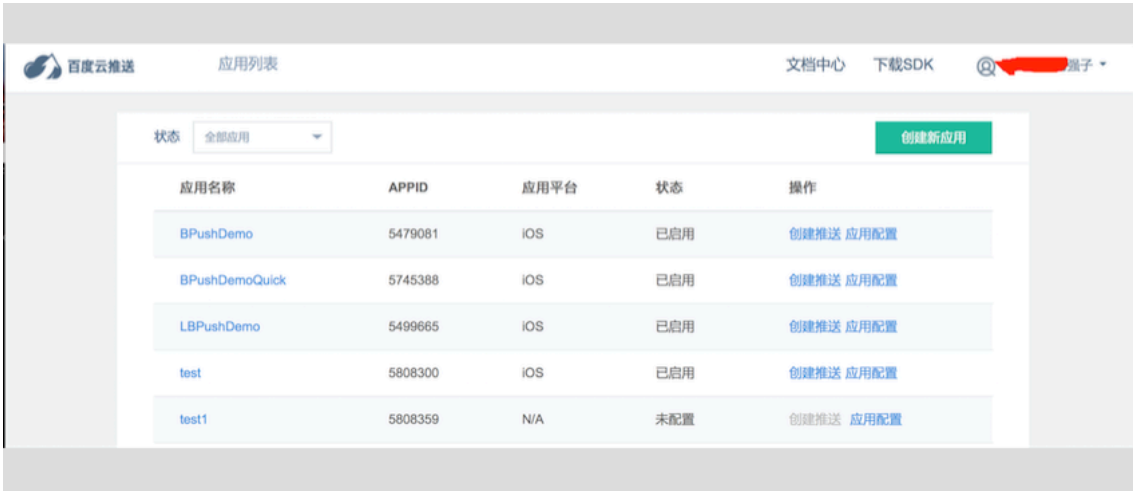
本章节将帮助你如何借鉴SDK包中的Demo源码进行简单的消息推送，快速地使用百度云推送服务。具体操作通过以下几步骤即可完成体验。

3.1 注册成为百度帐号

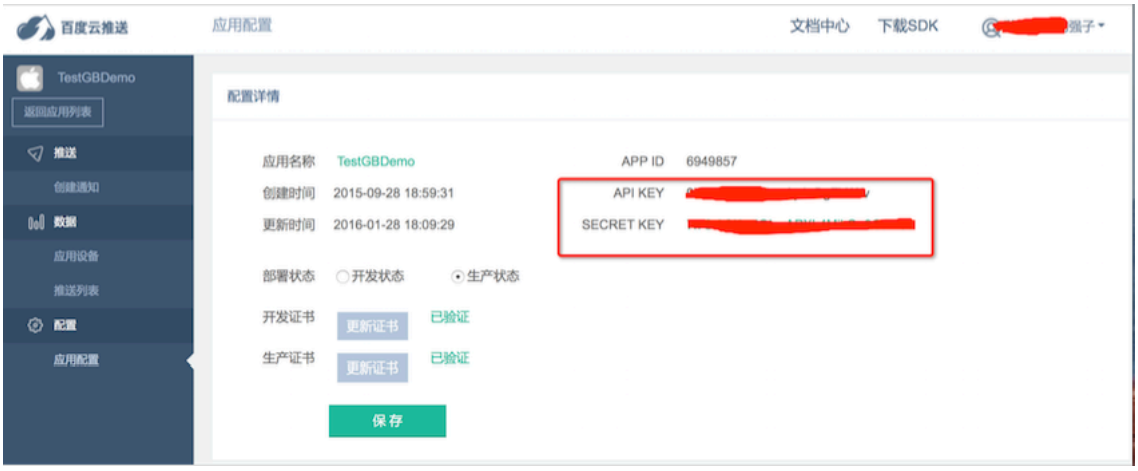
在官网百度云推送，注册百度账号，已注册用户可直接登录。页面如下图所示：

3.2 基本设置

登录成功后，可在【首页】- 【点击用户】进入应用列表总览界面，如图所示：



点击进入新建或已有的工程，需要进行【应用配置】将会显示出应用的基本信息，其中 API key 需要在 Demo 中使用，如图所示：



说明（iOS）：

开发证书：需上传推送证书的“开发版本” 的pem文件。

生产证书：需上传推送证书的“生产版本” 的pem文件。

部署状态：开发测试期间选择 【开发状态】，待 App 上线完成后可更改为【生产状态】。

注：有关如何申请证书的步骤可在文档最后【第六章-申请推送证书】中参考。

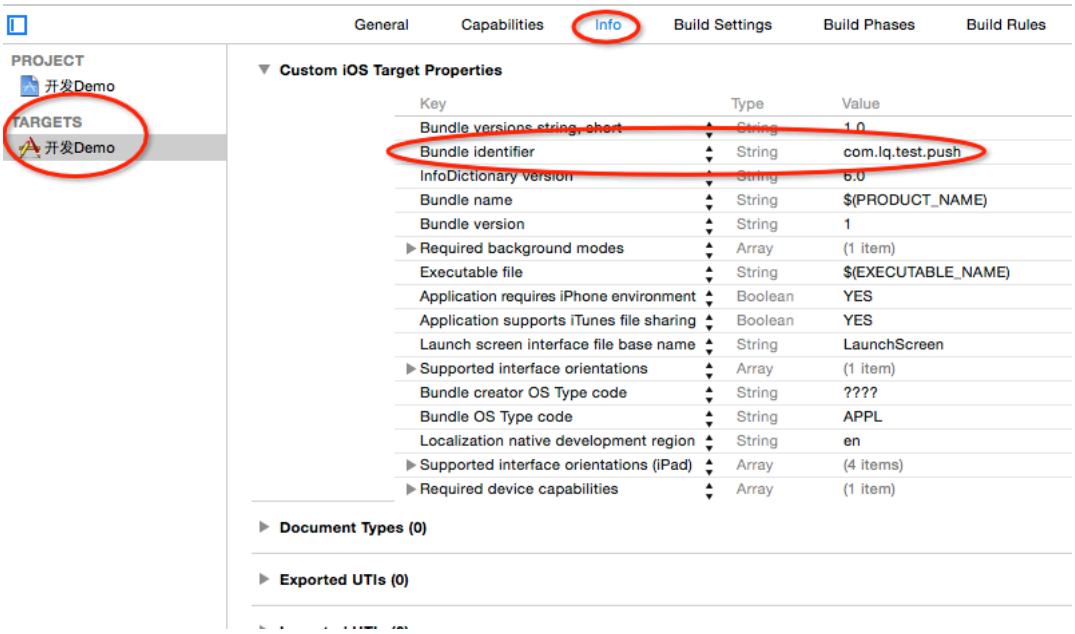
完成以上信息后，点击“保存设置”按钮即可保存信息。

3.3 下载开发工具包

下载地址：iOS SDK下载 下载后，你将得到一个压缩包，里面有相关的文档、库文件、头文件和Demo包。

3.4 修改工程设置

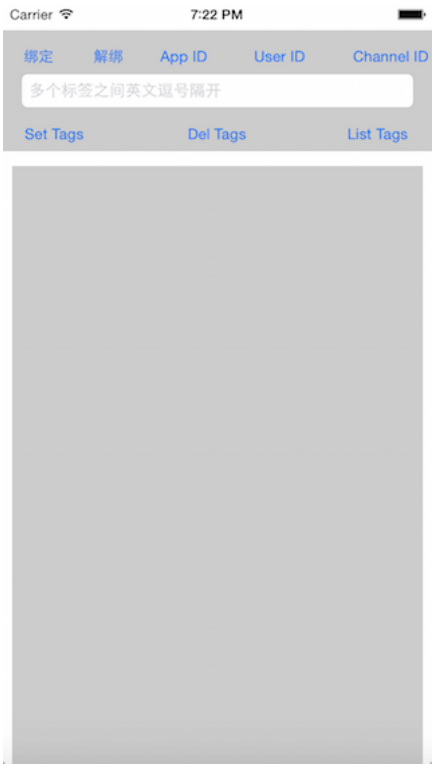
选择相应的Demo分开发Demo和发布Demo打开.xcodeproj 工程，首先需要修改 Bundle Identifier，修改为在创建证书时所选择的 Bundle ID，如下图:



修改下面方法中的apikey的值为自己的apikey,并配置为自己的证书，如下图:

```
    // iOS8 下需要使用新的 API
    if ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0) {
        UIUserNotificationType myTypes = UIUserNotificationTypeBadge | UIUserNotificationTypeSound | UIUserNotificationTy

        UIUserNotificationSettings *settings = [UIUserNotificationSettings settingsForTypes:myTypes categories:nil];
        [[UIApplication sharedApplication] registerUserNotificationSettings:settings];
    }else {
        UIRemoteNotificationType myTypes = UIRemoteNotificationTypeBadge|UIRemoteNotificationTypeAlert|UIRemoteNotificat
        [[UIApplication sharedApplication] registerForRemoteNotificationTypes:myTypes];
    }

    #warning 测试 开发环境 时需要修改BPushMode为BPushModeDevelopment 需要修改Apikey为自己的Apikey
                                                                    ⚠ 测试 开发环境 时需要修改BPushMode为BPushMod
    // 在 App 启动时注册百度云推送服务，需要提供 Apikey
    [BPush registerChannel:launchOptions apiKey:在百度云推送官网上注册后得到的apikey pushMode:BPushModeDevelopment withFirst
        nil withCategory:nil isDebug:YES];
    // App 是用户点击推送消息启动
    NSDictionary *userInfo = [launchOptions objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
    if (userInfo) {
        NSLog(@"从消息启动:%@",userInfo);
        [BPush handleNotification:userInfo];
    }
    //角标清0
    [[UIApplication sharedApplication] setApplicationIconBadgeNumber:0];
    /*
     // 测试本地通知
    [self performSelector:@selector(testLocalNotifi) withObject:nil afterDelay:1.0];
    */

    NSLog(@"device =======%@",[[UIDevice currentDevice]name]);
    return YES;
}

- (void)testLocalNotifi
{
```

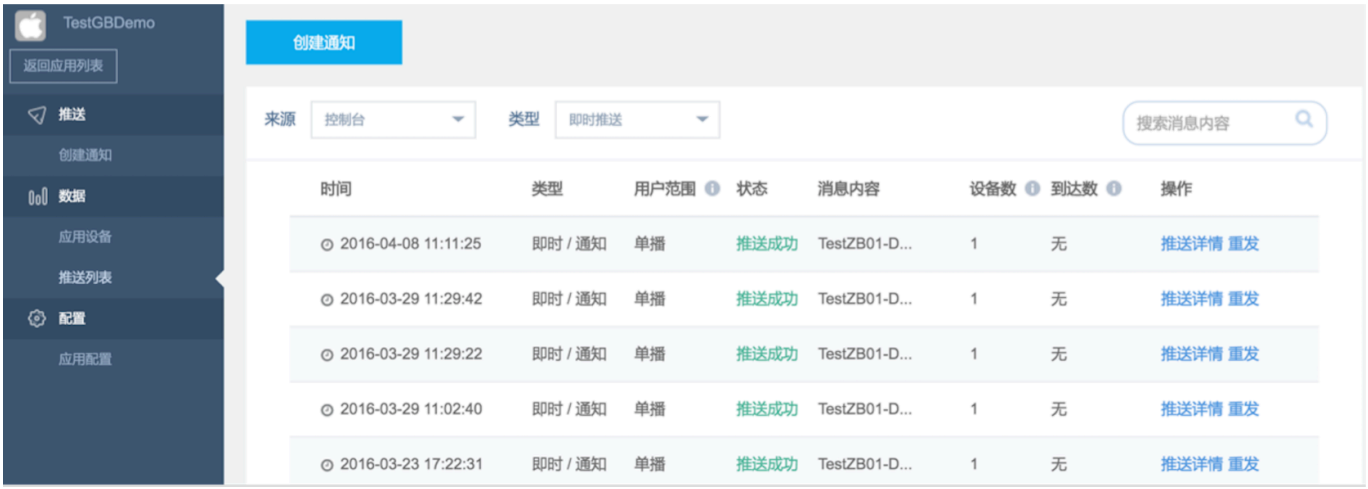
3.5 运行Demo应用



若以上步骤均无误，即可对 Demo 进行真机测试(推送通知必须在真机环境下进行测试)。如下图所示：

3.6 Demo客户端查看消息

打开Demo后，可以绑定、解绑以及添加删除tag，Demo界面会显示服务器返回数据。在绑定成功后，使用百度开放服务平台推送消息，进入指定的应用，选择云推送，将显示以下界面，如图所示：



有关控制台推送的更多内容请查阅文档。

第四章 iOS SDK开发准备

4.1 运行环境

• iOS 5.1及以上版本

• GPRS、3G、4G及Wi-Fi网络

• Apple应用ID以及对应的推送证书（可参照附录）

4.2 接入百度开放服务平台

关于如何注册百度开发者以及如何创建应用等内容可参照文档第三章「3.1 注册成为开发者」，其中，应用ID(APPID)用于标识开发者创建的应用程序，API Key(即Client_id)是开发者创建的应用程序的唯一标识，开发者在调用百度API时必须传入此参数。

4.3 用户账户支持

4.3.1 已有百度账户

开发者可选择使用oauth2.0协议接入百度开放平台，所有用户标识使用百度的user id作为唯一标识，使用AccessToken作为验证凭证。详细信息可前往百度开放服务平台-百度OAuth了解。

4.3.2 无账号登录体系

1. API Key登陆 开发者无需接入百度账户体系，每个终端直接通过API Key向Server请求用户标识user id，此id是根据端上的属性生成，具备唯一性，开发者可通过此id对应到自己的账户系统，登陆方式方便灵活，但需要开发者自己设计账户体系和登录界面。
2. 开发者AccessToken登陆 Access Token可以通过以下两种方式获取： 第一种，普通用户百度账户登陆获取，当应用程序使用百度账号作为账户体系时使用，即4.3.1节所示，可以换取百度账户的唯一的user id； 第二种，开发者百度账户登录获取，注册server会根据不同终端的device id分配不同的user id。这种方式可以实现不依赖于百度账户的第三方登陆体系，使用该登陆方式时，开发者需要定期的与端上做Access Token的同步，以保证端上的Access Token不过期。

4.4 获取客户端和服务端SDK开发工具包

开发者可以去官网下载：下载地址

第五章 iOS SDK开发步骤

5.1 添加SDK到工程中

添加到SDK到工程中的步骤如下：

• 将libBPush.a和BPush.h添加到自己的工程下，添加时需要注意勾选当前Target

• SDK需要以下库：Foundation.framework、CoreTelephony.framework、libz.dylib、SystemConfiguration.framework，CoreLocation.framework 如果使用了idfa版需要添加AdSupport.framework 在工程中添加。

5.2 iOS SDK API

5.2.1 API总览

API主要包含在头文件BPush.h中，目前有以下接口可供支持：

功能	API 函数原型
注册Push	+ (void)registerChannel:(NSDictionary *)launchOptions apiKey:(NSString *)apikey pushMode:(BPushMode)mode withFirstAction:(NSString *)rightAction withSecondAction:(NSString *)leftAction withCategory:(NSString *)category useBehaviorTextInput:(BOOL)behaviorTextInput isDebug:(BOOL)isdebug
关闭lbs推送	+ (void)disableLbs
设置Access Token	+ (void)setAccessToken:(NSString *)token
注册Device Token	+ (void)registerDeviceToken:(NSData *)deviceToken
绑定	+ (void)bindChannelWithCompleteHandler:(BPushCallBack)handler
解除绑定	+ (void)unbindChannelWithCompleteHandler:(BPushCallBack)handler
处理推送消息	+ (void)handleNotification:(NSDictionary *)userInfo
设置 tag	+ (void)setTag:(NSString *)tag withCompleteHandler:(BPushCallBack)handler
删除 tag	+ (void)delTag:(NSString *)tag withCompleteHandler:(BPushCallBack)handler
列举 tag	+ (void)listTagsWithCompleteHandler:(BPushCallBack)handler
获取appId	+ (NSString *) getAppId
获取channelId	+ (NSString *) getChannelId
获取userId	+ (NSString *) getUserId
创建本地消息通知	+ (void)localNotification:(NSDate *)date alertBody:(NSString *)body badge:(int)bage withFirstAction:(NSString *)leftAction withSecondAction:(NSString *)rightAction userInfo:(NSDictionary *)userInfo soundName:(NSString *)soundName region:(CLRegion *)region regionTriggersOnce:(BOOL)regionTriggersOnce category:(NSString *)category useBehaviorTextInput:(BOOL)behaviorTextInput
在前台展现本地消息通知	+ (void)showLocalNotificationAtFront:(UILocalNotification *)notification identifierKey:(NSString *)notificationKey
根据通知的标识删除本地消息通知	+ (void)deleteLocalNotificationWithIdentifierKey:(NSString *)notificationKey
删除指定本地消息通知	+ (void)deleteLocalNotification:(UILocalNotification *)localNotification
获取指定本地消息通知	+ (NSArray *)findLocalNotificationWithIdentifier:(NSString *)notificationKey
清除所有本地消息通知	+ (void)clearAllLocalNotifications

注意：API 调用的返回结果都是用BPushCallback回调.

5.2.2 相关常量的定义：

1. 百度Push请求，返回结果的键。

   NSString *const BPushRequestErrorCodeKey; 错误码。0成功，其它失败，具体参见BpushErrorCode。

NSString *const BPushRequestErrorMsgKey; 错误信息。成功时为空。

NSString *const BPushRequestRequestIdKey; 向百度Push服务发起请求的请求ID，用来追踪定位问题。

NSString *const BPushRequesAppIdKey; 绑定成功时返回的app id。

NSString *const BPushRequestUserIdKey; 绑定成功时返回的user id。

NSString *const BPushRequestChannelIdKey; 绑定成功时，返回的channel id。

2. 百度 Push 请求错误码，BPushErrorCode 枚举类型。

   BPushErrorCode*Success = 0, BPushErrorCode*MethodTooOften = 22, // 调用过于频繁 BPushErrorCode*NetworkInvalible = 10002, // 网络连接问题 BPushErrorCode*InternalError = 30600, // 服务器内部错误 BPushErrorCode*MethodNodAllowed = 30601, // 请求方法不允许 BPushErrorCode*ParamsNotValid = 30602, // 请求参数错误 BPushErrorCode*AuthenFailed = 30603, // 权限验证失败 BPushErrorCode*DataNotFound = 30605, // 请求数据不存在 BPushErrorCode*RequestExpired = 30606, // 请求时间戳验证超时 BPushErrorCode*BindNotExists = 30608, // 绑定关系不存在

3. 方法名，即 onMethod。

   NSString *const BpushRequestMethod_Bind; bind 方法。

   NSString *constBpushRequestMethod_Unbind; unbind 方法。

   NSString *const BpushRequestMethod_SetTag; setTags 方法。

   NSString *const BpushRequestMethod_DelTag; delTags 方法。

   NSString *constBpushRequestMethod_ListTag; listTag 方法。

5.2.3 API 说明

注册 Push

功能描述：

注册百度Push服务。

函数原型：

```
+ (void)registerChannel:(NSDictionary *)launchOptions apiKey:(NSString *)apikey pushMode:(BPushMode)mode withFirstAction:(NSStri
ng *)rightAction withSecondAction:(NSString *)leftAction withCategory:(NSString *)category useBehaviorTextInput:(BOOL)behaviorTe
xtInput isDebug:(BOOL)isdebug;
```

参数说明：

- launchOptions：App 启动时系统提供的参数。
- apiKey：通过apikey注册百度推送。
- mode：当前推送的环境。
- isdebug：是否是debug模式。
- leftAction：第二个按钮名字默认会关闭应用 iOS 9 快捷回复需要先设置此参数
- rightAction：快捷回复通知的第一个按钮名字默认为打开应用
- category 自定义参数 一组动作的唯一标示 需要与服务端ans的category匹配才能展现通知样式 iOS 9 快捷回复需要先设置此参数
- behaviorTextInput 是否启用 iOS 9快捷回复

返回结果：

（无）

设置Access Token

函数原型：

```
+ (void)setAccessToken:(NSString *)token;
```

功能描述：

当App用Access Token方式绑定时，用于设置Access Token。无账号绑定(API Key绑定)时无需调用该接口，如果App帐号体系用的不是百度的帐号，也可以用开发者自己的百度帐号获取Access Token给所有端使用，即4.3.2节无账号登录体系的第二种方式。必须注意的是，Access token有过期时间，需要及时为每个端更新Access token，并重新执行绑定操作。

参数说明：

- token：通过百度帐号认证方式获取的Access token。

返回结果：

（无）

关闭位置定位

函数原型：

```
+ (void)disableLbs;
```

功能描述：

关闭lbs推送模式，默认是开启的（需要设置获取地理位置权限），用户可以选择关闭。

参数说明：

- deviceToken：直接使用应用程序委托中的回调方法application:didRegisterForRemoteNotificationsWithDeviceToken:传入的deviceToken参数即可。

返回结果：

（无）

收集BPush崩溃日志

函数原型：

```
+ (void)uploadBPushCrashLog;
```

功能描述：

开启BPush 崩溃日志收集，没有使用其他第三方崩溃收集工具的，建议调用此接口，BPush 会收集由于BPush SDK 本身引起的崩溃 便于SDK搜集已知问题，更快的修复问题。

参数说明：

（无）

返回结果：

（无）

注册Device Token

函数原型：

```
+ (void)registerDeviceToken:(NSData *)deviceToken;
```

功能描述：

向百度 Push服务注册客户端的Device token，Push服务推送消息时必须用到，由于Device token是可变的，所以需要经常性的向服务端注册。

参数说明：

- deviceToken：直接使用应用程序委托中的回调方法application:didRegisterForRemoteNotificationsWithDeviceToken:传入的deviceToken参数即可。

返回结果：

（无）

绑定

函数原型：

```
+ (void)bindChannelWithCompleteHandler:(BPushCallBack)handler;
```

功能描述：

绑定Push服务通道，必须在设置好Access Token或者API Key并且注册deviceToken后才可绑定。绑定结果通过BPushCallBack回调返回。

参数说明：

（无）

返回结果：

BPushCallBack中有绑定结果的反馈，error_code 为0时绑定成功。绑定成功后可以获取appid,channelid,userid等信息。

解除绑定

函数原型：

```
+ (void)unbindChannelWithCompleteHandler:(BPushCallBack)handler;
```

功能描述：

解除已经绑定的Push服务通道，成功解除绑定后，将无法接收云推送消息，也无法进行set tag和del tag操作。重新绑定后即可恢复推送功能。解绑请求的结果通过BPushCallBack回调返回。

参数说明：

（无）

返回结果：

BPushCallBack中有解绑结果的反馈,解绑成功会得到requestid。

处理Push消息

函数原型：

```
+ (void)handleNotification:(NSDictionary *)userInfo;
```

功能描述：

处理Push消息。用于对推送消息的反馈和统计，如果想对消息的推送情况获取及时的反馈，请正确调用该方法。在application:didReceiveRemoteNotification:方法中调用即可。

参数说明：

- userInfo：直接使用application:didReceiveRemoteNotification:方法中的userInfo参数即可。

返回结果：

（无）

设置tag

函数原型：

```
+ (void)setTag:(NSString *)tag withCompleteHandler:(BPushCallBack)handler;
```

功能描述：

BPushCallBack中有设置标签结果的反馈,，error_code 为0时设置成功会得到设置结果。

参数说明：

- tag：需要设置的tag。

返回结果：

（无）

设置tags

函数原型：

```
+ (void)setTags:(NSArray *)tags withCompleteHandler:(BPushCallBack)handler;
```

功能描述：

开发者需要在绑定成功的回调中调用此方法来设置多个标签。

参数说明：

- tags：需要设置的tag数组。

返回结果：

BPushCallBack中有设置标签结果的反馈,，error_code 为0时设置成功会得到设置结果。

删除tag

函数原型：

```
+ (void)delTag:(NSString *)tag withCompleteHandler:(BPushCallBack)handler;
```

功能描述：

开发者需要在绑定成功的回调中调用此方法来删除标签。

参数说明：

- tag：需要删除的tag。

返回结果：

BPushCallBack中有删除标签结果的反馈,error_code 为0时设置成功会得到删除的结果。

删除tags

函数原型：

```
+ (void)delTag:(NSString *)tag withCompleteHandler:(BPushCallBack)handler;
```

功能描述：

开发者需要在绑定成功的回调中调用此方法来删除多个标签。

参数说明：

- tags：需要删除的tag数组。

返回结果：

（无）

获取当前设备应用的tag列表

函数原型：

```
+ (void)listTagsWithCompleteHandler:(BPushCallBack)handler;
```

功能描述：

开发者需要在绑定成功的回调中调用此方法来获取当前设备应用的tag列表。

参数说明：

（无）

返回结果：

BPushCallBack中有获取标签结果的反馈,error_code 为0时表示获取标签成功会得到返回结果。

获取应用绑定信息

函数原型：

```
+ (NSString *) getChannelId;
+ (NSString *) getUserId;
+ (NSString *) getAppId;
```

功能描述：

在应用成功绑定后，可以通过调用这三个接口获取appId、channelid和userId，在绑定之前或者解绑之后，返回空。

参数说明：

（无）

返回结果：

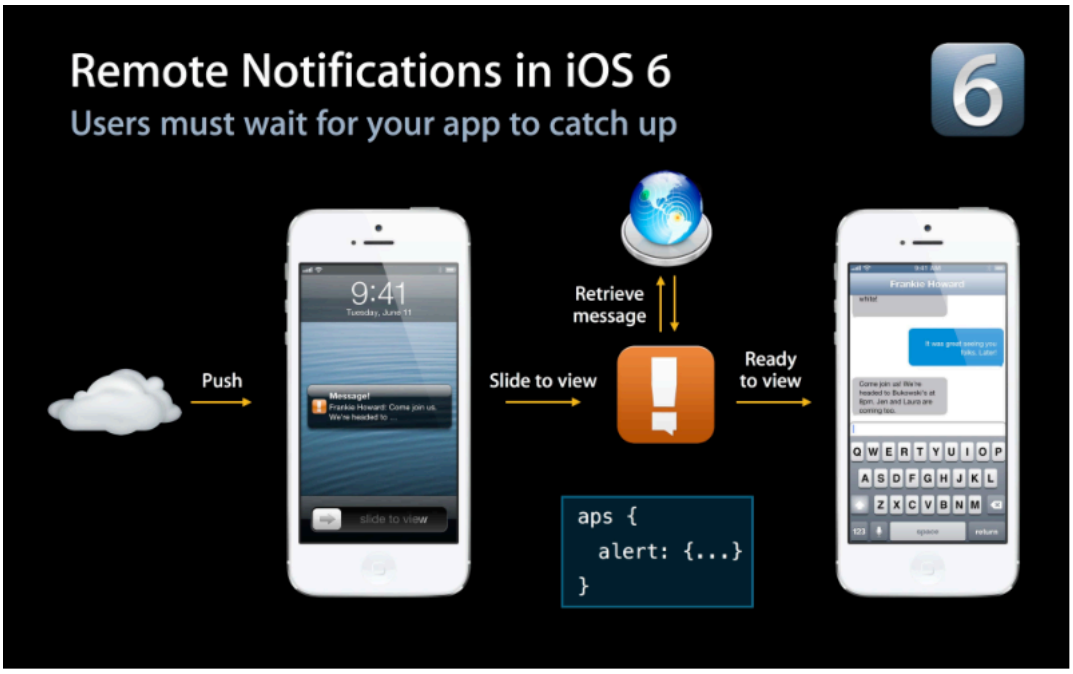
字符串类型的appid、channelid或userid

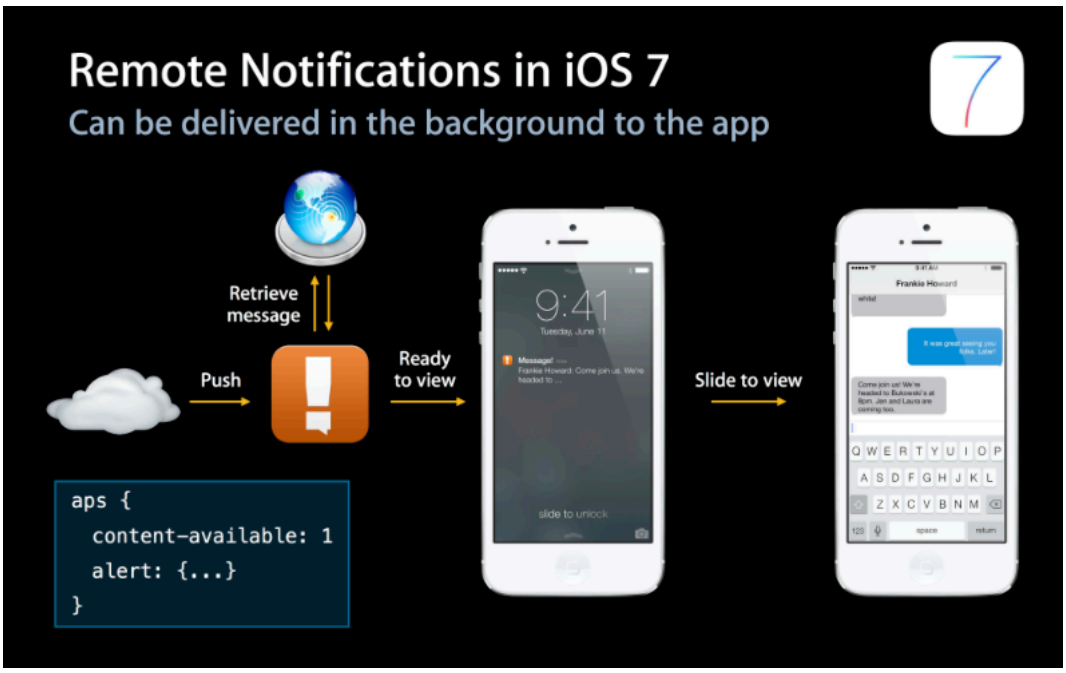
第六章 限制与注意

6.1 iOS7、iOS8、iOS9、iOS10 特性

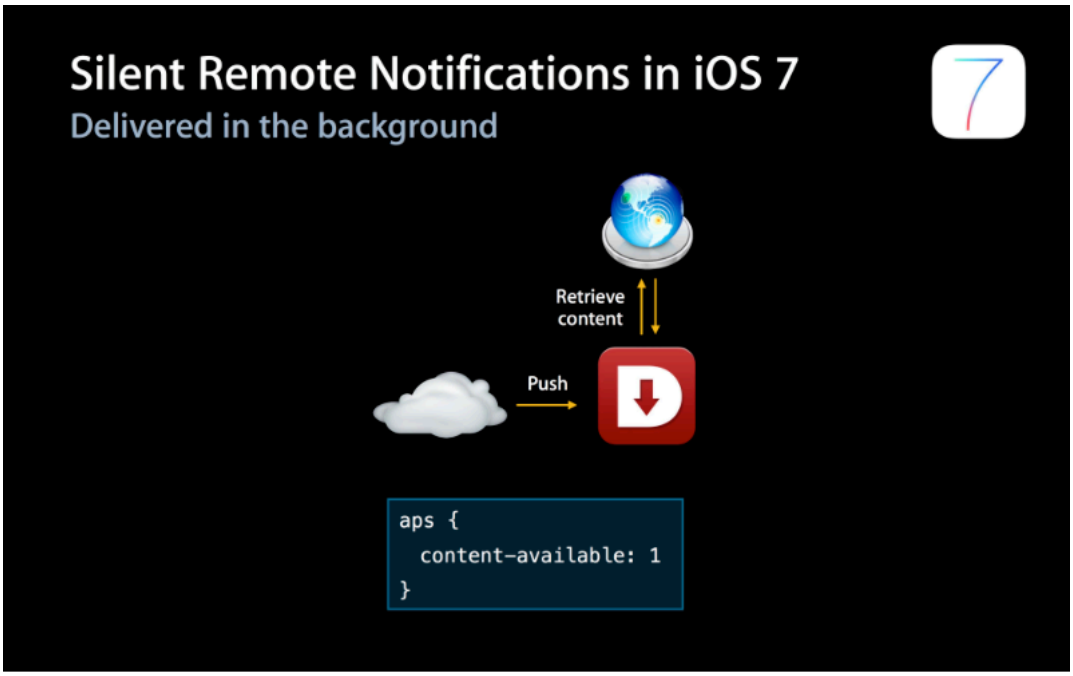
iOS 7在推送方面最大的变化就是允许应用收到通知后在后台状态下运行一段代码，可用于从服务器获取内容更新。功能使用场景：（多媒体）聊天，Email更新，基于通知的订阅内容同步等功能，可以提升用户的体验效果。

ios7远程通知与之前版本的对比可以参考下面两张 Apple 官方的图片便可一目了然。

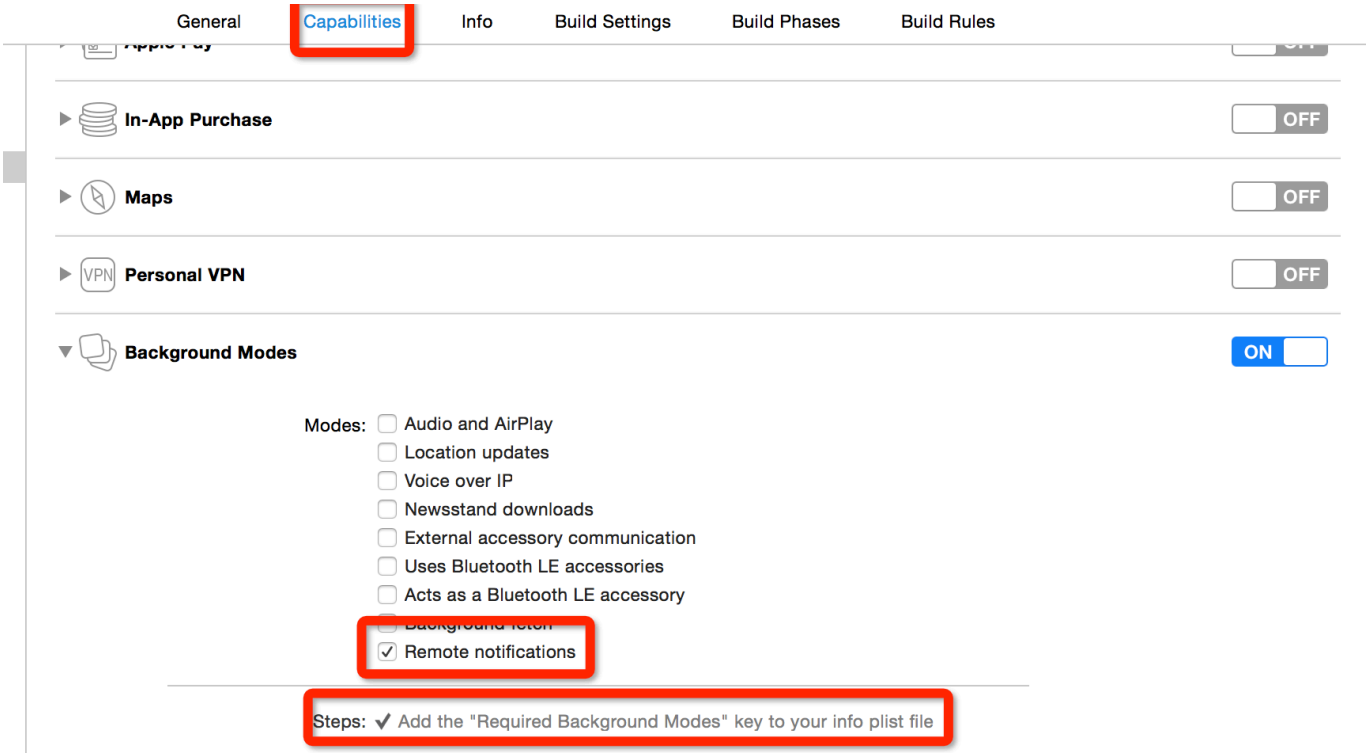




如果只携带content-available: 1 不携带任何badge，sound 和消息内容等参数，则可以不打扰用户的情况下进行内容更新等操作即为“Silent Remote Notifications”(静默推送)。



客户端设置开启Remote notifications,需要在Xcode 中修改应用的 Capabilities 开启Remote notifications，(红框内的必须要做哦)请参考下图：



iOS7之后如果像上面一样设置好了，并且服务端aps字段中添加content-available字段为1的话，了那么收到远程通知后，应用在后台的话会在下面的方法中接受到远程通知，对应程序中的代码是（注意，iOS7之后没有开启后台的话也可以通过点击通知调起下面的方法只是不能在后台状态下运行一段代码）：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
```

iOS7之前没有后台运行代码的功能，只能在接受到通知，并点击通知调起下面的方法，对应程序中的代码是：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
```

iOS9特性，远程通知需要在注册方法中设置leftAction，并且设置category，需要与服务端aps的category字段匹配。设置behaviorTextInput值为YES. 回调的方法iOS9使用了两个新的回调方法来处理点击按钮的事件:

```
- (void)application:(UIApplication *)application handleActionWithIdentifier:(nullableNSString *)identifier forLocalNotification:(UILocalNotification *)notification withResponseInfo:(NSDictionary *)responseInfo completionHandler:(void(^)())completionHandler NS_AVAILABLE_IOS(9_0)

- (void)application:(UIApplication *)application handleActionWithIdentifier:(nullableNSString *)identifier forRemoteNotification:(NSDictionary *)userInfo withResponseInfo:(NSDictionary *)responseInfo completionHandler:(void(^)())completionHandler NS_AVAILABLE_IOS(9_0)
```

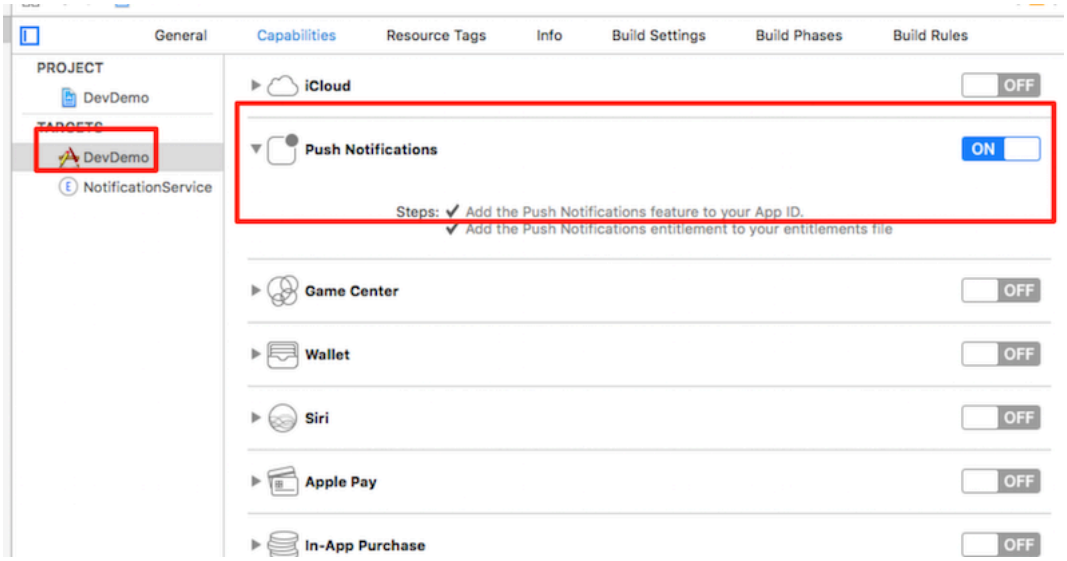
iOS10特性，可以推送富媒体通知，需要做如下处理： 服务端添加aps字典内添加 mutable-content 值为1时，作为富媒体通知的标识，在用户自定义字段添加需要下载的富媒体链接。如下图：

```
{
"aps":{
    "alert":{
        "title":"Hello_title",
        "subtitle":"Hello_subtitle",
        "body":"Your message Here"
        },//字典格式
    "sound":"test.caf",
    "badge":20,
    "content-available":1, //静默通知标识
    "mutable-content":1 //富媒体通知标识
},
"custom_content":{
        "myname":"qiangzi",
        "myage":"30"
        },//用户自定义字段
"imageAbsoluteString":"http://ww1.sinaimg.cn/large/65312d9agw1f59leskkcij20cs0csmym.jpg”//富媒体attachment字段
}
```

客户端需要在工程内添加新的 target（通知扩展） 继承 UNNotificationServiceExtension 在

```
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentHandler:(void (^)(UNNotificationContent * _Nonn
```

回调方法内进行处理。具体处理参考 Demo 的 NotificationService 类。 xcode8 中需要注意打开push能力开关，如下图：



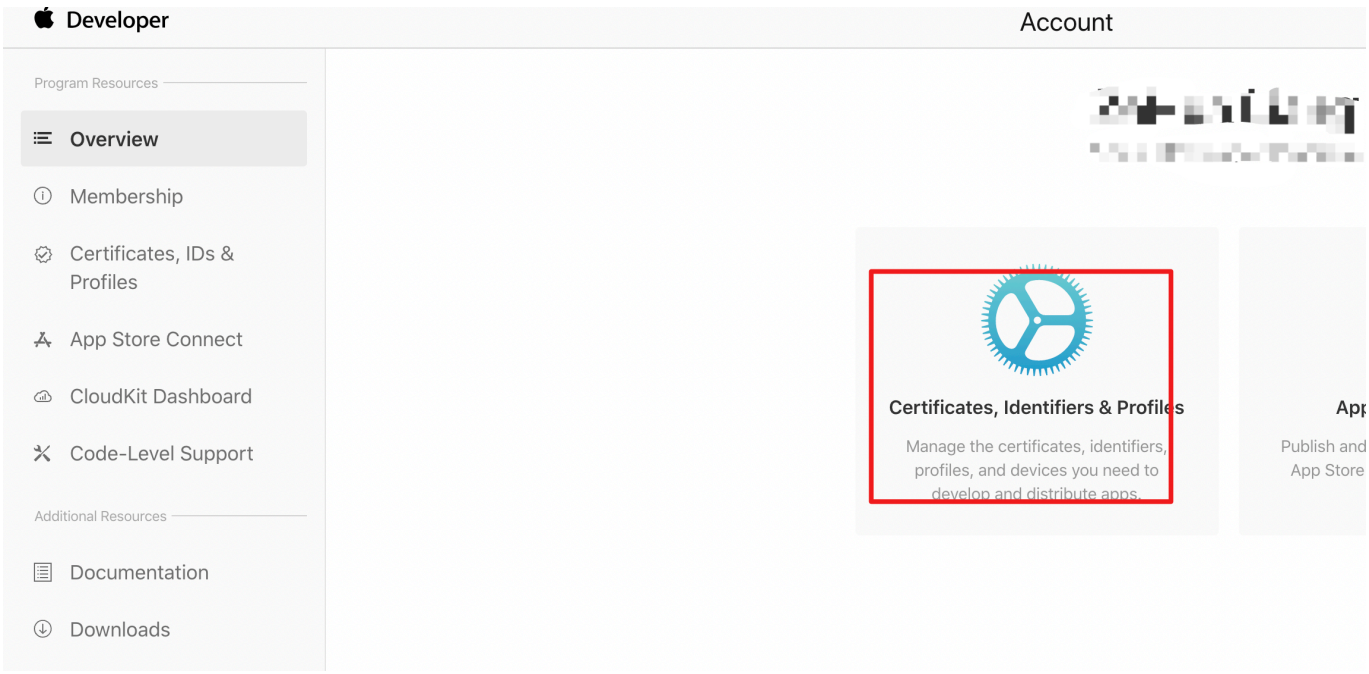
第七章 iOS证书指导

在 iOS App 中加入消息推送功能时，必须要在 Apple 的开发者中心网站上申请推送证书，每一个 App 需要申请两个证书，一个在开发测试环境下使用，另一个用于上线到 AppStore 后的生产环境。

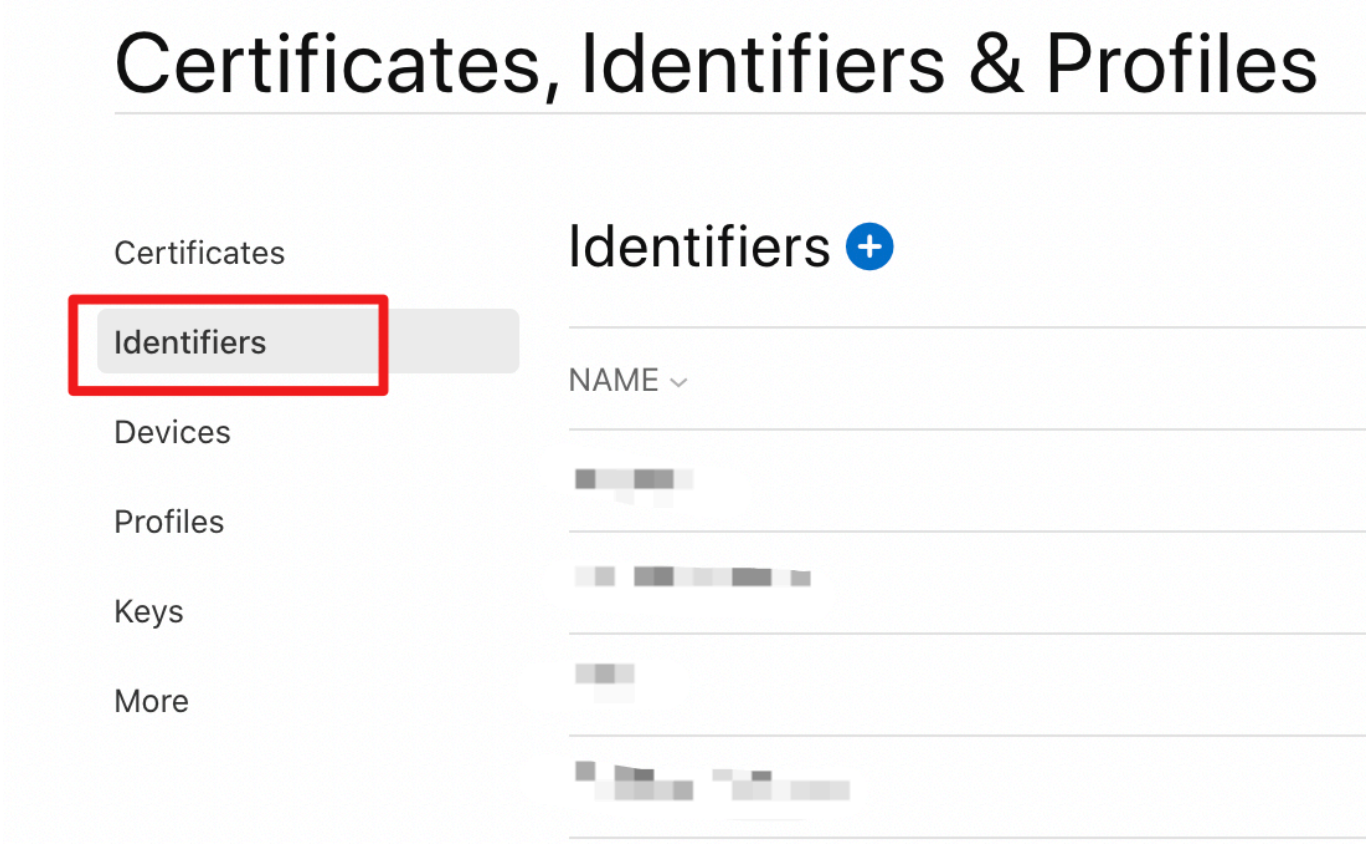
7.1 为你的 App 创建 App ID

iOS 中每个 App 都需要对应一个 App ID，同一个公司可能会使用类似于 `com.example.*` 这样通用的 App ID，但是如果要在 App 中加入消息推送功能，那么是不能使用通用 ID 的，需要为之单独创建一个。

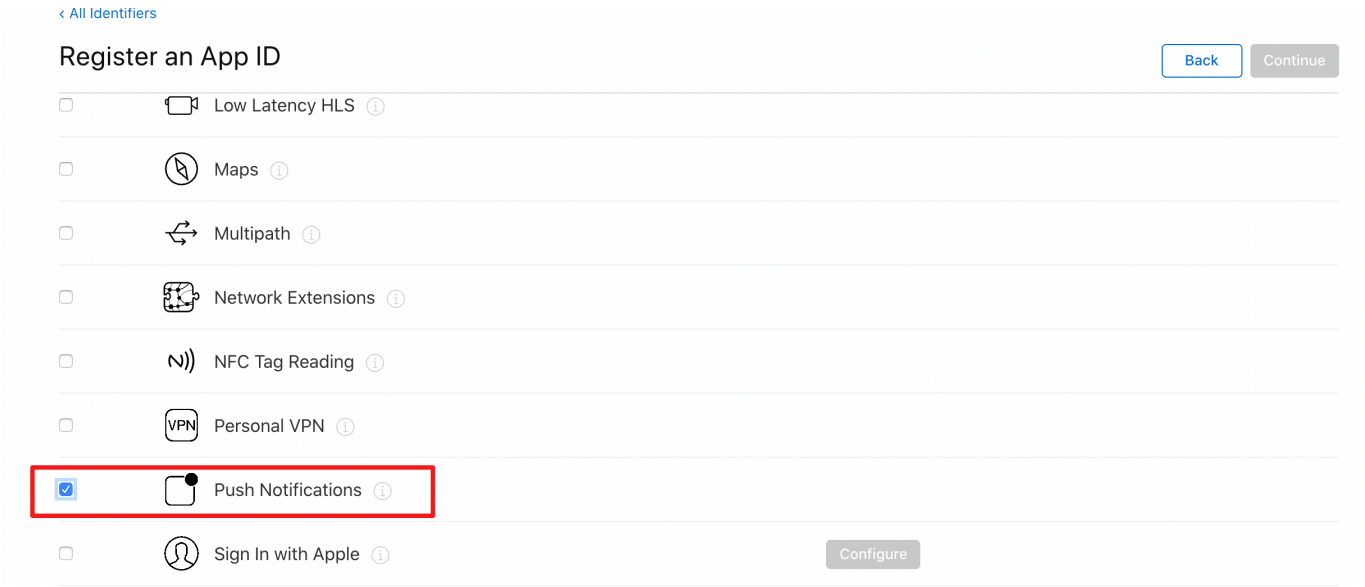
首先登陆 iOS Dev Center ，然后进入 Member Center，然后选择 Certificates，Identifiers & profiles，如下图：



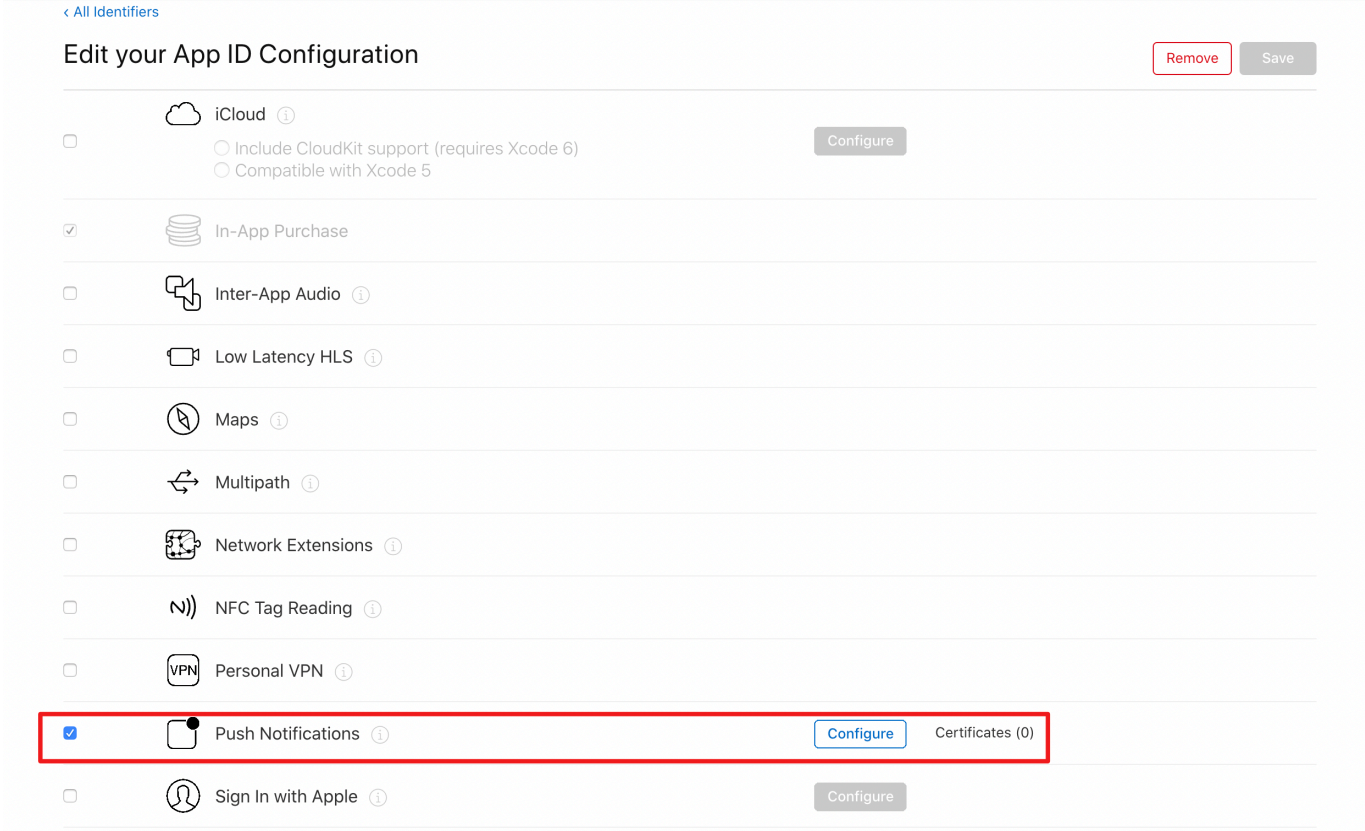
然后点击下图红框中的Identifiers，进入创建App ID界面，如下图：



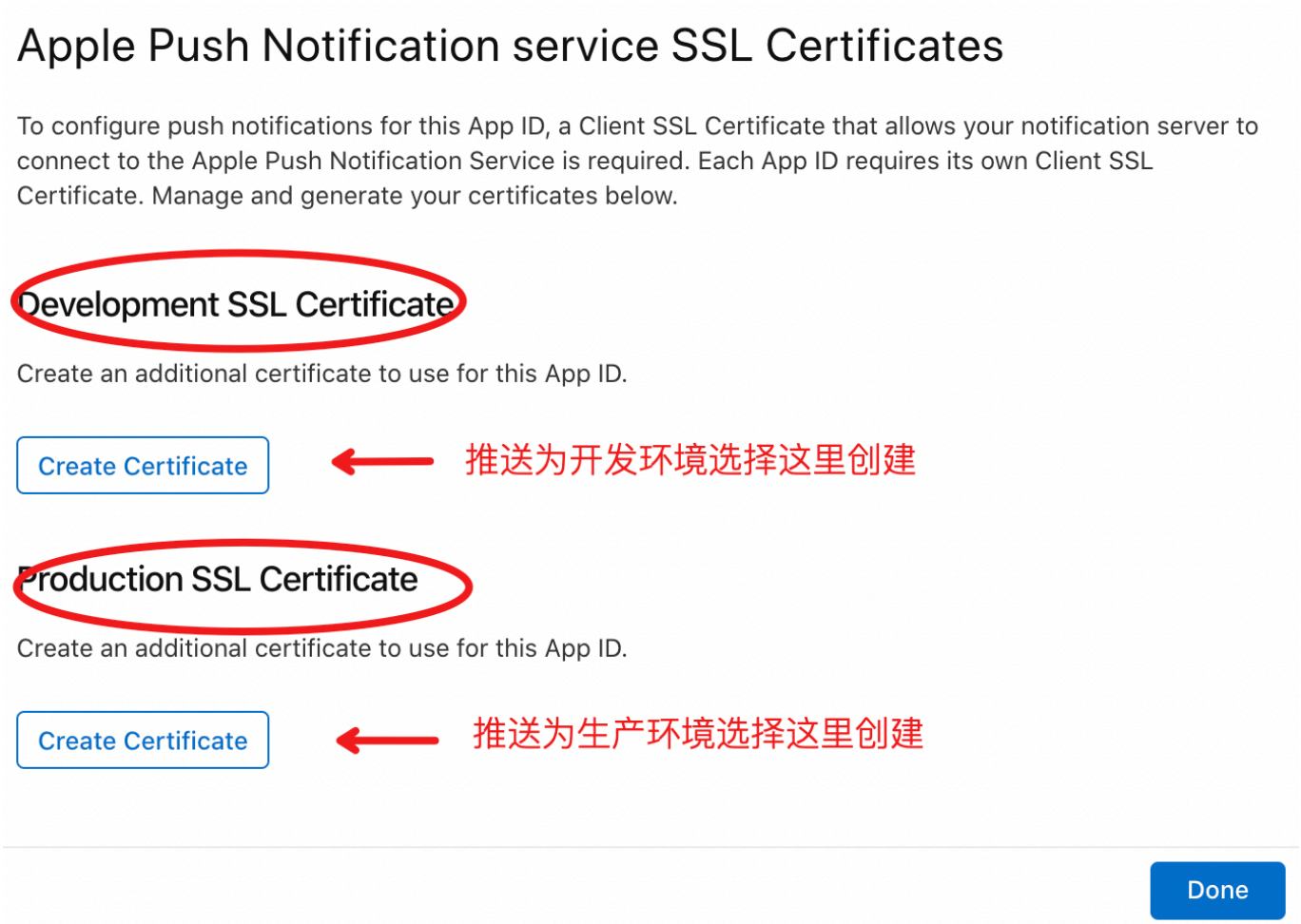
在创建 App ID 的过程中，需要勾选 Push 服务，如下图：



创建完App ID后，点击你创建的APP ID，可以看到Push Notifications变为可编辑状态，如下图：



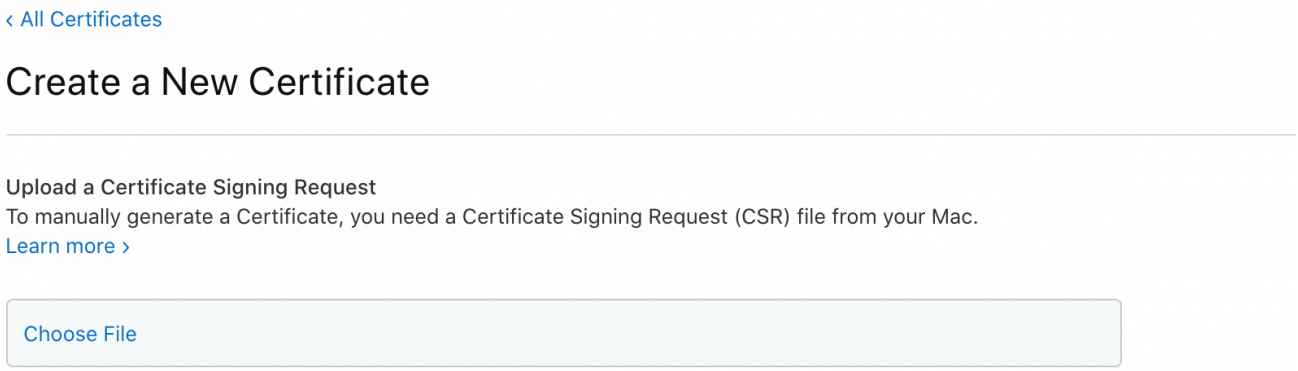
然后配置好对应的推送环境，个人版和企业版的开发环境都是选择创建Development SSL Certificate类型的。个人版和企业版的发布环境。发布环境分以下三种：1.in-house必须是企业开发账户（企业内）（299美金） 2.ad-hoc个人账户或公司Company账户（99美金），但只用于内部测试（总共100个设备）.3.上线Appstore只能是个人账户或公司Company账户（99美金））如下图:



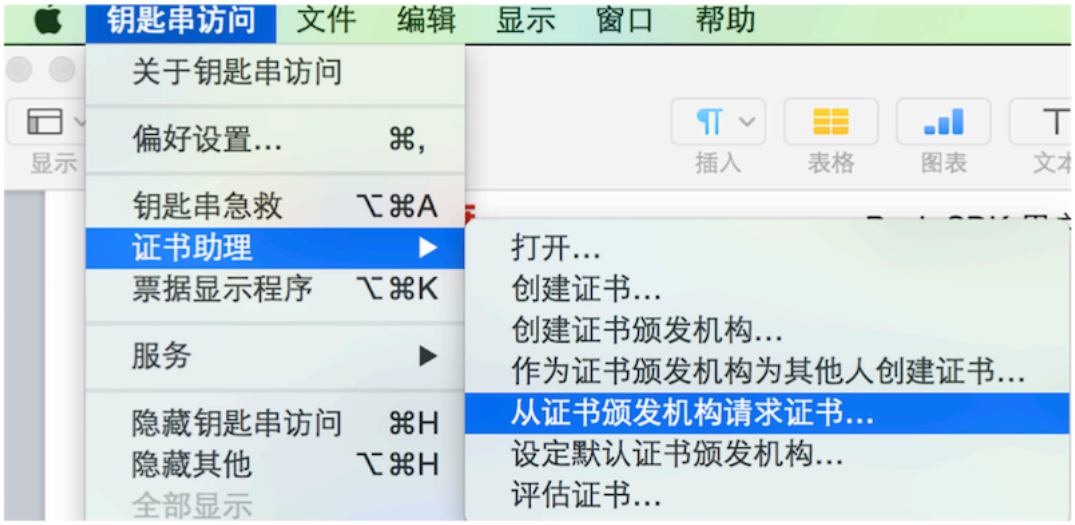
如果你是为已有的 App 增加消息推送功能，那么打开原有的 App ID，开启 Push Notification 选项即可。流程跟上面的一样。

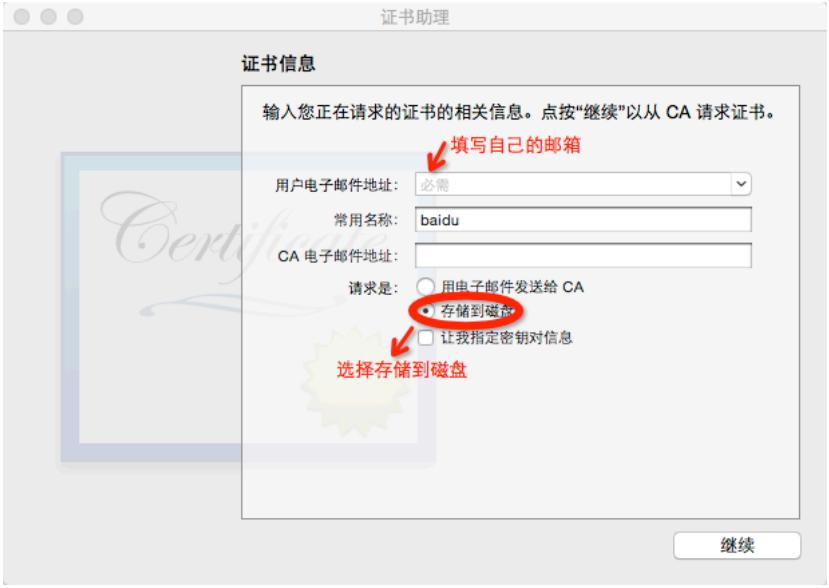
7.2 创建及下载证书

点击 Create Certificate按钮后会出现“About Creating a Certificate Signing Request (CSR)”，如下图：

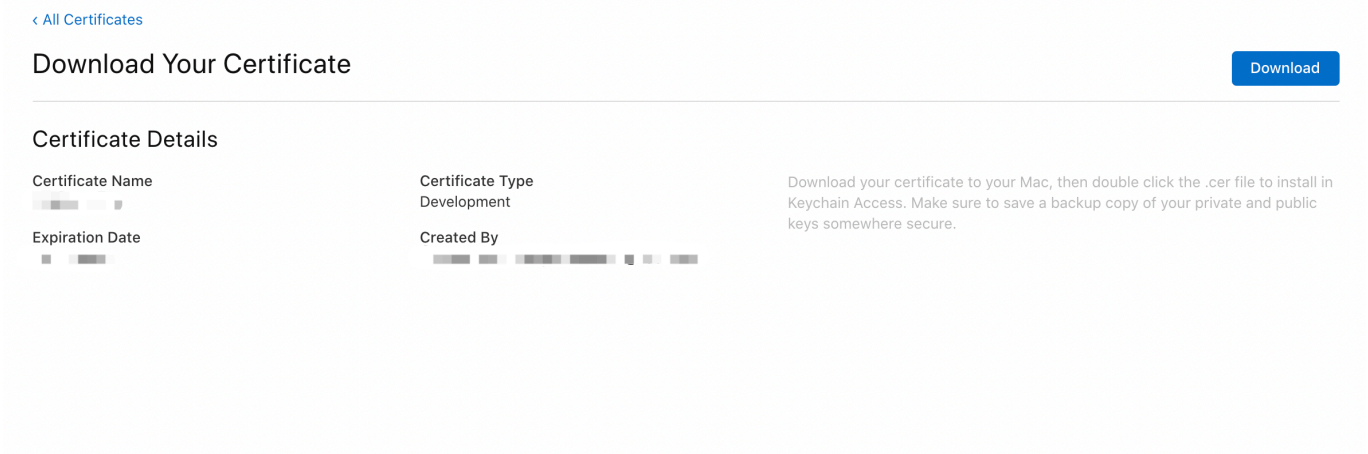


到了这里，需要停下制作 CSR 文件，制作过程比较简单，下面是制作的过程。打开 Mac 系统软件'钥匙串访问'，选择 '证书助理' 及 '从证书颁发机构请求证书'，制作 CSR 文件，如下图:



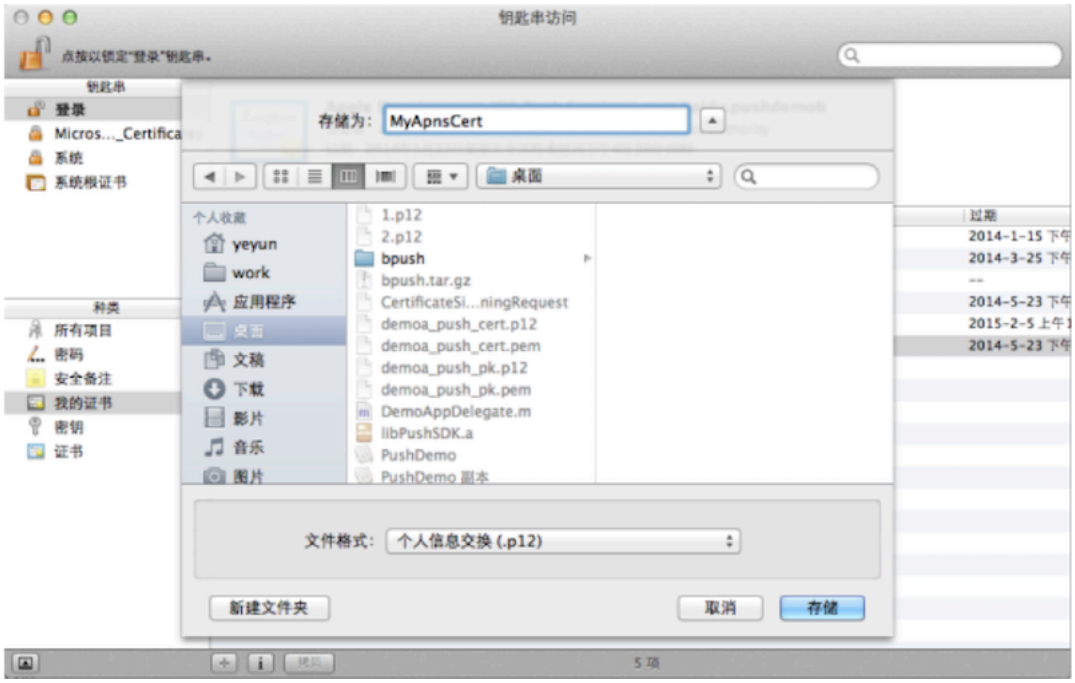


生成证书后，返回到 “About Creating a Certificate Signing Request (CSR)” 的界面，点击 “Choose File” ,选择生成的CSR文件，最后点击 Continue，生成证书。如下图:



现在证书制作已经完成。下载并双击用“钥匙串访问” 程序打开后，在左边一栏，上面选择登录，下面选择证书，然后选择刚刚打开的证书，切记不要展开它，直接右击导出p12，如下图:

将文件保存为 .p12 格式，输入密码，如图所示：

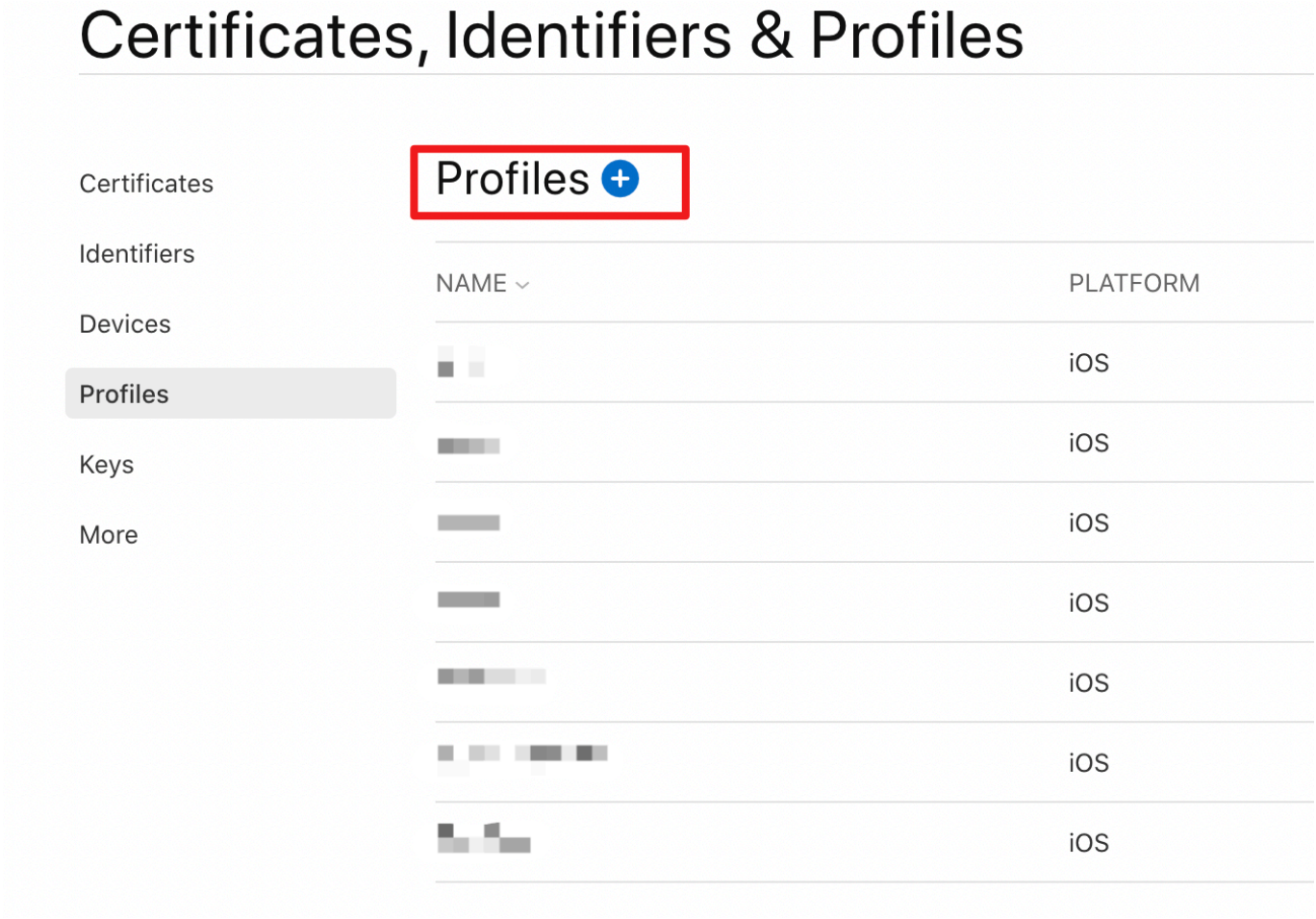


最后进入终端，到证书目录下，运行以下命令将p12文件转换为pem证书文件：

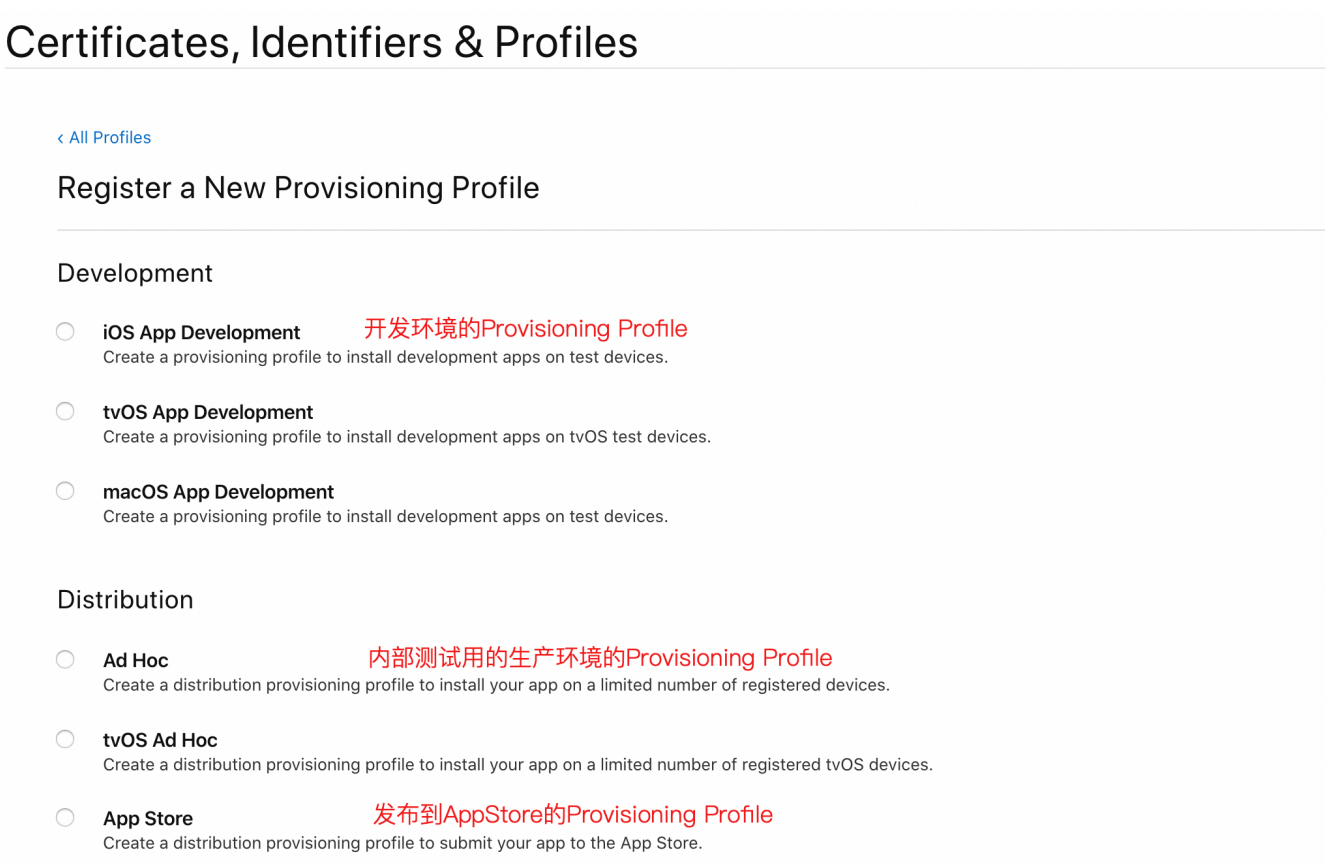
```
openssl pkcs12 -in MyApnsCert.p12 -out MyApnsCert.pem -nodes
```

提示需要输入密码，输入刚才导出 p12 时的密码即可。

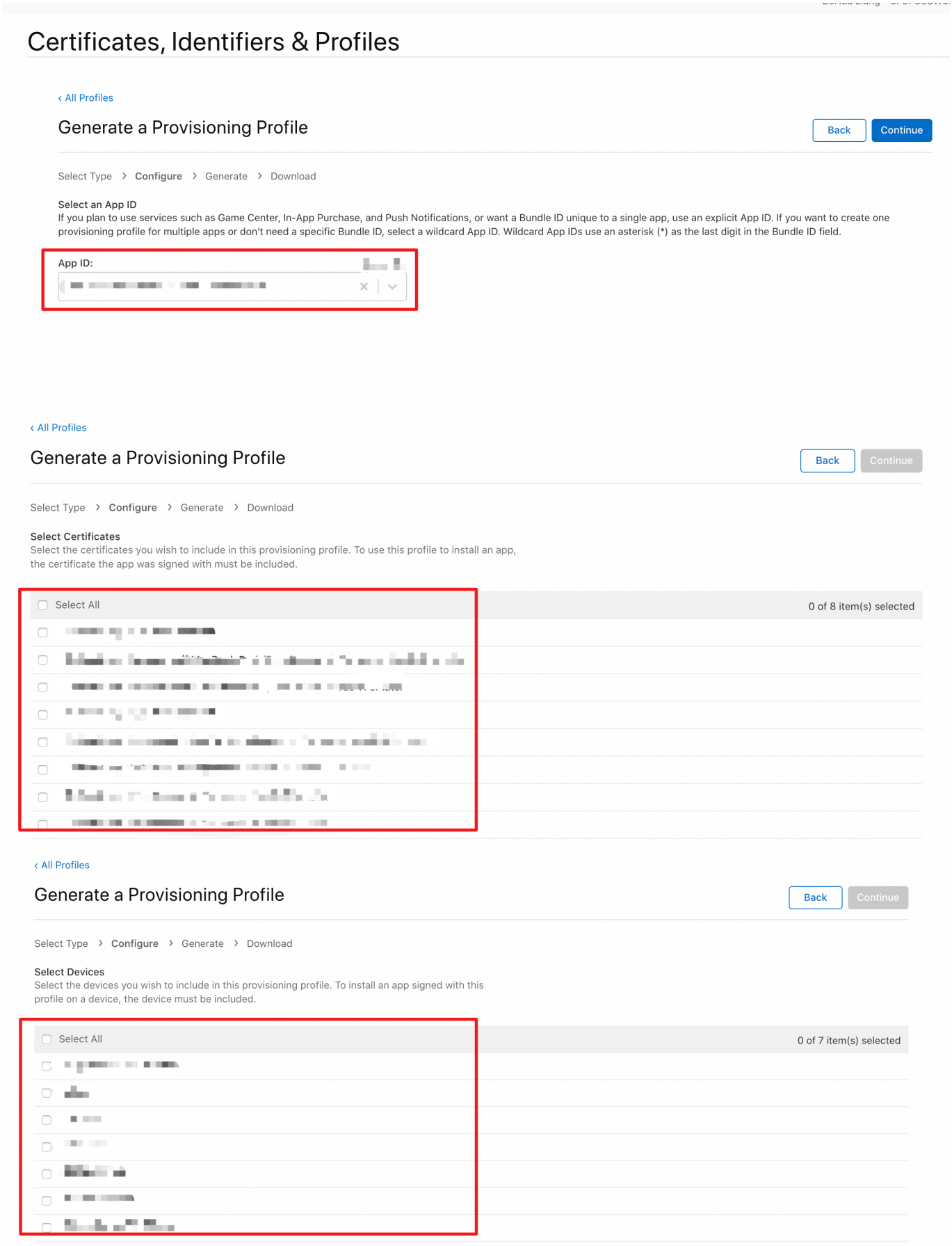
Provisioning Profile的创建 点击下图的+按钮开始创建profile



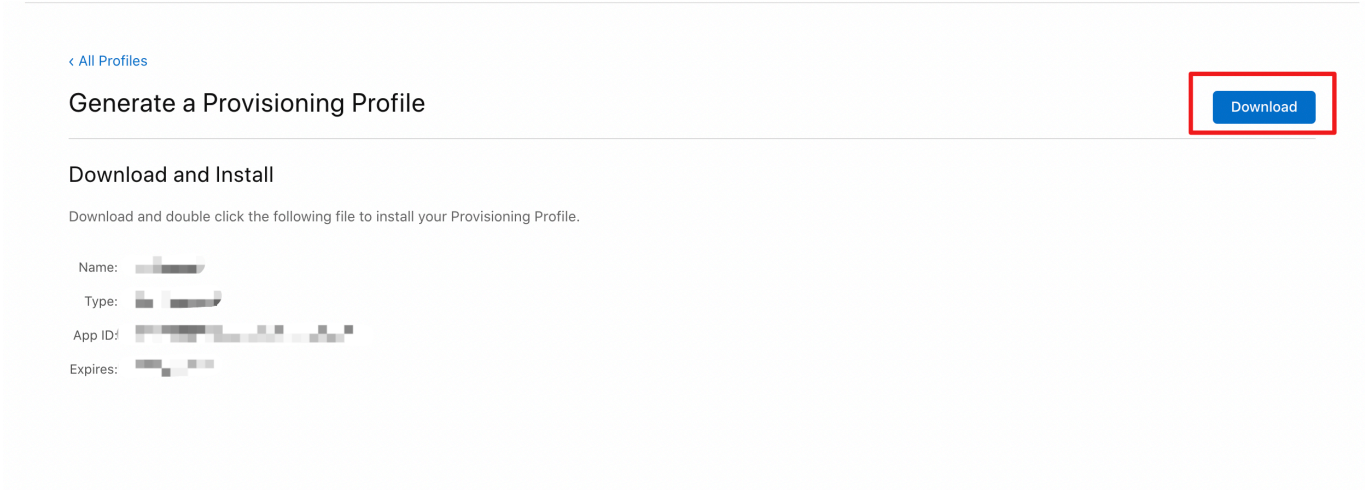
选择profile的环境



选择创建profile的appid和开发者证书，并选择设备，最后生成profile



最后下载profile配置到Xcode中进行开发测试



第八章 iOS推送简介

在移动应用中，推送已经成为不可缺少的重要功能。本文档主要介绍有关 iOS 推送的基础知识，并从几个典型问题出发，分析如何解决在实现推送中出现的一些问题。

8.1 本地通知和远程通知简介

在 iOS 设备上（模拟器无法使用推送），系统收到通知后这样处理：

- 在屏幕上弹出一些选项，或者在屏幕顶部显示横幅（banner）如下图左
- App 的角标数值发生变化，具体表现为 App icon 右上角的小红点及数字，如邮件中的红点
- 伴随推送消息的提示声音

当应用处于前台运行时，系统是不会在屏幕上显示通知，但是仍会调用相应的 API。

只有真机可以使用推送功能。

用户可以设置每一个 App 的通知权限，如下图最后一个：

用户可以选择关闭某个应用的推送功能。还可以设置通知是否在通知中心显示、通知到达时是否发出声音、通知能否改变 App 角标以及锁屏时是否显示该 App 的通知，还可以设置通知到达时的提醒样式。当然，这些都可以分别对每一个应用进行单独设置。

8.1.1 本地通知

本地通知是一种基于时间的提醒方式。

本地通知最多向系统注册 64 个，当超过这个数量后，最早注册的本地通知会被丢弃。

本地通知在 iOS 设备上的显示与远程推送通知一样。

8.1.2 远程通知

iOS App 运行在后台时，无法主动进行网络连接。

远程通知可以用来提醒用户。发送远程通知时，服务器首先需要使用推送证书与 APNs（Apple Push Notification service）建立安全连接，然后将消息传递给它。当 APNs 收到消息后，会通过与手机之间的长连接下发到对应的手机上，然后 iOS 弹出这条消息来提醒用户。

用户看通过点击或滑动通知来运行 App。可通过程序中相应的方法可以获取通知信息，然后做相应的逻辑处理。

如果不是通过通知启动 App，那么无法在程序中获取通知信息。例如通过点击应用图标启动 App。

推送通知都包含 Payload：一个 Apple 已经定义好的属性列表，操作系统根据它决定使用哪一种方式来提醒用户。还可以在 Payload 中加入一些自定义数据。

通知并不能一定到达。如果在用户无法收到推送通知时（关机或网络不可用），这种情况下 APNs 收到了多条发往这台设备的通知，那么只会保留最后一条，最早的会被丢弃。

APNs 首先通过移动蜂窝网络发送通知，只有当移动蜂窝网络不可用时才使用 Wi-Fi。

8.2 通知的两种推送环境

在使用 iOS 远程推送功能时，有两种不同的环境。开发环境（Development）以及生产环境（Production）。

App 当前使用的推送环境与 Xcode - Build Settings - Code Signing - Provisioning Profile 文件的模式一致。

8.2.1 证书与证书校验

与 APNs 之间是加密的连接，因此需要使用证书来加密连接。每个的推送环境有自己单独的推送证书，即开发证书和生产证书。

在将证书最终转为 pem 格式后，可通过与 APNs 连接来测试证书是否有效。

开发环境：

```
openssl s_client -connect gateway.sandbox.push.apple.com:2195 -cert MyApnsDev.pem
```

生产环境：

```
openssl s_client -connect gateway.push.apple.com:2195 -cert MyApnsPro.pem
```

当输入完命令回车后，终端首先会输出很多相关信息。

当连接建立失败时，会直接关闭。

当连接建立成功时，终端会停止输出，并等待你输入，你可以随便输入一些字符后摁回车，然后连接才会关闭。

以上命令在 Mac 下没有问题，在其他操作系统下需要指定服务器当前的 CA 根证书，具体可以从网站上下载：

entrust*2048*ca.cer 下载

下载完成之后，在以上命令后面加上 -CAfile entrust2048ca.cer 即可

8.2.2 DeviceToken

通知需推送到具体某一台设备，而 DeviceToken 就是这台设备的标识符。在向 APNs 发送推送通知时，需要使用 DeviceToken 来指定这条通知将要到达的设备 。

每一台设备，不同的推送环境下分别有一个 DeviceToken。即 DeviceToken 也分开发环境和生产环境。

应用在向 APNs 注册推送通知时，会根据当前 Xcode 工程中 target 对应的 Provisioning Profile 决定请求对应环境的 DeviceToken。即系统返回的 DeviceToken 其环境与 Provisioning Profile 的环境对应。

一般在使用 Xcode 开发时，Provisioning Profile 为 Development ，因此获取的 DeviceToken 也是开发模式。

在 App 需要打包为 ipa 文件或者需要上传到 AppStore 时，会将 Provisioning Profile 文件修改为 Distribution，这时获取到的 DeviceToken 是生产模式。

8.3 在应用程序中注册远程推送功能

App 必须要向 APNs 请求注册以实现推送功能，在请求成功后，APNs 会返回一个设备的标识符即 DeviceToken 给 App，服务器在推送通知的时候需要指定推送通知目的设备的 DeviceToken。在 iOS 8 以及之后，注册推送服务主要分为四个步骤：

1. 使用 `registerUserNotificationSettings:` 注册应用程序想要支持的推送类型
2. 通过调用 `registerForRemoteNotifications` 方法向 APNs 注册推送功能

3. 请求成功时，系统会在应用程序委托方法中返回 DeviceToken，请求失败时，也会在对应的委托方法中给出请求失败的原因。
4. 将 DeviceToken 上传到服务器，服务器在推送时使用。

上述第一个步骤注册的 API 是 iOS 8 新增的，因此在 iOS 7，前两个步骤需更改为 iOS 7 中的 API。

DeviceToken 有可能会更改，因此需要在程序每次启动时都去注册并且上传到你的服务器端。

> 注意：iOS 设备与 APNs 需要建立一条长连接，之后的推送注册以及推送获取都需要通过这条长连接。
>
> 当 iOS 设备无网络可用时，在向 APNs 请求注册后，既不会有注册成功的回调，也不会有注册失败的回调。如果发生这种情况，需要检查设备的网络连接。
>
> 开发环境和生产环境建立不同的长连接，且设备上要至少有一个开发环境的 App 并且注册推送，才会建立开发环境下的长连接。
>
> 当已经注册过推送服务后，以后系统会立刻返回 DeviceToken。还有一点是，返回 DeviceToken 的回调方法并不一定是只有在你注册或者再次注册时才被系统调用，当DeviceToken 改变时也会被系统直接调用。

8.4 在程序中处理通知

我们来看一下当系统传递本地或远程通知的时候都有哪些情况。

通知到达时，应用不在前台。

这种情况下，操作系统将呈现通知，弹出一个选项或者在屏幕上部显示 banner 提醒，给 App 角标设置对应的数字，可能播放提示音，当用户稍微往下滑动通知时显示更多的 action 按钮（如果有设置）。

通知到达时，应用在前台运行。

这种情况下，不会弹出 banner。系统会根据当前的通知是本地通知还是远程通知，调用不同的方法，并且在方法中传递通知的 Payload 数据。

在 iOS8 的通知中，用户点击通知上的自定义 action 按钮进入应用。

这种情况下，系统会调用 iOS8 中对 action 按钮新增的处理方法。

用户点击通知进入应用。

系统会根据当前的通知是本地通知还是远程通知，调用不同的方法，并且在方法中传递通知的 Payload 数据。

> 注意：只有当通过通知进入到 App 时，才可以获取到通知信息。

8.4.1 自定义通知提示音

你可以在 App 的 Bundle 中加入一段自定义提示音文件。然后当通知到达时可以指定播放这个文件。必须为以下几种数据格式：

- Linear PCM
- MA4（IMA/ADPCM）
- μLaw
- aLaw

你可以将它们打包为 `aiff` 、 `wav` 或 `caf` 文件。自定义的声音文件时间必须小于 30 秒，如果超过了这个时间，将被系统声音代替。

8.4.2 Payload

Payload 是通知的一部分，每一条推送通知都包含一个 Payload。它包含了系统提醒用户通知到达的方式，还可以添加自定义的数据。即通知主要传递的数据为 Payload。

Payload 本身为 JSON 格式的字符串，它内部必须要包含一个键为 `aps` 的字典。aps 中可以包含以下字段中的一个或多个：

- alert：其内容可以为字符串或者字典，如果是字符串，那么将会在通知中显示这条内容
- badge：其值为数字，表示当通知到达设备时，应用的角标变为多少。如果没有使用这个字段，那么应用的角标将不会改变。设置为 0 时，会清除应用的角标。
- sound：指定通知展现时伴随的提醒音文件名。如果找不到指定的文件或者值为 default，那么默认的系统音将会被使用。如果为空，那么将没有声音。
- content-available：此字段为 iOS 7 silent remote notification 使用。不使用此功能时无需包含此字段。

第九章百度云推送故障排查指南

使用百度云推送时，可能出现一些问题，你可以根据下面的项目进行排查。

使用百度云推送实现 iOS 推送主要分为三步：

- 制作推送证书。需要分别制作开发环境和生产证书两个证书。然后测试证书是否正确创建并导出。在制作证书的过程中，会为 App 开启推送功能。
- 在百度云推送官网控制台中创建应用。需要为安卓版本和 iOS 版本分别创建应用，然后为 iOS 应用上传已经制作成功的 pem 格式推送证书。
- 在 App 中集成百度云推送 SDK，按照文档中的步骤即可。集成完成后，App 与百度云推送服务器做一次绑定，App 收到绑定成功的回复后，保存回复字典中的 channelID，它是设备的唯一标识。userID 暂时不被使用。

完成以上步骤即可在官网的控制台中对安装你应用的 iOS 设备进行推送。如果还需服务端推送的支持，按照文档集成服务端推送 SDK 即可。

9.1 注册远程推送通知

在获取推送消息之前，需要先向 Apple Push Notification service （APNs）注册推送服务。

1. 应用程序调用注册推送服务的 API，需要注意 iOS 8 之前和之后需要使用不同的 API。
2. 应用程序实现 `UIApplicationDelegate` 委托的 `application:didRegisterForRemoteNotificationsWithDeviceToken:` 回调方法，用于获取 APNs 返回给设备的 DeviceToken。
3. 应用程序实现 `UIApplicationDelegate` 委托的 `application:didFailToRegisterForRemoteNotificationsWithError:` 回调方法，用于当获取不到 DeviceToken 时，查看其错误原因。

然后应用程序将 DeviceToken 传递给云推送。

注册推送服务比较简单，但是仍然有几点比较重要你需要注意。

9.2 注册百度云推送服务

在注册百度云推送服务时，一定要根据当前应用所处的环境，正确设置注册方法的 PushMode 参数：

```
+ (void)registerChannel:(NSDictionary *)launchOptions apiKey:(NSString *)apikey pushMode:(BPushMode)mode withFirstAction:(NSString
```

在开发时使用 Development 模式，在上线前一定需要修改为 Production 模式。如果上面方法的 PushMode 设置错误，那么无法收到通知。

如果在开发时发现设置错误，那么在修改模式之后，需要做一次解除绑定操作再进行绑定。即 PushMode 参数每一次修改后，都需要调用一次 unBind 方法来清除缓存。

> requestid 为 88888888 时，表示此次请求的回复是从缓存中获取的。在修改 PushMode 参数后，需要调用 unbind 方法或删除应用重新安装来清除缓存。

9.3 没有委托回调

当含有推送功能的 App 第一次安装时，iOS 尝试与 APNs 建立一条长连接供所有操作系统上具有推送能力的 app 使用。如果 `application:didRegisterForRemoteNotificationsWithDeviceToken` 和 `application:didFailToRegisterForRemoteNotificationsWithError` 这两条回调方法都没有被调用。这意味着 iOS 设备与 APNs 之间没有成功建立起长连接。

这可能没有任何错误提示。系统可能由于蜂窝网络无信号或距离 Wi-Fi 点太远，或者开启了飞行模式。

要注意到网络状态是经常变化的，一旦长连接建立起来了，之前提到的回调方法就会被调用。

在 iOS 上，推送通知首先检查移动蜂窝网络是否能正常使用，即使当前已经连接了 Wi-Fi，只有当移动蜂窝网络无法使用时才会使用 Wi-Fi。

如果你的设备支持移动网络，那么将检查它是否有一个活动的蜂窝数据计划。在设备的【设置】中，关闭 Wi-Fi 选项，然后查看是否能够使用 Safari 浏览器打开网页等。另一方面，如果推送服务使用 Wi-Fi 服务，那么需要确保 Wi-Fi 热点防火墙允许 端口为 5223 的 TCP 通信。

> 注意：不同的推送环境使用单独的长连接，操作系统建立与 Sandbox 环境的长连接用于开发。ad hoc 和生产模式会连接到生产环境下。

9.4 回调错误的 aps-environment：需重新生成 provisioning profile 文件

支持推送服务的 iOS App 在第一次运行时，iOS 会询问用户是否想要接收这个 app 的推送通知。如果这个 App 之前没有提供接收推送通知的选项，那么它将失去 `aps-environment` 代码签名授权的机会。这种资格是在 building app 的时候从 provisioning profile 中得到，它控制了为了接收推送通知时，app 应该使用哪一种推送环境去建立长连接。

如果 iOS app 运行在设备上，在获取推送服务时发现代码签名授权获取失败，则 `application:didFailToRegisterForRemoteNotificationsWithError` 方法会被调用。如果你没有实现这个方法，那么你应该使用它来检查你的 `aps-environment` 授权是否在运行时获取到。

如果没有发现对应的代码签名授权，那么说明你还没有配置 app ID 开启推送服务，或者开启后没有更新 Provisioning Profile 文件，需要去 Member Center 去更新。

你需要在 Member Center 中为 App ID 配置开启推送服务后重新创建 provisioning profile，如果你 Xcode 中的 provisioning profile 是未配置前的，那么需要将其删除后重新生成新的并添加到 Xcode。

更新完这些后，你需要 rebuild 你的 app。

9.5 注册成功但收不到通知

如果你的 app 注册推送服务成功，并且能获取到 DeviceToken，但是收不到通知。那么有可能是你设备的问题，也有可能是发送通知的服务器的问题。下面简述了如何排查这些问题。

设备有可能断掉了与 APNs 的长连接且不能重新建立，因而无法通过长连接将通知发送到你的 iOS 设备上。尝试关闭 app 并且重启设备看是否解决了问题。（在 iOS4 之后，app 可以后台运行，因此你需要强制杀死 app）。如果重启后 app 的注册服务不能完成，即获取不到 DeviceToken，那么 iOS 设备还是不能与 APNs 建立长连接。你需要按照之前章节中介绍的内容再次排查问题。

如果可以获取 DeviceToken，你需要上传对应的 DeviceToken 到百度云推送，使用云推送 SDK 中的 `RegisterDeviceToken` 方法即可 。在 App 每次启动注册推送服务的时候，它总是会向 APNs 请求 DeviceToken，所以你不需要保存 DeviceToken 并且尝试复用它，因为它有时候会改变。

> 注意：系统每次启动都会通过 `application:didRegisterForRemoteNotificationsWithDeviceToken` 方法返回设备的 DeviceToken，你需要在这个方法中调用云推送的 `RegisterDeviceToken` 方法。

9.6 只有部分通知到达

你如果在很短的一段时间内，发送大量通知到同一台设备，那么 Apple 的 Push 服务将只会发送最后一个通知。原因如下，设备收到的每一条通知后都会进行确认。在 APNs 收到这个确认信息前，它只能假设设备已经离线，然后将通知存储在质量服务（QoS）队列中并尝试再次发送。往返的网络延迟是一个主要原因。

QoS 队列为每一台设备中每一个 App 仅保留一条通知，如果保留的这条通知还没发出去时又收到了一条新的通知，那么旧的通知将被丢弃。

所有的这一切指出，推送通知是提示 App 它所感兴趣的一些事情已经改变，通知不应该包含数据，因为它并不是无论如何都能到达的，并且通知也不应该是具有状态性的。

你的设备无法连接到 APNs 时，任何通知都不能立刻发送成功，需要入队等待以后再次发送。需要考虑网络延迟，最长的情况下 60 秒后 APNs 会认为是超时的。

9.7 单播收不到

在使用百度云推送进行单播推送时，如果一直收不到。首先检查手机的网络连接是否正常。

确认你当前的推送环境，具体可参照推送环境一项。针对不同的推送环境，需要在百度云推送 SDK 中的

`+ (void)registerChannel:(NSDictionary *)launchOptions apiKey:(NSString *)apikey pushMode:(BPushMode)mode withFirstAction:(NSString`
方法中，修改 BPushMode为正确的环境。

卸载应用并且重新创建，根据绑定成功后返回的 channelID 再次进行单播推送。

9.8 广播收不到

在测试广播推送时，如果有部分能收到，那么你的推送服务配置是没有问题的。对于这部分没有收到的设备，首先检查其网络连接，然后尝试对这部分设备推单播，单播收不到时可以按照上面的方法进行排查。

如果广播中所有的用户都无法收到，可以尝试对其中部分用户推送单播，然后按照单播推送的排查方法解决。

9.9 APNs 与 iOS 设备之间的长连接

在 iOS 推送中，通知是通过 Apple 的 APNs 与设备之间的长连接进行推送的。开发环境和生产环境对应不同的长连接，设备上至少有一个开发环境的App，并且这个 App 注册推送服务后，设备才会与 APNs 建立开发环境下的长连接。

第十章 联系我们

如果以上信息无法帮助你解决在开发中遇到的问题，请通过以下方式联系我们： 邮箱：push-support@baidu.com，百度的工程师会在第一时间回复您。

如流官方技术讨论群：2385611