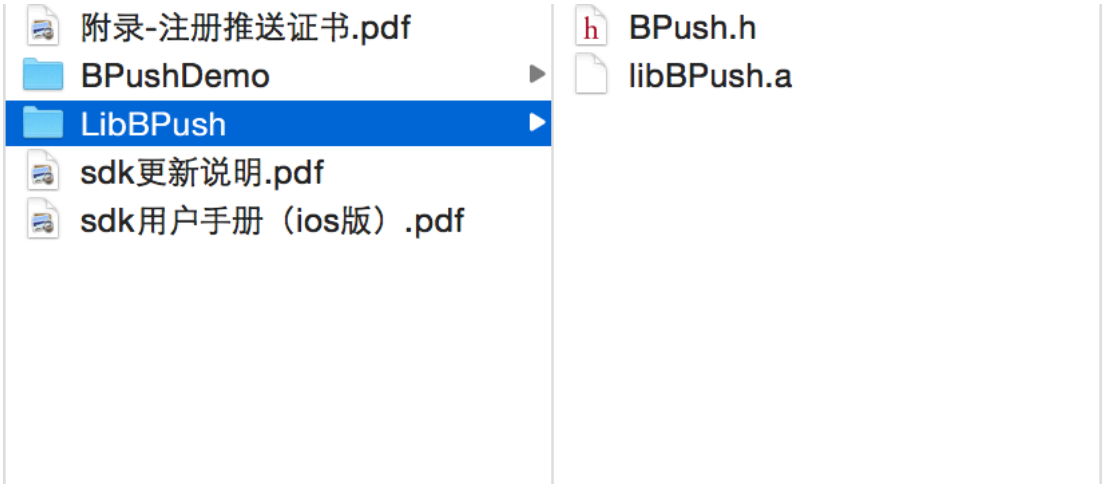
百度Push服务SDK用户手册（iOS版）

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 Baidu-Push-SDK-iOS-L1-1.4.0.zip，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

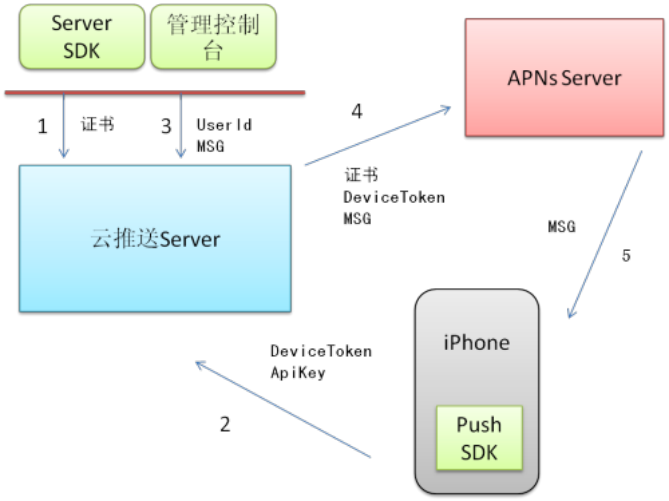
在iOS App 中加入消息推送功能的基本步骤分为如下两步

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

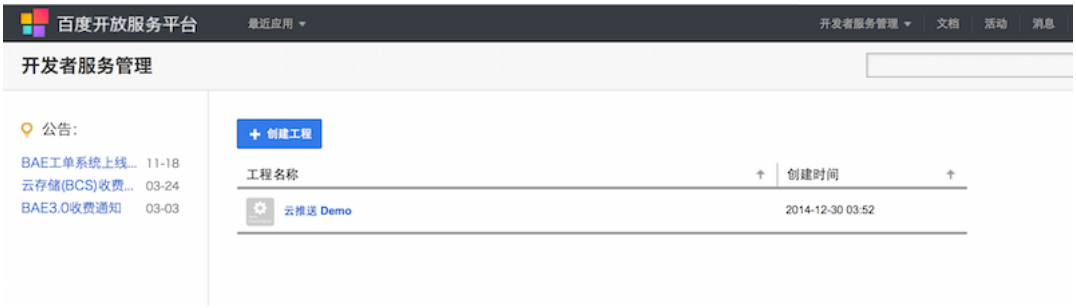
本章节将帮助您如何借鉴SDK包中的Demo源码进行简单的消息推送，快速地使用百度云推送服务。具体操作通过以下几步骤即可完成体验。

3.1 注册成为百度开发者

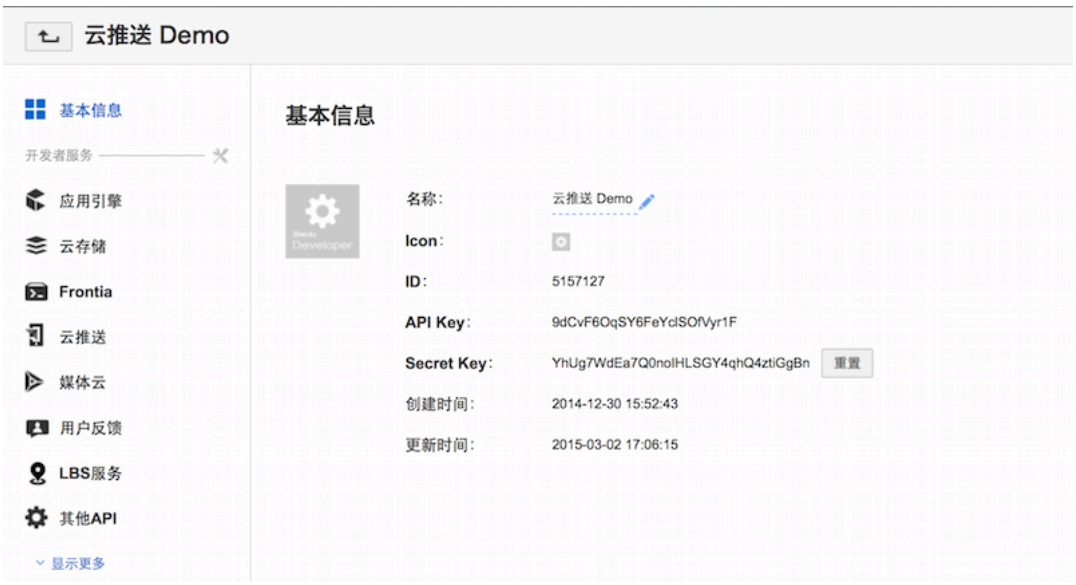
登录百度开放服务平台 ，注册百度账号，并成为百度开发者，(参考开发者注册)，已注册用户可直接登录。页面如下图所示：

3.2 基本设置

登录成功后，可在【管理控制台】- 【开发者服务管理】进入工程总览界面，如图所示：



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



进入左边导航栏，选择“云推送”功能，首次选择时，需要进行【推送设置】，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【附录-申请推送证书】中参考。

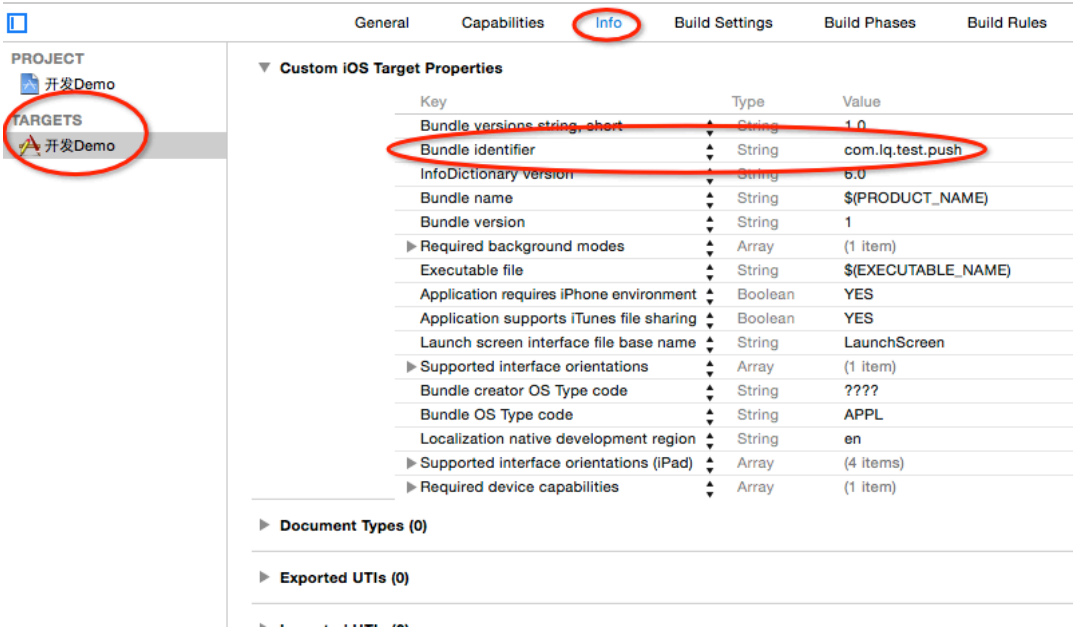
完成以上信息后，点击“保存设置”按钮即可保存信息。

3.3 下载开发工具包

下载地址：iOS SDK下载 下载后，您将得到一个压缩包，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:

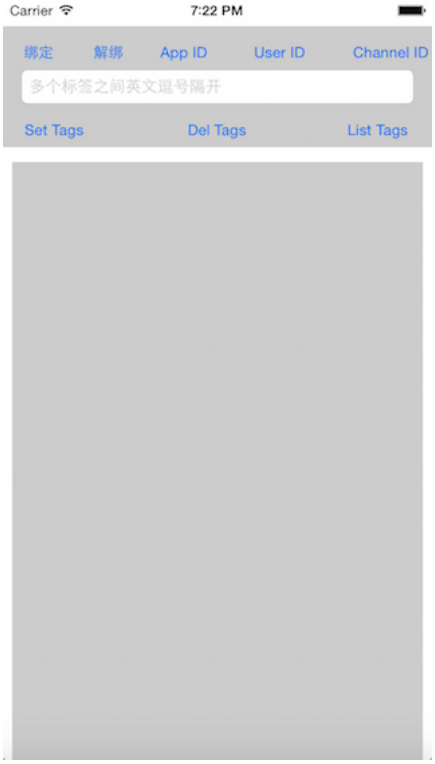


修改下面方法中的apikey的值为自己的apikey,并配置为自己的证书，如下图:

```
    // 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，请在工程中添加。

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 *)leftAction withSecondAction:(NSString *)rightAction withCategory:(NSString *)category isDebug:(BOOL)isdebug
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
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:
(NSString *)leftAction withSecondAction:(NSString *)rightAction withCategory:(NSString *)category 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：快捷回复通知的第一个按钮名字
- rightAction：第二个按钮名字
- ategory 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

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）

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：

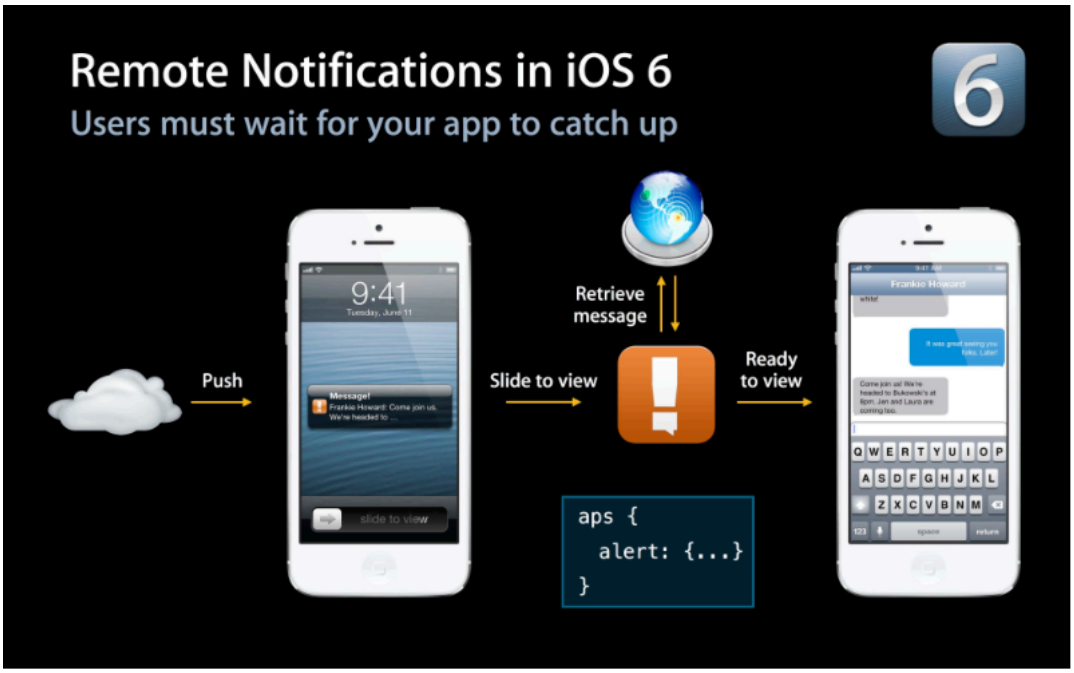
字符串类型的appid、channelid或userid

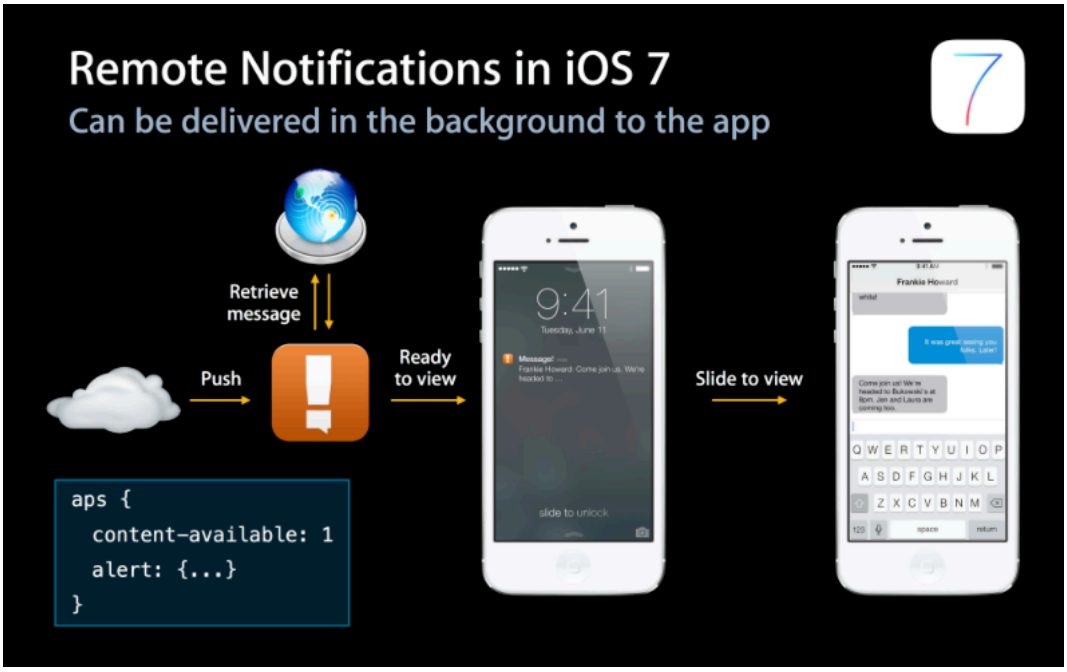
第六章 限制与注意

6.1 iOS7和iOS8特性

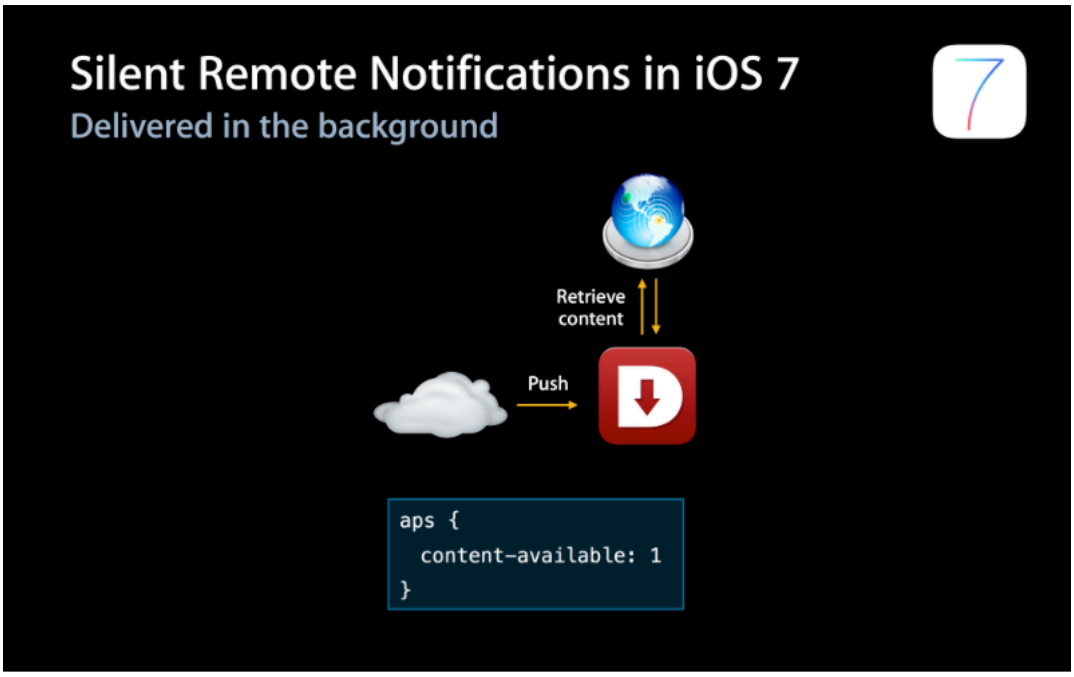
iOS 7在推送方面最大的变化就是允许应用收到通知后在后台状态下运行一段代码，可用于从服务器获取内容更新。功能使用场景：（多媒体）聊天，Email更新，基于通知的订阅内容同步等功能，可以提升用户的体验效果。

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。

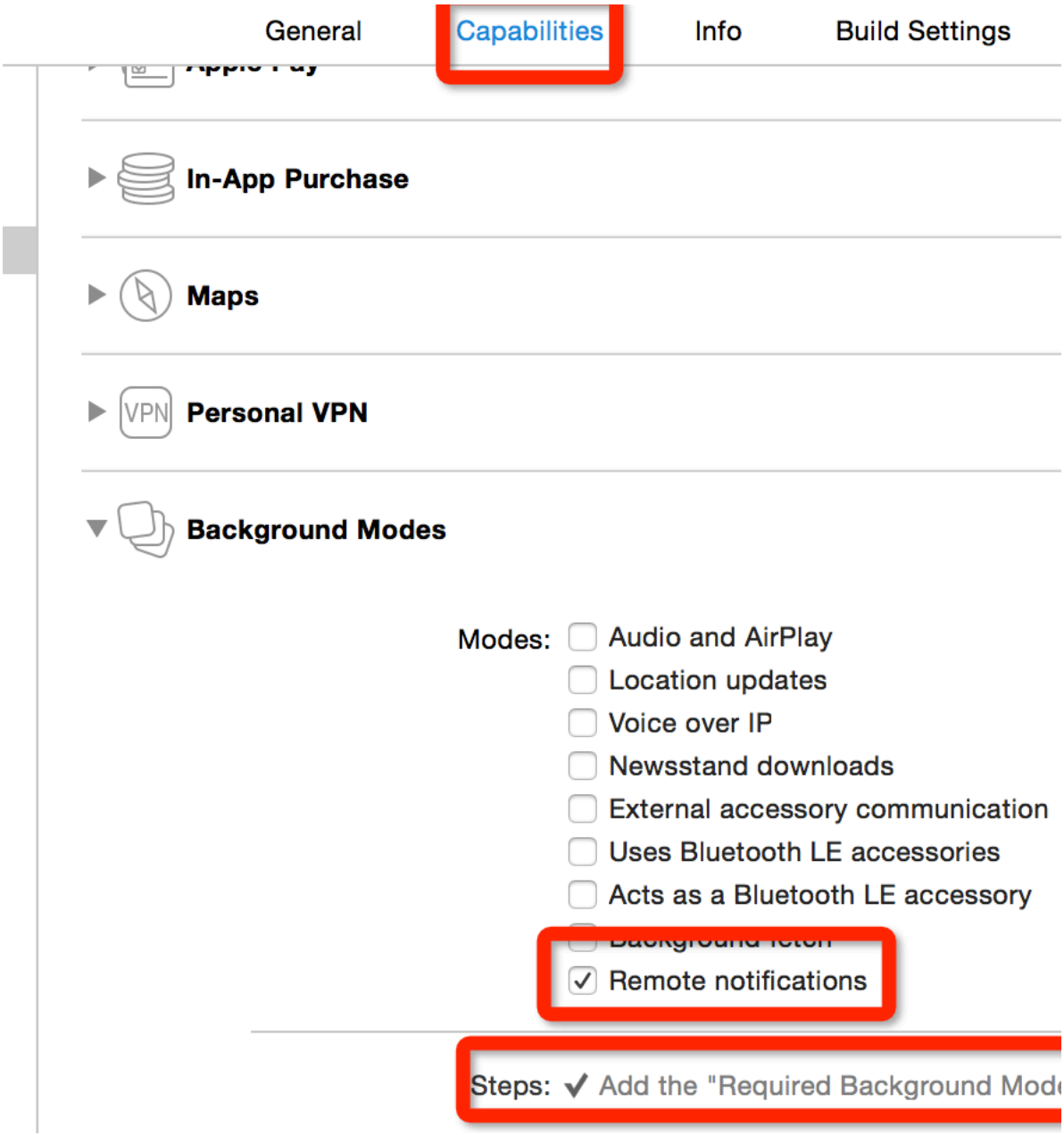




如果只携带content-available: 1 不携带任何badge，sound 和消息内容等参数，则可以不打扰用户的情况下进行内容更新等操作即为“Silent Remote Notifications”(静默推送)。



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
```

第七章 联系我们

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

百度hi官方技术讨论群：1405944

QQ群1：348807820（可加）

QQ群2：242190646（可加）

QQ群3：324533810（可加）

云推送微信公众账号：云推送（微信号：baidu-push）

微信扫一扫：