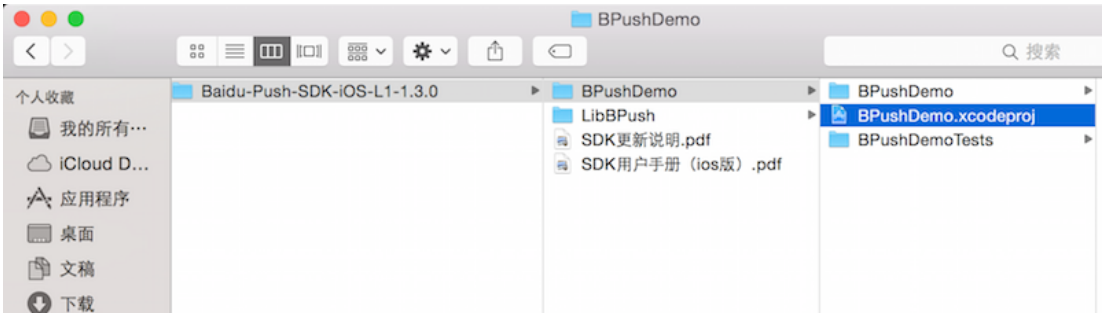
百度云推送SDK用户手册（iOS版）

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 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-L1-1.3.0.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 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

开发者需要使用百度账号登录 百度开发者中心 注册成为百度开发者并创建应用,方可获取应用 ID、对应的 API Key 及 Secret Key 等信息。具体信息,请参考 开发者服务管理的相关介绍。 其中,应用 ID(即:APP ID)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。

第五章 iOS SDK开发步骤

5.1 应用程序工程配置

5.1.1 添加 SDK 到应用程序工程

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。

5.1.2 在代码中实现推送相关 API

完整的使用推送服务，还需要在 App 工程的代码中实现与消息推送服务有关的 API，具体操作按照如下步骤即可：

在 AppDelegate 中的 `application: didFinishLaunchingWithOptions:` 中调用 API，初始化Push：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    ...

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

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

    #warning 上线 AppStore 时需要修改 pushMode
    // 在 App 启动时注册百度云推送服务，需要提供 Apikey
    [BPush registerChannel:launchOptions apiKey:@"9dCvF6OqSY6FeYclSOfVyr1F" pushMode:BPushModeDevelopment isDebug:NO];

    // 设置 BPush 的回调
    [BPush setDelegate:self];

    // App 是用户点击推送消息启动
    NSDictionary *userInfo = [launchOptions objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
    if (userInfo) {
        [BPush handleNotification:userInfo];
    }

    ...

    return YES;
}
```

为了支持 iOS8，还需要在 AppDelegate 中添加下面的方法：

```
- (void)application:(UIApplication *)application didRegisterUserNotificationSettings:(UIUserNotificationSettings *)notificationSettings
{

    [application registerForRemoteNotifications];
}
```

在 `application: didRegisterForRemoteNotificationsWithDeviceToken:` 中调用 API，注册device token，并且绑定 Push 服务：

```
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    [BPush registerDeviceToken:deviceToken];
    [BPush bindChannel];
}
```

如果上面的方法不被调用，可以实现下面的方法来查看原因：

```
// 当 DeviceToken 获取失败时，系统会回调此方法
- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
    NSLog(@"DeviceToken 获取失败，原因：%@",error);
}
```

在 `application: didReceiveRemoteNotification:` 中调用 API，处理接收到的Push消息：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    // App 收到推送通知
    [BPush handleNotification:userInfo];
}
```

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)register:(NSDictionary *)launchOptions apiKey:(NSString *)apiKey pushMode:(BPushMode)mode isDebug:(BOOL)isdebug
设置 Push delegate	+ (void)setDelegate:(id)delegate
设置 Access Token	+ (void)setAccessToken:(NSString *)token
注册 Device Token	+ (void)registerDeviceToken:(NSData *)deviceToken
绑定	+ (void)bindChannel
解除绑定	+ (void)unbindChannel
处理推送消息	+ (void)handleNotification:(NSDictionary *)userInfo
设置 tag	+ (void)setTags:(NSArray *)tags
删除 tag	+ (void)delTags:(NSArray *)tags
列举 tag	+ (void)listTags
获取 appId	+ (NSString *) getAppId
获取 channelId	+ (NSString *) getChannelId
获取 userId	+ (NSString *) getUserId

注意：API 调用的返回结果都是用delegate回调，被设置为delegate的类必须实现以下方法

```
- (void)onMethod:(NSString*)method response:(NSDictionary*)data;
```

5.2.2 相关常量的定义：

1. 向百度 Push 请求后，返回的结果字典的键。

```
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;
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，即代理方法中 onMethod 段的值，用于判断当前回调的是哪个方法。

```
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

Push代理设置

```
@protocol BPushDelegate <NSObject>
```

函数原型：

```
- (void)onMethod:(NSString*)method response:(NSDictionary*)data;
```

功能描述：

开发者向 Push 服务发起bind、setTags、delTags等请求时，请求结果均在此代理方法中返回。

参数说明：

- method：表明是哪个方法的返回。可参照5.2.2相关常量的定义。
- data：对应方法返回的结果字典，NSDictionary类型，可参照5.2.2相关常量的定义。

返回结果：

（无）

注册 Push

功能描述：

注册百度 Push 服务，需要在应用程序委托 AppDelegate 中 application:didFinishLaunchingWithOptions: 方法中调用。

函数原型：

```
+ (void)register:(NSDictionary *)launchOptions apiKey:(NSString *)apiKey pushMode:(BPushMode)mode i
sDebug:(BOOL)isdebug;
```

参数说明：

- launchOptions：App 启动时系统提供的参数。
- apiKey：需要通过在云推送官网创建应用，然后根据应用的 apikey 值注册百度云推送。
- mode：当前推送的环境，分为开发环境和生产环境两种。
- isdebug：是否是debug模式，debug 模式下会在控制台打印一些调试语句。

返回结果：

（无）

设置Push delegate

函数原型：

```
+ (void)setDelegate:(id)delegate;
```

功能描述：

设置BPushDelegate，用于方法调用后的结果返回。未设置delegate或者未实现onMethod:response方法时，无法接收返回值。

参数说明：

- delegate：为需要实现 BPushDelegate 的实例，实例需要实现 onMethod:response 回调方法。

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）

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）

设置 Bduss

函数原型：

```
+ (void)setBduss:(NSString *)bduss forApp:(NSString *)appid;
```

功能描述： 设置应用程序的 budss，做登陆验证。在使用百度账号登陆体系时需要。

参数说明：

- bduss：bduss的主要功能是当用户登陆时，生成设置在客户端浏览器的bduss；当用户访问baidu网页时，通过session校验bduss的合法性。
- appid：在云推送官网上创建的应用的 Appid。

返回结果：

（无）

绑定

函数原型：

```
+ (void)bindChannel;
```

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BPushDelegate的onMethod:response:回调返回。

参数说明：

（无）

返回结果：

（无）

解除绑定

函数原型：

```
+ (void)unbindChannel;
```

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BPushDelegate的onMethod:response:回调返回。

参数说明：

（无）

返回结果：

（无）

设置用户标签

函数原型：

```
+ (void)setTags:(NSArray *)tags;
```

功能描述：

根据需要，为使用当前设备用户设置标签，设置成功后可在控制台按照标签推送，旨在于为开发者提供个性化的推送。

参数说明：

- tags：为用户设置的标签值数组，为字符串类型。

返回结果：

（无）

删除用户标签

函数原型：

```
+ (void)delTags:(NSArray *)tags;
```

功能描述：

删除当前用户的某些标签。

参数说明：

- tags：需要删除的标签值数组，为字符串类型。

返回结果：

（无）

列出用户标签

函数原型：

```
+ (void)listTags;
```

功能描述：

列出当前用户的所有标签。

参数说明：

（无）

返回结果：

（无）

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）

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：

返回结果：

字符串类型的appid、channelid或userid

第六章 SDK 版本更新说明

Push iOS SDK 1.3 版本相比之前做了一些改变，开发者需要通过下面的引导说明来完成升级。

在 `1.2` 版本中，App 注册 Push 服务调用的 API 为：

```
+ (void)setupChannel:(NSDictionary *)launchOptions;
```

并且需要在应用程序 Bundle 中新建一个名为 `BPushConfig.plist` 的文件。

但是在 `1.3` 版本中， 这条 API 做了修改，不需要再创建 `plist` 文件，而是改用在参数中直接传入的方式，如下：

```
+ (void)registerChannel:(NSDictionary *)launchOptions apiKey:(NSString *)apikey pushMode:(BPushMode
)mode isDebug:(BOOL)isdebug;
```

修改完成后，删除掉原有的 `plist` 配置文件以及 openSource 下的第三方库，使用最新的 `BPush.a` 静态库以及 `BPush.h` 头文件，即可完成更新。

详细请参考 SDK 包中 `版本更新` 说明文档。

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：