



Push 服务

SDK 用户手册

(iOS 版)

发布日期： 2013年10月23日

百度开发者中心

(版权所有，翻版必究)

目录

第 1 章	简介.....	3
第 2 章	阅读对象.....	3
第 3 章	SDK 功能说明.....	3
3.1	框架设计.....	3
3.2	主要功能.....	4
第 4 章	开发前准备	4
4.1	运行环境.....	4
4.2	参数申请及权限开通.....	4
4.3	账户支持.....	5
第 5 章	使用 SDK 开发应用	5
5.1	添加 SDK 到 APP 工程.....	5
5.2	调用 API.....	6
第 6 章	API 说明	7
6.1	相关常量定义.....	7
6.2	API.....	9
第 7 章	联系我们.....	12
第 8 章	缩略语.....	12

第1章 简介

百度 Push 服务 iOS SDK 是百度官方推出的 Push 服务的 iOS 平台开发 SDK，提供给 iOS 开发者简单的接口，轻松集成百度 Push 推送服务。

Push iOS SDK 的完整下载包为 Baidu-Push-SDK-iOS-L1-VERSION.zip，下载解压后的目录结构如下所示：

- demo:
示例工程，帮助用户快速了解如何使用 SDK；
- lib:
存放 libPushSDK.a，SDK 以静态库方式提供；
以及头文件 BPush.h；
opensource 目录下存放 sdk 中引用到的开源库源码，如果您的工程没有其中的库源码，请拷贝到您的工程中一同构建。
- SDK 用户手册
- SDK 版本说明书

第2章 阅读对象

本文档面向所有使用该 SDK 的 iOS 开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户。

第3章 SDK 功能说明

3.1 框架设计

Push iOS SDK 是开发者与 Push 服务器之间的桥梁；可以使用户越过复杂的 Push HTTP/HTTPS API，直接和 Push 服务器进行交互来使用 Push 服务。（框架设计如图 1 所示）

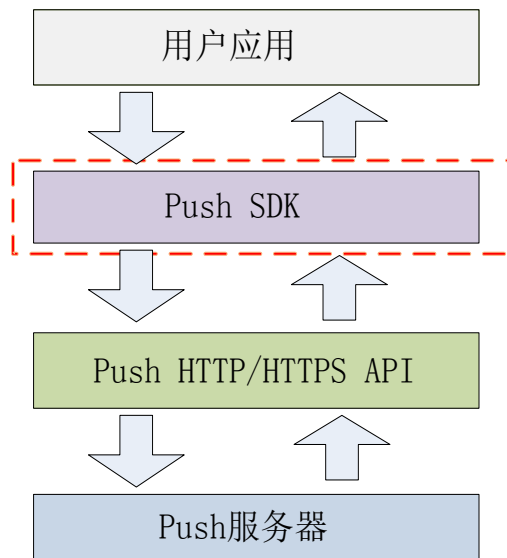


图 1 Push SDK 框架图

3.2 主要功能

本 SDK 主要提供以下功能的接口：

1. Push 服务
 - Push 服务初始化及绑定
 - Push 服务解绑定
2. Tag 管理

您可以创建或者删除标签

 - 创建 Tag
 - 删除 Tag
3. 通知推送
4. 推送效果反馈

第4章 开发前准备

4.1 运行环境

可运行于 iOS 5.0 及以上版本，5.0 以下版本暂不支持，不保证能正常工作。

4.2 参数申请及权限开通

4.2.1 获取应用 ID 及 API Key

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[百度开发者中心](#)上的“[创建应用](#)”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 账户支持

4.3.1 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 `userid` 作为唯一标识，使用 `AccessToken` 作为验证凭证。

4.3.2 无账户登录体系

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 `apiKey` 向 `Server` 请求用户标识 `userid`，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，此方式方便灵活，但需要开发者自己设计账户体系和登录界面。

2. 开发者 Access Token 登陆

`Access Token` 有两种方式可以获取：第一种，普通用户百度账户登陆获取，这在应用使用百度账号作为账户体系时使用，即 4.3.1 节所示；第二种，开发者百度账户登录获取。第一种可以换取百度账户的唯一的用户 `user id`；第二种我们的注册 `server` 会根据不同终端的 `device id` 分配不同的 `user id`。用第二种方式可以实现不依赖于百度账户的第三方登陆体系，实现该登陆方式，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5章 使用 SDK 开发应用

5.1 添加 SDK 到 APP 工程

1. 将 `libBPush.a` 和 `BPush.h` 添加到 Xcode 工程目录

2. 工程必须引用的库：

`Foundation.framework`
`CoreTelephony.framework`
`SystemConfiguration.framework`
`libz.dylib`

3. 添加必要的开源库源文件到 Xcode 工程

`libBPush.a` 引用了若干开源库。目前有：`JSONKit`、`Base64`、`GzipCompressor`、`OpenUDID`、`Reachability`。

如果您的工程已经使用了该库，可以省略这一步。

4. 创建并配置 `BPushConfig.plist` 文件

在工程中创建一个新的 `Property List` 文件，并命名为 `BPushConfig.plist`，添加以下键值：

```
"API_KEY" = "pDUCHGTbD346jt2klpHRjHp7"
"PRODUCTION_MODE" = NO
"DEBUG" = NO
"BPUSH_CHANNEL" = "91"
```

API_KEY: 必选。百度开发者中心为每个 app 自动分配的 api key，在开发者中心 app 基本信息中可以查看。

PRODUCTION_MODE: 必选。应用发布模式。开发证书签名时，值设为"NO"；发布证书签名时，值设为"YES"。请在调试和发布应用时，修改正确设置这个值，以免出现推送通知无法到达。

DEBUG: 可选。Push SDK 调试模式开关，值为 YES 时，将打开 SDK 日志。

BPUSH_CHANNEL: 可选。渠道号，云推送将会进行统计，在控制台可以看到统计结果。

5.2 调用 API

1. 在 application: didFinishLaunchingWithOptions: 中调用 API，初始化 Push:

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[UIScreen
mainScreen] bounds]];
    self.mainViewController = [[ViewController alloc]
initWithNibName:@"MainWindow" bundle:nil];
    self.window.rootViewController = self.mainViewController;
    [window makeKeyAndVisible];
    // 必须
    [BPush setupChannel:launchOptions];
    // 必须。参数对象必须实现 (void)onMethod:(NSString*)method
response:(NSDictionary*)data 方法，本示例中为 self
    [BPush setDelegate:self];

    [application registerForRemoteNotificationTypes:
        UIRemoteNotificationTypeAlert
        | UIRemoteNotificationTypeBadge
        | UIRemoteNotificationTypeSound];
    return YES;
}
```

2. 在 application: didRegisterForRemoteNotificationsWithDeviceToken: 中调用 API，注册 device token:

```

- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    // 必须
    [BPush registerDeviceToken:deviceToken];
    // 必须。可以在其它时机调用，只有在该方法返回（通过 onMethod:response:
    回调）绑定成功时，app 才能接收到 Push 消息。一个 app 绑定成功至少一次即可（如
    果 access token 变更请重新绑定）。
    [BPush bindChannel];
}

```

3. 实现 BPushDelegate 协议，实现函数 onMethod:response::

```

// 必须，如果正确调用了 setDelegate，在 bindChannel 之后，结果在这个回调中返回。
// 若绑定失败，请进行重新绑定，确保至少绑定成功一次
- (void) onMethod:(NSString*)method response:(NSDictionary*)data {
    if ([BpushRequestMethod_Bind isEqualToString:method]) {
        NSDictionary* res = [[NSDictionary alloc] initWithDictionary:data];
        NSString *appid = [res valueForKey:BPushRequestAppIdKey];
        NSString *userid = [res valueForKey:BPushRequestUserIdKey];
        NSString *channelid = [res valueForKey:BPushRequestChannelIdKey];
        int returnCode = [[res valueForKey:BPushRequestErrorCodeKey] intValue];
        NSString *requestid = [res valueForKey:BPushRequestRequestIdKey];
    }
}

```

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

```

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo {
    // 可选
    [BPush handleNotification:userInfo];
}

```

第6章 API 说明

6.1 相关常量定义

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 BPushRequesAppldKey;`

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

用枚举 `BpushErrorCode` 来定义百度 Push 返回的错误码，如下：

```
typedef enum BPushErrorCode
{
    BpushErrorCode_Success = 0,
    BpushErrorCode_MethodTooOften = 22, // 方法调用太频繁，如循环调用 bind
    BpushErrorCode_NetworkInvalible = 10002, // 网络连接错误
    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
} TBpushErrorCode;
```

3. 方法名，即 `onMethod:response:` 方法的第一个参数取值范围

`NSString *const BpushRequestMethod_Bind;`

`bind` 方法。

`NSString *const BpushRequestMethod_Unbind;`

`unbind` 方法。

`NSString *const BpushRequestMethod_SetTag;`

`setTags` 方法。

`NSString *const BpushRequestMethod_DelTag;`

`delTags` 方法。

6.2 API

1. API 主要包装在 Bpush 接口里，目前支持以下接口：

功能	API 函数原型
初始化 Push	+ (void)setupChannel:(NSDictionary *)launchOptions
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
解绑定	+ (void)unbindChannel
处理 Push 消息	+ (void)handleRemoteNotification:(NSDictionary *)userInfo
设置 tag	+ (void)setTags:(NSArray *)tags
删除 tag	+ (void)delTags:(NSArray *)tags
获取 appid	+ (NSString *)getAppId
获取 channelId	+ (NSString *)getChannelId
获取 userid	+ (NSString *)getUserId

2. BPushDelegate protocol 用来返回绑定等调用的结果，必须实现方法：

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

6.2.1 初始化 Push

- 函数原型

+ (void)setupChannel:(NSDictionary *)launchOptions

- 功能

初始化百度 Push 服务。请在 application: didFinishLaunchingWithOptions: launchOptions 中调用。

- 参数

launchOptions: 就是 application: didFinishLaunchingWithOptions: launchOptions 中传入的 launchOptions。

- 返回结果

无

6.2.2 设置 Push delegate

- 函数原型

+ (void)setDelegate:(id) delegate

- 功能

设置 BPushDelegate。如果不设置该 delegate，或者未实现 onMethod:response:方法，bind 等方法仍可正常调用，不过将无法得到其返回值。

- 参数

delegate: 实现 BPushDelegate 的实例，在 AppDelegate 中实现 onMethod:response:方法，再将 delegate 设置成 self 即可方便实现。

- 返回结果

无

6.2.3 设置 Access Token

- 函数原型

```
+ (void)setAccessToken:(NSString *)accessToken
```

- 功能

当 APP 用 access token 方式绑定时，用于设置 access token，无账号绑定时（api key 绑定）无需调用该接口。

如果 APP 账号体系用的不是百度账号，可以用开发者自己的百度账号获取 access token 给所有端使用，即 4.3.2 节无账号登陆体系的第二种。必须注意的是：access token 有过期时间，请及时为每个端更新 access token，并重新执行绑定操作。

- 参数

accessToken: 通过百度账号认证方式获取的 Access Token。

- 返回结果

无

6.2.4 注册 Device Token

- 函数原型

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

- 功能

向百度 Push 服务注册端的 device token，百度 Push 服务通过 APNs 向每个端推送消息时，必须用到。

- 参数

deviceToken: 直接使用 application: application

didRegisterForRemoteNotificationsWithDeviceToken: deviceToken 传入的 deviceToken 参数即可。

- 返回结果

无

6.2.5 绑定

- 函数原型

```
+ (void)bindChannel
```

- 功能

绑定 Push 服务通道。在这些条件满足时才能绑定成功：设置好 access token 或者 api key；注册了 device token。

绑定请求结果通过 BpushDelegate 的 onMethod:response:回调返回。

- 参数

无

- 返回结果

无

6.2.6 解绑定

- 函数原型

```
+ (void)unbindChannel
```

- 功能

解绑定 Push 服务通道。成功解绑定后，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，可以恢复推送功能。

解绑定请求结果通过 BpushDelegate 的 onMethod:response:回调返回。

- 参数

无

- 返回结果

无

6.2.7 处理 Push 消息

- 函数原型

```
+ (void)handleRemoteNotification:(NSDictionary *)userInfo
```

- 功能

处理 Push 消息。用于对每条推送消息的反馈和统计，如果您想对消息的推送获取及时的反馈，请正确调用该方法。在 application: didReceiveRemoteNotification: 中调用。

- 参数

userInfo: 直接使用 application: didReceiveRemoteNotification:传入的参数即可。

- 返回结果

无

6.2.8 BPushDelegate

- 原型

```
@protocol BPushDelegate <NSObject>
- (void)onMethod:(NSString*)method response:(NSDictionary*)data;
```

@end

- 功能
用于向 Push 服务发起 bind、setTags、delTags 等服务请求时的结果返回。
- 参数
method: 表明是哪个方法的返回, 可能值: bind、set_tag、del_tag。
- 返回结果
无

6.2.9 获取应用绑定信息

- 函数原型

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

- 功能
在应用成功绑定后, 可以通过调用这三个接口来获取 appid、channelid 和 userid, 在绑定之前或者被解绑定后, 将返回空。
- 参数
无
- 返回结果
appid、channelid、userid

第7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具体问题, 请通过以下方式联系我们:
邮箱: dev_support@baidu.com
百度工程师会在第一时间回复您。

第8章 缩略语

缩略语	英文全称	说明
SDK	Software Development Kit	软件开发工具包。