

**信鸽iOS SDK**

**开发者指南**

腾讯（深圳）科技有限公司

（版权所有）

目录

1. 简介 3

2. 运行环境 3

3. 功能 3

3.1. 获取AppID和AppKey 3

3.2. 工程配置 4

4. API接口 4

4.1. 注册设备信息 4

4.2. 初始化Push信息 5

4.3. 设置标签 5

4.4. 删除标签 6

4.5. 推送被打开效果统计 6

4.6. 注销设备 7

4.7. 设置别名(帐号) 7

4.8. 本地推送 7

4.9. 本地推送，App在前端运行时响应 8

4.10. 删除本地推送，方法一 8

4.11. 删除本地推送，方法二 9

4.12. 清空本地推送 9

4.13. 回调函数的说明 9

# 简介

信鸽iOS SDK是一个能够提供Push服务的开发平台，提供给开发者简便、易用的API接口，方便快速接入。

# 运行环境

iOS5.0 及以上版本。在程序启动时注册通知，参考以下代码，

- (void) registerNofitication {  
[[UIApplication sharedApplication] registerForRemoteNotificationTypes:(UIRemoteNotificationTypeAlert | UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound)];  
}  
  
- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions  
{  
    [self registerNofitication];   
    return YES;  
}

# 功能

本SDK主要提供以下功能：

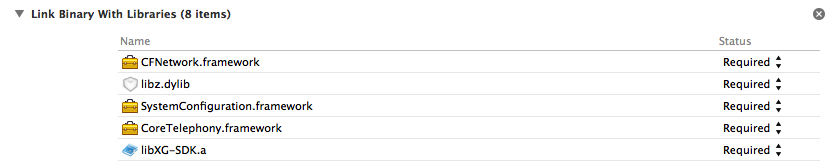
1. **注册设备**
   1. 初始化设备信息(返回deviceToken)
   2. 注销设备
2. **初始化Push信息**
   1. Push服务的初始化
3. **标签（Tag）**
   1. 设置标签
   2. 删除标签
4. **效果统计**
   1. 推送点击效果(app在前台运行时)
   2. 推送点击效果(app不在前台运行,点击通知激活时)

## 获取AppID和AppKey

前往前台<http://xg.qq.com>注册并获取AppKey。

## 工程配置

1. 下载信鸽SDK压缩包到本地并解压；
2. 创建或打开Xcode iOS工程；
3. 将XGPush.h和libXG-SDK.a添加到Xcode工程；
4. 添加对以下libraries的引用。包括CFNetwork.framework , SystemConfiguration.framework , CoreTelephony.framework , libz.dylib , libXG-SDK.a，



如果是信鸽2.0以后的版本，还需要增加对libsqlite.dylib的引用

# API接口

|  |  |
| --- | --- |
| **类名** | **说明** |
| XGPush | Push服务，推送效果 |

注意: 在Console(控制台)里会看到”[xgpush]Disconnected.”为正常现象，在调用信鸽接口时候，如果服务器响应超时或者已经成功收到服务器的返回数据，手机会断开和服务器的连接。

## 注册设备信息

在application: didRegisterForRemoteNotificationsWithDeviceToken中调用。

**注意**:设备的deviceToken由苹果下发，可能会产生变化。同一设备在开发环境和生产环境是不相同的。

发起注册条件 1:未注册 2:account变更 3:距离上次注册成功超过15小时。达到以上条件才能向后台进行网络请求，以及触发回调函数。

* **函数原型**
* +(NSString\*)registerDevice:(NSData \*)deviceToken;
* **参数**

deviceToken：

函数didRegisterForRemoteNotificationsWithDeviceToken的参数。

* **返回值**

NSString：获取到的deviceToken字符串

* **示例**
* - (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)deviceToken {

//注册设备

NSString \* deviceTokenStr = [XGPush registerDevice: deviceToken];

//打印获取的deviceToken的字符串

NSLog(@"deviceTokenStr is %@",deviceTokenStr);

}

## 初始化Push信息

在application: didFinishLaunchingWithOptions中调用startWithAppkey接口，初始化。

* **函数原型**
* +(void)startApp:(uint32\_t)appId appKey:(NSString \*)appKey;
* **参数**

appId:

通过xg.qq.com注册app时得到的appID。

appkey：

通过xg.qq.com注册app时得到的key。

* **返回值**

无

* **示例**
* [XGPush startApp:101 appKey:@"akey"];

## 设置标签

开发者可以针对**不同的用户**设置标签。

* **函数原型**
* +(void)setTag:(NSString\*)tag;
* **参数**

tag：

字符串

* **返回值**

无

* **示例**
* NSString\* tag= @"age:10";

[XGPush setTag: tag];

## 删除标签

开发者可以删除设置过的标签。

* **函数原型**
* +(void)delTag:(NSString \*)tag;
* **参数**

tag：

字符串

* **返回值**

无

* **示例**
* NSDictionary\* tag= @"age:10";

[XGPush delTag: tag];

## 推送被打开效果统计

如果需要统计由信鸽推送的点击或者打开，需要开发者在didReceiveRemoteNotification(如果没有则按照下面的示例手动添加)中调用handleReceiveNotification , 然后在didFinishLaunchingWithOptions中调用handleLaunching。

* **函数原型**

+(void)handleReceiveNotification:(NSDictionary \*)userInfo;//app在前台运行时

+(void)handleLaunching:(NSDictionary \*)launchOptions;//app不在前台运行时，点击推送激活时

* **参数**

事件本身默认的参数名

* **返回值**

无

* **示例**

- (void)application:(UIApplication\*)application didReceiveRemoteNotification:(NSDictionary\*)userInfo  
{  
  [XGPush handleReceiveNotification:userInfo];  
}

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

[XGPush handleLaunching: launchOptions];

}

## 注销设备

注销设备,此台设备不接收推送。

* **函数原型**
* +(void)unRegisterDevice;
* **参数**

无

* **返回值**

无

* **示例**

 [XGPush unRegisterDevice];

## 设置别名(帐号)

上报用户的别名(帐号)，以便支持按别名(帐号)推送。

在初始化信鸽后，注册设备之前调用。

* **函数原型**
* +(void)setAccount:(NSString \*)account;
* **参数**

account:

用户的别名(帐号)

* **返回值**

无

* **示例**

 [XGPush setAccount:@"123456"];

## 本地推送

在指定时间进行本地推送。

* **函数原型**
* +(void)localNotification:(NSDate \*)fireDate alertBody:(NSString \*)alertBody badge:(int)badge alertAction:(NSString \*)alertAction userInfo:(NSDictionary \*)userInfo;
* **参数**

fireDate:

推送触发的时间

alertBody:

推送的内容

badge:

改变App的角标，如果为-1，则不改变。

alertAction:

推送的右侧按钮的文字，默认为“启动”。

userInfo:

自定义的一组key/value信息

* **返回值**

无

* **示例**

 [XGPush localNotification:fireDate alertBody:@"测试本地推送" badge:2 alertAction:@"确定" userInfo:userInfo];

## 本地推送，App在前端运行时响应

App在前台运行时，苹果的推送默认不弹窗。通过此接口可进行弹窗。

* **函数原型**
* +(void)localNotificationAtFrontEnd:(UILocalNotification \*)notification userInfoKey:(NSString \*)userInfoKey userInfoValue:(NSString \*)userInfoValue;
* **参数**

userInfoKey:

标识本地推送的Key

userInfoValue:

标识本地推送的Value

* **返回值**

无

* **示例**

 [XGPush localNotificationAtFrontEnd:notification userInfoKey:@"clockID" userInfoValue:@"myid"];

## 删除本地推送，方法一

iOS最多允许一个App存储64个本地推送。以下函数可以进行删除。

* **函数原型**
* +(void)delLocalNotification:(NSString \*)userInfoKey userInfoValue:(NSString \*)userInfoValue;
* **参数**

userInfoKey:

标识本地推送的Key

userInfoValue:

标识本地推送的Value

* **返回值**

无

* **示例**

 [XGPush delLocalNotification:@"clockID" userInfoValue:@"myid"];

## 删除本地推送，方法二

iOS最多允许一个App存储64个本地推送。以下函数可以进行删除。

* **函数原型**
* +(void)delLocalNotification:(UILocalNotification \*)myUILocalNotification;
* **参数**

myUILocalNotification:

本地推送对象

* **返回值**

无

* **示例**

 [XGPush delLocalNotification:notification];

## 清空本地推送

iOS最多允许一个App存储64个本地推送。以下函数可以进行清空。

* **函数原型**
* [XGPush clearLocalNotifications];
* **参数**

无

* **返回值**

无

* **示例**

 [XGPush clearLocalNotifications];

## 回调函数的说明

从2.1.0(含)版本之后，带有网络请求的接口都增加了回调函数，包括成功时的回调succussCallback和失败时的回调errorCallback。2.1.0之前的版本可以通过Console里的rspCode判断结果，为0代表成功。

* **函数原型**
* succussCallback:(void (^)(void)) succussCallBack
* errorCallback:(void (^)(void)) errorCallback
* **返回值**

无

* **示例**

  void (^succussBlock)(void) = ^(void){

//成功之后的处理

NSLog(@"[xgpush]register succussBlock ,deviceToken: %@",deviceTokenStr);

};

void (^errorBlock)(void) = ^(void){

//失败之后的处理

NSLog(@"[xgpush]register errorBlock");

};

//注册设备

[XGPush registerDevice:deviceToken succussCallback:succussBlock errorCallback:errorBlock];