瞬推iOS集成文档

版本说明

- 版本号：V1.0.8

```
1：SDK新增非透传消息的本地通知展现，可通过setOnLineUnThroughNotificationShow关闭该
```

- 版本号：V1.0.7

```
1:接口加密
```

- 版本号：V1.0.6

```
1:修复重复上报的问题
```

- 版本号：V1.0.5

文档更新内容

```
1：兼容日历、地域本地推送；
2:兼容不同版本的推送Api;
3:调整初始化SDK方法、添加lanuchoptions参数；
```

- 版本号：V1.0.4

文档更新内容

```
1：简化SDK的引用和api调用、统一由ShareInstallPushSDK提供；
2：修复[SHTCPSocketHeartbeat setTimer:]崩溃
```

- 版本号：V1.0.3

文档更新内容

```
1：添加云控接口
```

- 版本号：V1.0.2

文档更新内容

```
1: SDK优化前后台切换时线程释放引起的卡顿问题；
2：增添推送消息的透传、本地通知/apns、在线/离线、前后台等类型的区分字段
3：消息回调代理调整：ShareInstallPushDelegate
    新增apns前台消息回调方法：handleForegroundNotificationViewInfoWithAps
```

- 版本号：V1.0.1

文档更新内容

```
1：SDK初始化API调整、移除接入方实现APNS注册；
2：在线消息回调调整：当开启非透传消息本地通知展现时，回调数据只包含透传消息；
                    当未开启非透传消息本地通知展现时，回调数据包含透传+非透传消息；
3:增加APNS前台消息回调代理：当前台消息不在通知栏展现时，消息通过 handleForegroundNo
4:修复进入后台时收不到推送的问题；
```

- 版本号：V1.0.0

文档更新内容

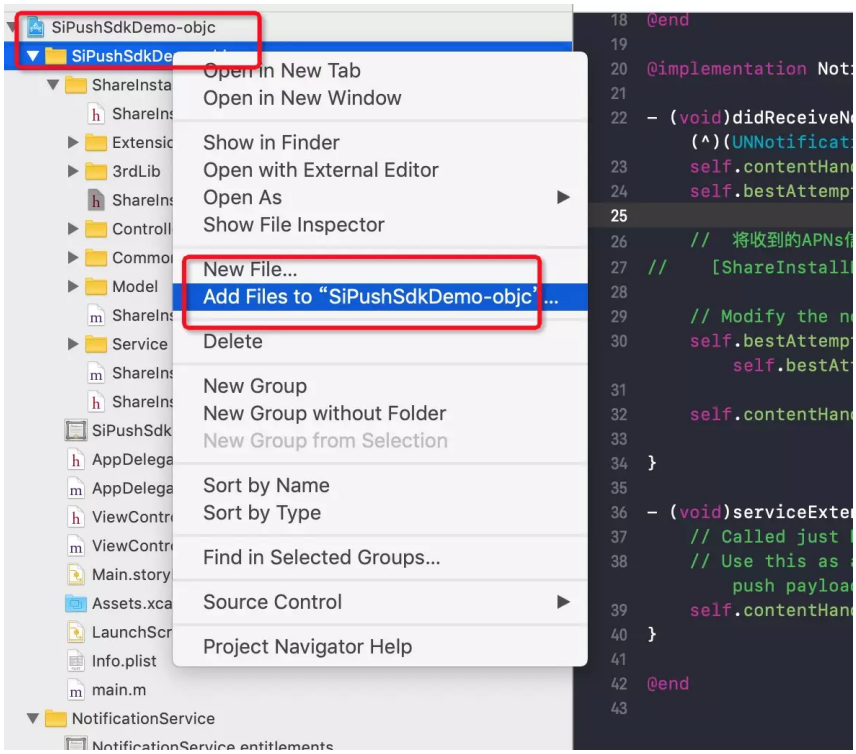
```
1：文档第一版
```

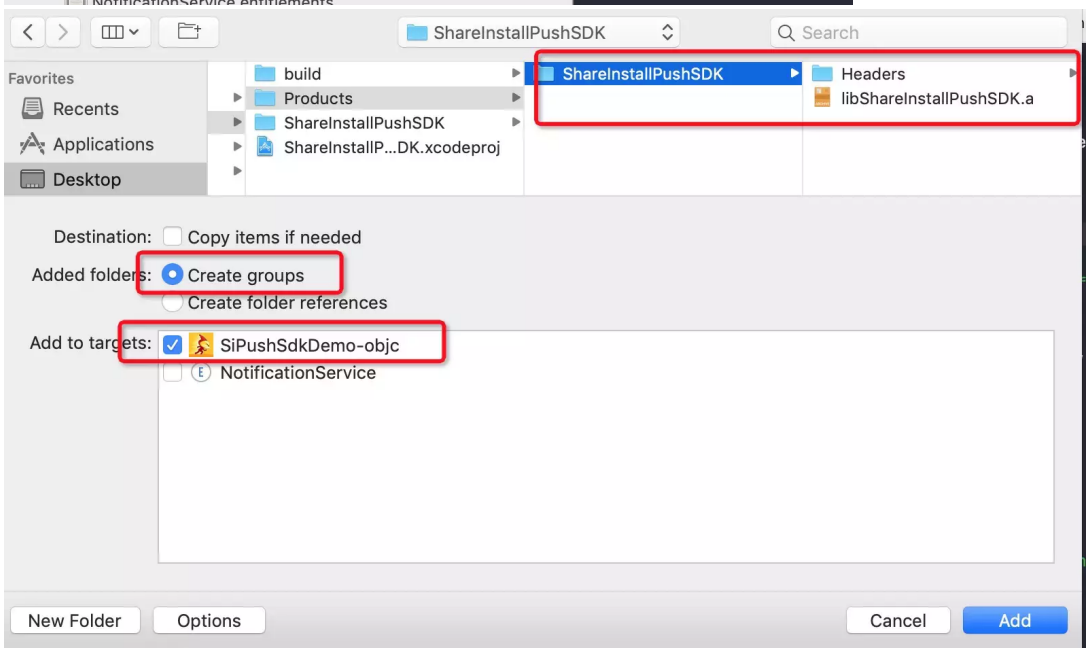
https://shimo.im/mindmaps/GCTk8XjxxcvrQh9h/ 《iOS瞬推SDK使用思维导图》
https://shimo.im/mindmaps/vKTGkrrCpjpWvYGy/ 《iOS推送消息响应思维导图》

一：SDK集成

- 导入瞬推SDK资源

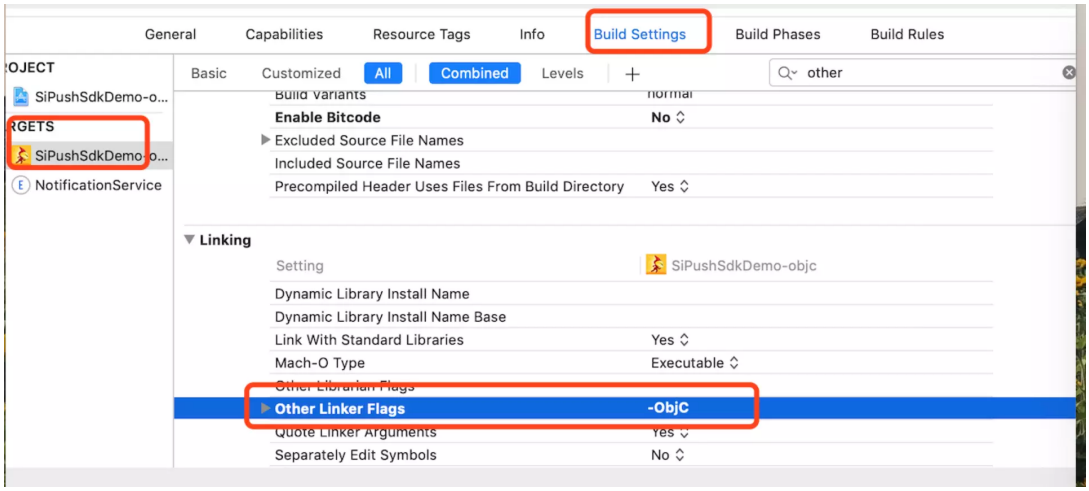
```
通过拖拽或addFiles方式 添加SDK到项目中 ，需勾选target ,选择Creates groups.   Copy
```



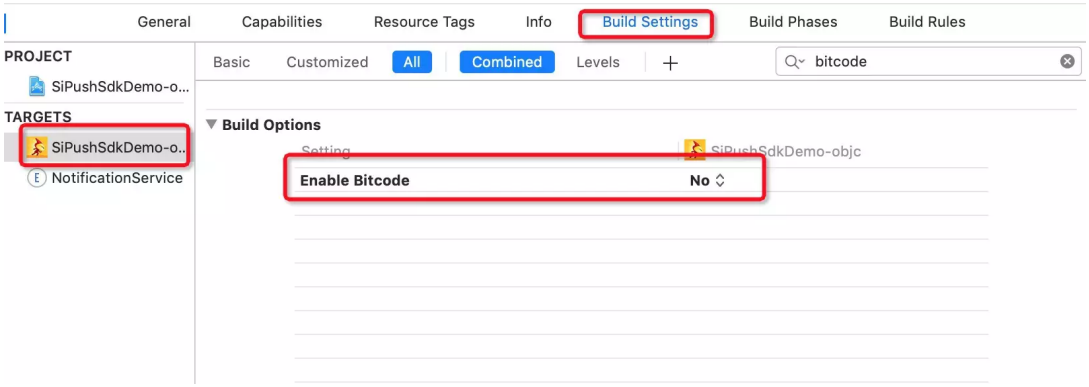


- 项目配置

1. 在项目的对应Target-->build setting中，设置Other Linker Flags :-Objc



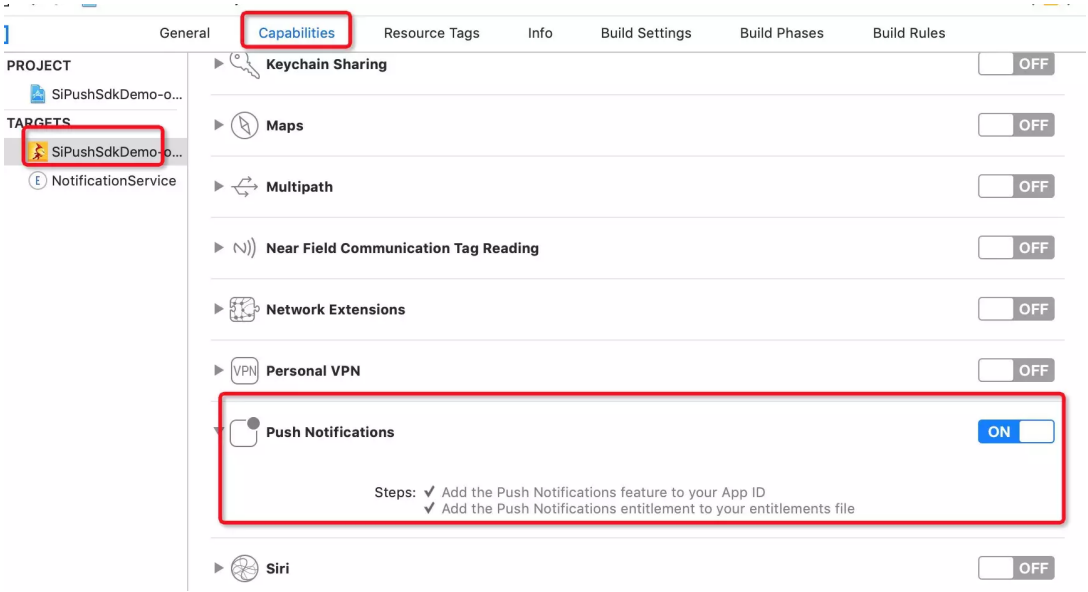
2. 在项目的对应Target-->build setting中，设置bitCode :NO.



二：推送配置

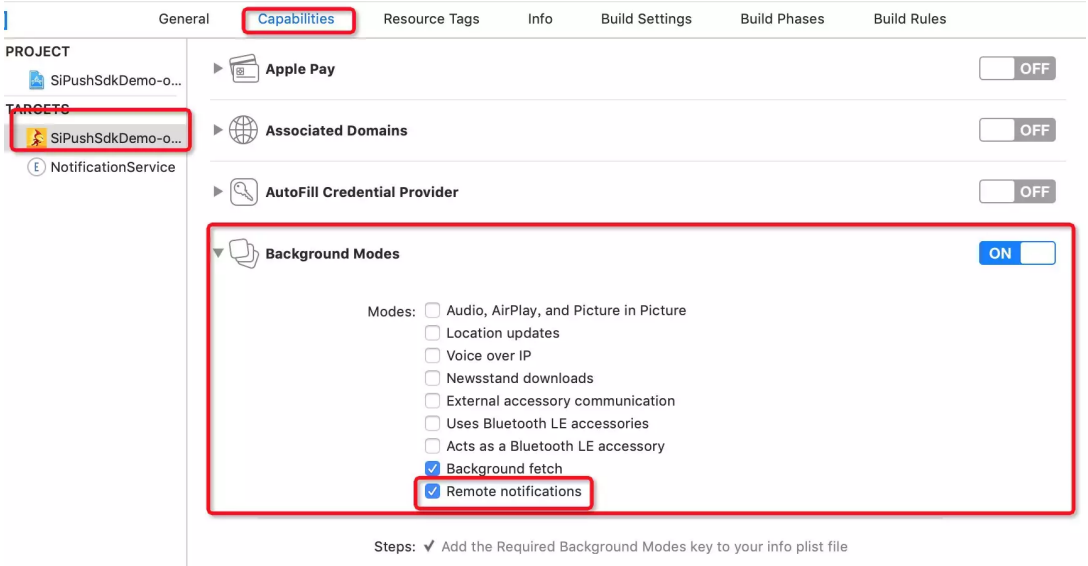
- 开启推送功能

在 Xcode 8.x 以上，必须开启Push Notification能力。找到应用Target设置中的Capabilities -> Push Notifications，确认开关已经设为ON状态。如果没有开启该开关，在 Xcode 8.x 上编译后的应用将获取不到DeviceToken：



- 后台运行权限

为了更好支持消息推送，提供更多的推送样式，提高消息到达率，需要配置后台运行权限



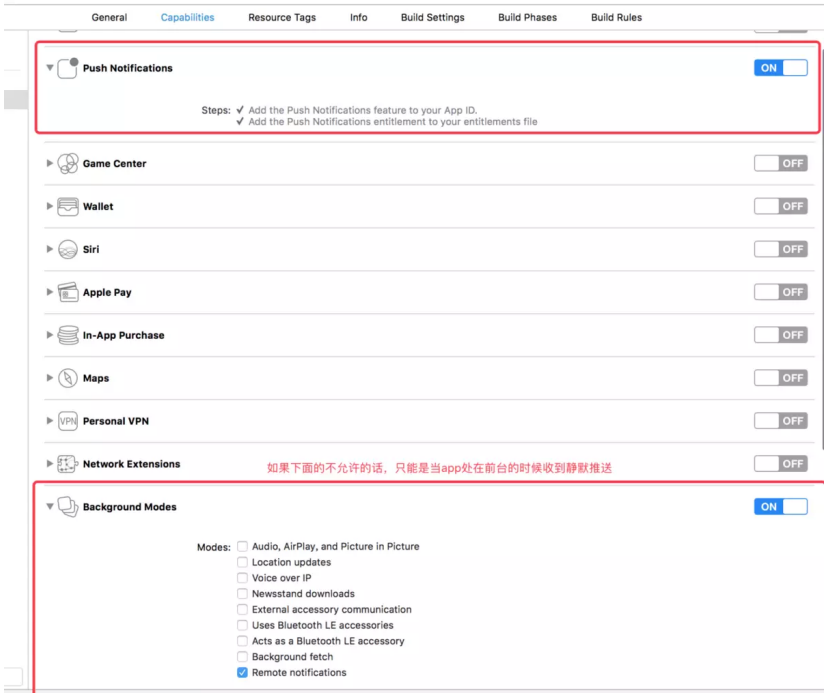
- 静默推送

静默推送：在用户察觉不到的情况下，没有声音，没有振动，没有弹框提示>进行远程推送，推送完毕根据推送过来的相关信息完成相关的操作

1. 推送格式

```
{
    "aps":{
        "aa":"123",
        "content-available":1//必须要有
    }
}
//必须不能携带 alert、badge、sound，不然就不是静默推送了。
```

2. 注意事项



3. iOS13适配

> iOS13以上静默消息服务端需配置apns-push-type。具体参考苹果Apns文档(https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/sending_notification_requests_to_apns)

三：SDK代码集成使用

- 引用SDK头文件、设置代理

先导入头文件ShareInstallPushSDK.h，并设置ShareInstallPushDelegate

```
#import "ShareInstallPushSDK.h"

@interface AppDelegate ()<ShareInstallPushDelegate>
@end

ShareInstallPushDelegate提供以下代理方法，用于传递在线离线消息、通知栏点击事件、SDK
/**
```

```
 获取TCP离线的数据 、SDK不做数据的任何处理
 包含两部分数据：1：透传消息，2：非透传消息
 @param json 数据
 */
- (void)offline:(NSArray <ShareInstallPushInfoModel *> *)json;

/**
 获取TCP在线的数据、SDK针对非透传消息默认发本地通知
 V1.0.0版本：包含两部分数据：1：透传消息，2：非透传消息
 V1.0.1版本：包含两部分数据：1：透传消息，2：非透传消息(当关闭非透传消息本地通知展现的
 @param json json
 */
- (void)online:(NSArray <ShareInstallPushInfoModel *> *)json;

/**
 点击通知栏(APNS)推送数据中的action事件处理
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环
 @param aps 推送数据
 */
- (void)handleActionWithAps:(ShareInstallPushInfoModel *)aps;

/**
 应用处于前台时，未做通知栏展现的APNS消息(iOS10以上不展现通知栏或iOS10以下的前台消息
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环
 @param aps 推送数据
 */
- (void)handleForegroundNotificationWithAps:(ShareInstallPushInfoModel *)aps

/**
 应用处于前台时，做通知栏展现的APNS消息(iOS10以上)
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环
 * 当点击通知栏消息时，该通知消息会通过handleActionWithAps 再次传递，并传递的消息类
 @param aps 推送数据
 */
- (void)handleForegroundNotificationViewInfoWithAps:(ShareInstallPushInfoMod

/**
 SDK初始化失败的代理
 @param error error
 */
- (void)shareInstallInitFailureWithError:(NSError *)error;


/**clientId回调
 @param clientId clientId
 */
- (void)registerClientIdBlock:(NSString *)clientId;
```

- 初始化SDK~~并注册APNS~~

在[AppDelegate didFinishLaunchingWithOptions]方法中调用瞬推SDK初始化方法，传入瞬推平台分配的 AppKey、AppSecret，并设置瞬推SDK数据统计上报需要的PublicInfo ，~~调用APNs注册方法，尝试获取APNs DeviceToken~~。示例代码如下

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptic
    //注册Apns
    //[self registerRemoteNotification];//新版SDK已不需要接入方注册
    //初始化SDK  设置PublicInfo，具体参数见SDK方法注释
    //[ShareInstallPushSDK setAppKey:kSiAppKey appSecret:kSiAppSecret delega
    [ShareInstallPushSDK setAppKey:kSiAppKey appSecret:kSiAppSecret delegate
];
    return YES;
}
```

~~设置~~

~~注册APNs获取DeviceToken的流程，根据项目设置的不同以及手机系统版本的不同，注册代码会有所区别，可以参考如下方式进行适配~~：新版SDK不需要接入方注册、若接入方也注册可能会出现推送提醒多次的问题,建议先unregisterForRemoteNotifications

```
- (void)registerRemoteNotification {
   /*
     警告：Xcode8 需要手动开启"TARGETS -> Capabilities -> Push Notifications"
     */
    /*
     警告：该方法需要开发者自定义，以下代码根据 APP 支持的 iOS 系统不同，代码可以对应
     以下为演示代码，注意根据实际需要修改，注意测试支持的 iOS 系统都能获取到 DeviceT
     */
    if ([[UIDevice currentDevice].systemVersion floatValue] >= 10.0) {
#if __IPHONE_OS_VERSION_MAX_ALLOWED >= __IPHONE_10_0 // Xcode 8编译会调用
        UNUserNotificationCenter *center = [UNUserNotificationCenter current
        center.delegate = self;
        [center requestAuthorizationWithOptions:(UNAuthorizationOptionBadge
            if (!error) {
                NSLog(@"request authorization succeeded!");
            }
        }];

        [[UIApplication sharedApplication] registerForRemoteNotifications];
#else // Xcode 7编译会调用
        UIUserNotificationType types = (UIUserNotificationTypeAlert | UIUser
        UIUserNotificationSettings *settings = [UIUserNotificationSettings s
        [[UIApplication sharedApplication] registerForRemoteNotifications];
        [[UIApplication sharedApplication] registerUserNotificationSettings:
#endif
    } else if ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0)
        UIUserNotificationType types = (UIUserNotificationTypeAlert | UIUser
        UIUserNotificationSettings *settings = [UIUserNotificationSettings s
        [[UIApplication sharedApplication] registerForRemoteNotifications];
        [[UIApplication sharedApplication] registerUserNotificationSettings:
```

```
    } else {
        UIRemoteNotificationType apn_type = (UIRemoteNotificationType)(UIRem
                                                                       UIRem
                                                                       UIRem
        [[UIApplication sharedApplication] registerForRemoteNotificationType
    }
}
```

四：推送消息回调处理

1. 消息推送方式

- 离线苹果原生Apns服务
- 在线TCP推送服务
- 当App处于前台或进入后台进程未杀死的情况下，默认启动TCP服务，当TCP服务启动失败，推送消息 以APNS服务进行推送；

2. 消息回调方式

- 苹果Apns消息回调，接入方无需实现通知代理方法，可通过ShareInstallPushDelegate的代理方法统一接收通知消息。若已实现通知代理切莫

```

/**
 点击通知栏(APNS)推送数据中的action事件处理
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环
 @param aps 推送数据
 */
- (void)handleActionWithAps:(ShareInstallPushInfoModel *)aps;
注意：
若未实现该代理方法，通知消息会按照默认的处理方式进行消息处理
at = 0 启动应用
at = 1 打开内部网页
at = 2 打开外部网页
at = 3 打开内部页面
at = 4 通过intent打开页面、iOS不支持
at = 5 打开其他应用
at = 6 打开下载连接

/**
 应用处于前台时，未做通知栏展现的APNS消息(iOS10以上不展现通知栏或iOS10以下的前台消息
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环

 @param aps 推送数据
 */
- (void)handleForegroundNotificationWithAps:(ShareInstallPushInfoModel *)aps

/**
 应用处于前台时，做通知栏展现的APNS消息(iOS10以上)
 * 不能在该回调中再发本地通知、会出现通知-->代理-->通知-->代理的死循环
 * 当点击通知栏消息时，该通知消息会通过handleActionWithAps 再次传递，并传递的消息类
```

```
 @param aps 推送数据
 */
- (void)handleForegroundNotificationViewInfoWithAps:(ShareInstallPushInfoMod
```

- 在线TCP消息回调

```
/**
 获取TCP离线的数据 、SDK不做数据的任何处理
 包含两部分数据：1：透传消息，2：非透传消息

 @param json 数据
 */
- (void)offline:(NSArray <ShareInstallPushInfoModel *> *)json;

/**
 获取TCP在线的数据、SDK针对非透传消息默认发本地通知
 V1.0.0版本：包含两部分数据：1：透传消息，2：非透传消息
 V1.0.1版本：包含两部分数据：1：透传消息，2：非透传消息(当关闭非透传消息本地通知展现的

 @param json json
 */
- (void)online:(NSArray <ShareInstallPushInfoModel *> *)json;
```

- SDK错误Error

```
SDK初始化错误、参数错误、接口异常等错误Error 通过ShareInstallPushDelegate代理回调
/**
 SDK初始化失败的代理
 @param error error
 */
- (void)shareInstallInitFailureWithError:(NSError *)error;
```

- ClientId获取

```
ClientId通过ShareInstallPushDelegate代理回调传递,当ClientId注册成功后会触发该代理
/**clientId回调
 @param clientId clientId
 */
- (void)registerClientIdBlock:(NSString *)clientId;
```

五：API调用

- DEBUG环境切换

```
Debug环境切换通过ShareInstallPushSDK的setDebug方法设置
/**
 切换环境
 @param debug YES:测试环境  ，NO：生产环境  默认NO
 */
+ (void)setDebug:(BOOL)debug;
//示例
```

```
[ShareInstallPushSDK setDebug:YES];
```

- 在线消息---透传消息的默认本地通知栏展现开关

```
方法设置
/**
 是否处理在线--透传消息的默认通知栏通知栏展现
 @param show 是否展现 默认YES
 */
+ (void)setOnLintNotificationShow:(BOOL)show;
//示例
[ShareInstallPushSDK setOnLintNotificationShow:YES];
```

- Log输出开关

```
/**
 是否关闭Log输出、默认NO

 @param off 开关
 */
+ (void)setLogOff:(BOOL)off;
```

- 注册DeviceToken

```
SDK在获取到deviceToken后会主动向推送服务端注册token,开发者也可以通过ShareInstallPu
/**
 注册DeviceToken 可用于刷新token,默认不用实现

 @param deviceToken deviceToken
 @return deviceToken是否有效
 */
+ (BOOL)registerDeviceToken:(NSString *)deviceToken;
```

- 重置ClientId

```
SDK提供刷新ClientId的方法，可通过ShareInstallPushSDK的resetClientId实现
/**重置缓存的clientId
 * 重置缓存的clientId
 * 下次启动生效或者在setAppkey之前调用生效
 */
+ (void)resetClientId;
```

- 设置别名

```
SDK提供别名推送，可通过ShareInstallPushApiManager的setAlias设置别名
/**
 设置用户别名
 @param alias 用户别名 最长16
 */
+ (void)setAlias:(NSString *)alias complate:(void(^)(id obj,NSError *error))
```

- 卸载别名

```
SDK提供别名卸载方法，可通过ShareInstallPushApiManager的unsetAlias卸载别名
```

```
/**
 卸载用户别名
 */
+ (void)unsetAliasWithComplate:(void(^)(id obj,NSError *error))complate;
```

- **获取别名**

```
SDK可通过ShareInstallPushApiManager的getAliasWithComplate获取别名
/**
 获取用户别名
 @param complate 回调函数
 */
+ (void)getAliasWithComplate:(void(^)(id obj,NSError *error))complate;
```

- **设置标签**

```
SDK提供标签推送方法，可通过ShareInstallPushApiManager的setTags设置标签
/**
 设置用户标签
 @param tags 标签集合 ,英文逗号隔开 如 上海,男,最多30个
 */
+ (void)setTags:(NSArray *)tags complate:(void(^)(id obj,NSError *error))com
```

- **获取标签**

```
SDK提供获取标签方法，可通过ShareInstallPushApiManager的getTagsWithComplate获取标
/**
 获取用户标签
 @param complate 回调函数
 */
+ (void)unsetTags:(NSArray *)tags complate:(void(^)(id obj,NSError *error))c
```

- **卸载标签**

```
SDK提供标签卸载方法，可通过ShareInstallPushApiManager的unsetTags
卸载
/**
 卸载用户标签
 @param tags 标签集合 ,英文逗号隔开 如 上海,男,最多30个
 */
+ (void)getTagsWithComplate:(void(^)(id obj,NSError *error))complate;
```

- **本地通知**

```
SDK提供了一个发送本地通知的API，
/**
 发送一个本地推送
 @param title :推送内容的标题
 subTitle:推送内容的子标题
 timeInterval:推送延迟时间，大于0 不能为0，若triggerWithTimeInterval:repeats方法
              否则会报*** Terminating app due to uncaught exception 'NSInterr
               reason: 'time interval must be at least 60 if repeating'的错误
 body:推送的主体内容
```

```
 userInfo:远程推送的推送内容
 */
- (void)sendLocalNotificationWithTitle:(NSString *)title
                              subTitle:(NSString *)subTitle
                          timeInterval:(NSInteger)timeInterval
                                  body:(NSString *)body
                              userInfo:(NSDictionary *)userInfo;

注意：
1：timeInterval:推送延迟时间，大于0 不能为0，若triggerWithTimeInterval:repeats方
2：userInfo的格式为字典格式。
```

六：推送上报

SDK中推送上报分三种：到达、点击、展现；

- 针对APNS的推送（通知栏），SDK会自动捕捉推送APNS代理方法的调用，然后触发上报。（同时上报到达、点击、展现）；
- 针对TCP服务的推送，SDK中会判断接入方是否实现了OnLine和offLine两个代理方法，当接入方实现了以上两个代理方法时，SDK会直接上报推送消息的到达、点击、展现三个上报。若接入方未实现，SDK只上报推送消息的到达。
- 以上两点均为SDK自动上报，接入方也可以通过调用API进行上报

```
 /**推送上报
 @param aps 远程推送消息
 @param handeleType 消息到达的类型
 @param actionType 消息的操作类型
 */
+ (void)submitPushInfo:(NSDictionary *)aps
          handeleType:(NotificationHandleType)handeleType
           actionType:(NotificationActionType)actionType;
```