信鸽 iOS SDK 开发指南

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。

接入方法

1. 获取 AppId 和 AppKey
2. 工程配置

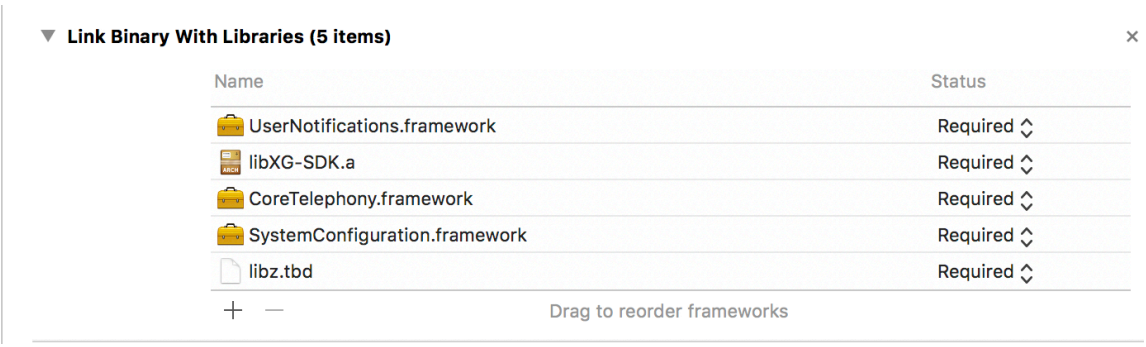
获取 AppId 和 AppKey

前往http://xg.qq.com注册并获取AppKey

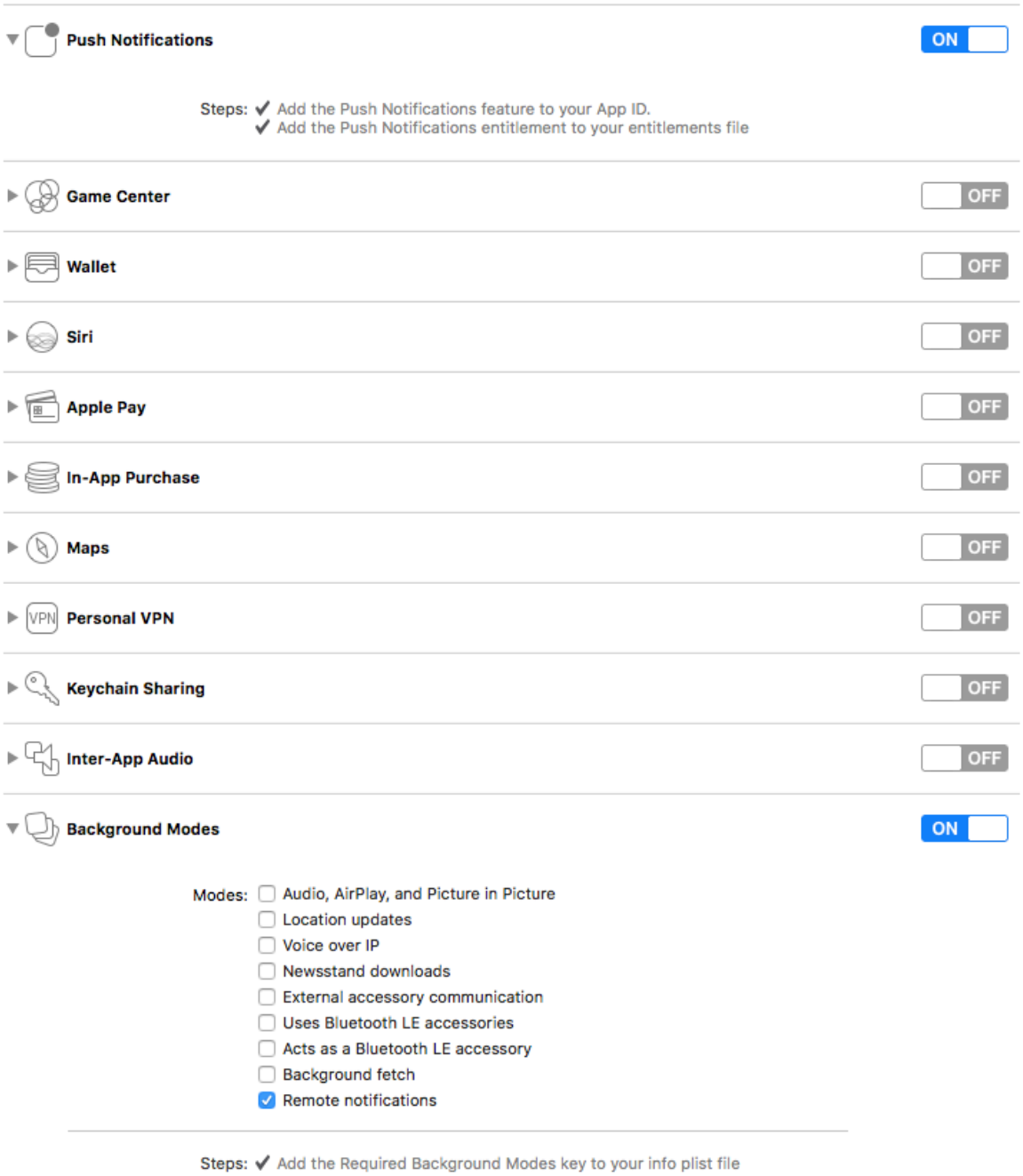
工程配置

1. 下载信鸽 SDK, 解压缩
2. 将XGPush.h 以及 libXG-SDK.a 添加到工程
3. 添加以下库/framework 的引用 CoreTelephony.framework, SystemConfiguration.framework, UserNotifications.framework, libXG-SDK.a 以及 libz.tbd.添加完成以后,库的引用如下:

   3.0.0以上版本需要添加libsqlite3.tbd

4. 在工程配置和后台模式中打开推送,如下图



5. 参考 Demo, 添加相关代码

管理信鸽推送服务

开启 Debug

打开 Debug 模式以后可以在终端看到详细的信鸽 Debug 信息，方便定位问题

示例

```
//打开debug开关
[[XGPush defaultManager] setEnableDebug:YES];
//查看debug开关是否打开
BOOL debugEnabled = [[XGPush defaultManager] isEnableDebug];
```

启动信鸽推送服务

接口

```
- (void)startXGWithAppID:(uint32_t)appID appKey:(nonnull NSString *)appKey
delegate:(nullable id<XGPushDelegate>)delegate ;
```

示例

```
[[XGPush defaultManager] startXGWithAppID:2200262432 appKey:@"I89WTUY132GJ"
delegate:<#your delegate#>];
```

终止信鸽推送服务

终止信鸽推送服务以后，将无法通过信鸽推送服务向设备推送消息

接口

```
- (void)stopXGNotification;
```

示例

```
[[XGPush defaultManager] stopXGNotification];
```

统计推送效果

为了更好的了解每一条推送消息的运营效果，需要将用户对消息的行为上报

统计消息推送的抵达情况

需要在UIApplicationDelegate的回调方法(如下)中调用上报数据的接口

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

接口

```
- (void)reportXGNotificationInfo:(nonnull NSDictionary *)info;
```

示例

```
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
      [[XGPush defaultManager] reportXGNotificationInfo:launchOptions];
}
```

统计消息被点击情况

- iOS 10 以前的系统版本，需要在 UIApplicationDelegate 的回调方法(如下)中调用上报数据的接口

```
- (void)application:(UIApplication *)application
    didReceiveRemoteNotification:(NSDictionary *)userInfo;
```

接口

```
- (void)reportXGNotificationInfo:(nonnull NSDictionary *)info;
```

示例

```
- (void)application:(UIApplication *)application
    didReceiveRemoteNotification:(NSDictionary *)userInfo {
    [[XGPush defaultManager] reportXGNotificationInfo:userInfo];
}
```

- iOS 10 or later 需要实现 XGPushDelegate 的回调方法(如下),并在其中调用上述上报接口

```
- (void)xgPushUserNotificationCenter:(UNUserNotificationCenter *)center
    didReceiveNotificationResponse:(UNNotificationResponse *)response
    withCompletionHandler:(void(^)())completionHandler;
```

示例:

```
- (void)xgPushUserNotificationCenter:(UNUserNotificationCenter *)center
    didReceiveNotificationResponse:(UNNotificationResponse *)response
    withCompletionHandler:(void(^)())completionHandler {
      [[XGPush defaultManager]
reportXGNotificationInfo:response.notification.request.content.userInfo
];

    completionHandler()
}
```

- 如果需要实现应用在前台时，也可以展示推送消息，需要实现以下方法，并在其中调用上报接口

```
- (void)xgPushUserNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void (^)
(UNNotificationPresentationOptions))completionHandler
```

示例

```
- (void)xgPushUserNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void (^)
(UNNotificationPresentationOptions))completionHandler {
[[XGPush defaultManager]
reportXGNotificationInfo:notification.request.content.userInfo];
    completionHandler(UNNotificationPresentationOptionBadge |
UNNotificationPresentationOptionSound |
UNNotificationPresentationOptionAlert);
}
```

信鸽推送服务协议(XGPushDelegate)

设置信鸽推送协议代理对象是为了方便查看接口调用的情况，开发者可根据自己的需求选择实现协议的方法

协议方法如下：

```
/**
 处理iOS 10 UNUserNotification.framework的对应的方法

 @param center [UNUserNotificationCenter currentNotificationCenter]
 @param notification 通知对象
 @param completionHandler 回调对象，必须调用
 */
- (void)xgPushUserNotificationCenter:(nonnull UNUserNotificationCenter
*)center willPresentNotification:(nullable UNNotification *)notification
withCompletionHandler:(nonnull void (^)(UNNotificationPresentationOptions
options))completionHandler __IOS_AVAILABLE(10.0);


/**
 处理iOS 10 UNUserNotification.framework的对应的方法

 @param center [UNUserNotificationCenter currentNotificationCenter]
 @param response 用户对通知消息的响应对象
 @param completionHandler 回调对象，必须调用
 */
- (void)xgPushUserNotificationCenter:(nonnull UNUserNotificationCenter
*)center didReceiveNotificationResponse:(nullable UNNotificationResponse
*)response withCompletionHandler:(nonnull void (^)(void))completionHandler
__IOS_AVAILABLE(10.0);

/**
 @brief 监控信鸽推送服务地启动情况

 @param isSuccess 信鸽推送是否启动成功
 @param error 信鸽推送启动错误的信息
 */
```

```
- (void)xgPushDidFinishStart:(BOOL)isSuccess error:(nullable NSError
*)error;

/**
 @brief 监控信鸽服务的终止情况

 @param isSuccess 信鸽推送是否终止
 @param error 信鸽推动终止错误的信息
 */
- (void)xgPushDidFinishStop:(BOOL)isSuccess error:(nullable NSError
*)error;


/**
 @brief 监控信鸽服务上报推送消息的情况

 @param isSuccess 上报是否成功
 @param error 上报失败的信息
 */
- (void)xgPushDidReportNotification:(BOOL)isSuccess error:(nullable NSError
*)error;
```

绑定/解绑标签和账号

开发者可以针对不同的用户绑定标签,然后对该标签推送.对标签推送会让该标签下的所有设备都收到推送.一个设备可以绑定多个标签.

接口

```
- (void)bindWithIdentifier:(nullable NSString *)identifier type:
(XGPushTokenBindType)type;
- (void)unbindWithIdentifer:(nullable NSString *)identifier type:
(XGPushTokenBindType)type;
```

示例

```
绑定标签:
[[XGPushTokenManager defaultTokenManager] bindWithIdentifier:@"your tag"
type:XGPushTokenBindTypeTag];

解绑标签
[[XGPushTokenManager defaultTokenManager] unbindWithIdentifer:@"your tag"
type:XGPushTokenBindTypeTag];

绑定账号:
  [[XGPushTokenManager defaultTokenManager] bindWithIdentifier:@"your
account" type:XGPushTokenBindTypeAccount];

解绑账号:
[[XGPushTokenManager defaultTokenManager] unbindWithIdentifer:@"your
account" type:XGPushTokenBindTypeAccount];
```

注1: 一个设备只能绑定一个账号,绑定账号的时候前一个账号自动失效.一个账号最多绑定15台设备,超过之后会随机解绑一台设备,然后再进行注册.

管理设备Token协议(XGPushTokenManagerDelegate)

设置设备token绑定协议代理对象是为了方便查看token绑定的结果，开发者可根据自己的需求选择实现协议的方法

示例

```
[[XGPushTokenManager defaultTokenManager].delegate = <#your delegate#>;
```

协议方法如下:

```
/**
 @brief 监控token对象绑定的情况

 @param identifier token对象绑定的标识
 @param type token对象绑定的类型
 @param error token对象绑定的结果信息
 */
- (void)xgPushDidBindWithIdentifier:(nullable NSString *)identifier type:
(XGPushTokenBindType)type error:(nullable NSError *)error;


/**
 @brief 监控token对象解绑的情况

 @param identifier token对象绑定的标识
 @param type token对象绑定的类型
 @param error token对象绑定的结果信息
 */
```

```
- (void)xgPushDidUnbindWithIdentifier:(nullable NSString *)identifier type:
(XGPushTokenBindType)type error:(nullable NSError *)error;
```

iOS Extension SDK API (iOS 10+)

上报推送消息回执，此接口的目的是统计推送消息是否抵达终端

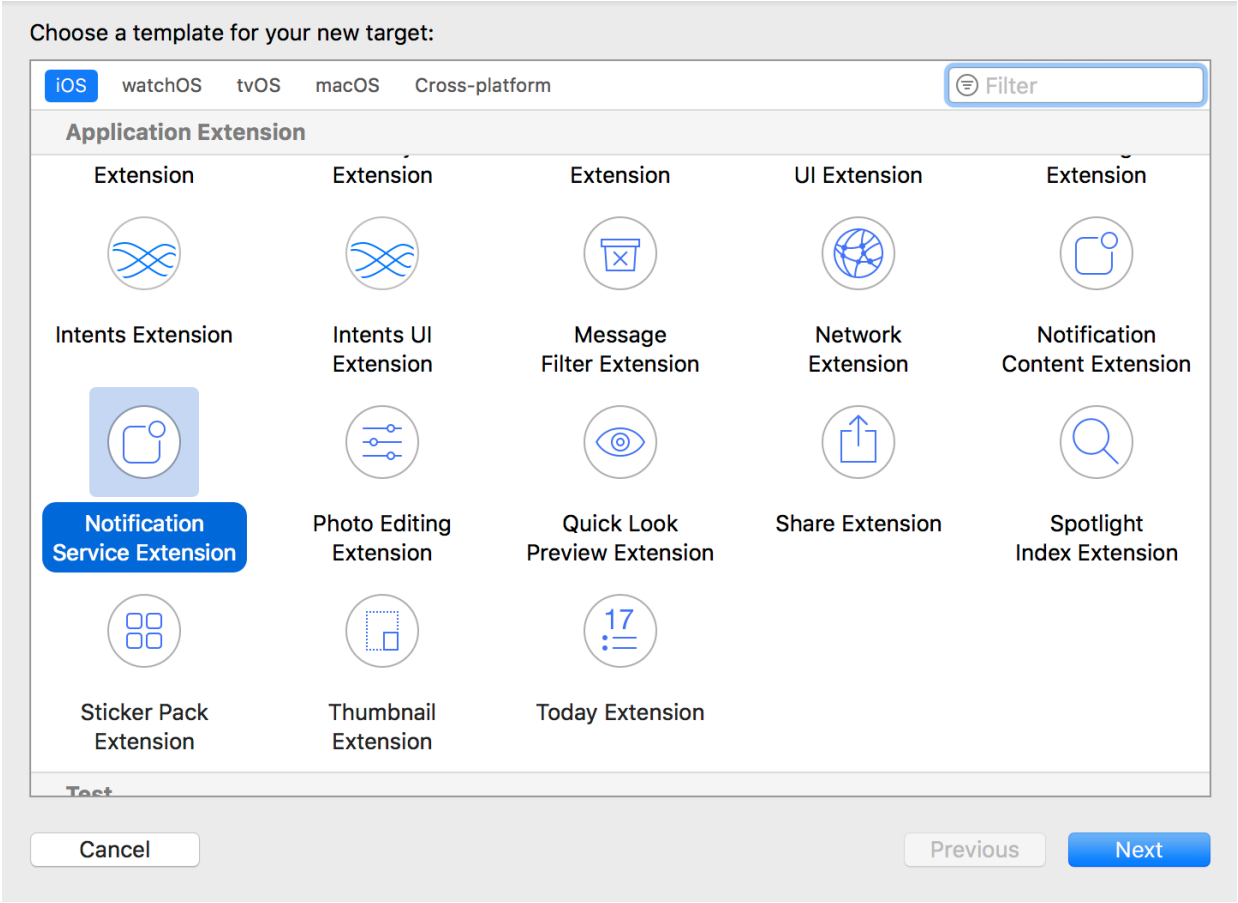
```
/**
 @brief 信鸽推送处理抵达到终端的消息

 @param request 推送请求
 @param appID 信鸽应用ID
 @param handler 处理消息的回调，回调方法中处理关联的富媒体文件
 @note 关联的富媒体文件，需要在推送前端设置资源的完整URL地址，SDK内部会自动处理下载
 */
- (void)handleNotificationRequest:(nonnull UNNotificationRequest *)request
appID:(uint32_t)appID contentHandler:(nullable void(^)( NSArray
<UNNotificationAttachment *>* _Nullable attachments,  NSError * _Nullable
error))handler;
```

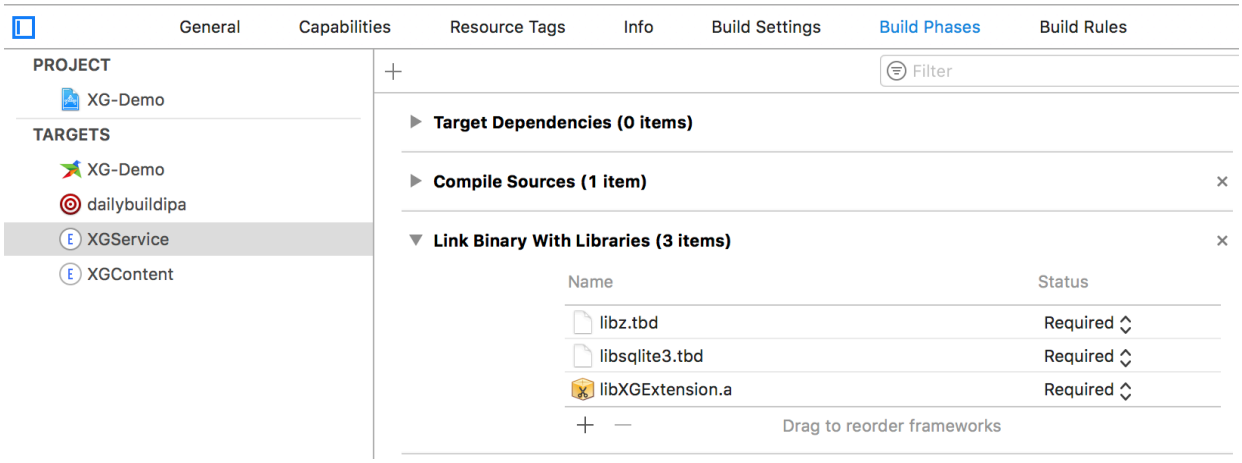
说明：

为了使用extension SDK，操作步骤如下：

1.新增Target



2.配置Target，添加依赖库文件：libXGExtension.a, libz.tbd, libsqlite3.tbd



3.调用SDK统计上报接口

示例

```
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request
withContentHandler:(void (^)(UNNotificationContent *
_Nonnull))contentHandler {
    [[XGExtension defaultManager] handleNotificationRequest:request appID:
<#your xg AppID#> contentHandler:nil];
    self.contentHandler = contentHandler;
    self.bestAttemptContent = [request.content mutableCopy];
    self.contentHandler(self.bestAttemptContent);
}
```

本地推送

本地推送相关功能请参考[苹果开发者文档](#).