北京基调网络股份有限公司

# 听云平台部署说明

听云 APP iOS SDK 部署说明

Chengmy

2018/7/24

# 前言

## 产品版本

与本手册相对应的产品版本如下所示。

| 产品名称 | 产品版本 | 手册版本 |
| --- | --- | --- |
| iOS 探针 | 2.4.4 | V1.0 |
| iOS 探针 | 2.5.0 | V2.0 |
| iOS 探针 | 2.6.0 | V2.1 |
| iOS 探针 | 2.7.0 | V2.2 |
| iOS 探针 | 2.8.0 | V2.3 |
| iOS 探针 | 2.9.0 | V2.4 |
| iOS 探针 | 2.9.0 | V2.5 |
| iOS 探针 | 2.10.0 | V2.6 |
| iOS 探针 | 2.11.0 | V2.7 |

## 内容介绍

本手册主要介绍了 iOS SDK 探针的 Framework 部署方法及步骤。

## 读者对象

本手册适用于以下人员：

- iOS 开发工程师
- iOS 测试工程师
- 网络监控工程师
- 系统维护工程师

## 约定

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

| 符号 | 说明 |
| --- | --- |
| 🔔 | 以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。 |
| 📖 | 以本标志开始的文本是正文的附加信息，是对正文的强调和补充。 |

# 修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有版本的更新内容。

## 版本 V2.7（2018-10-12）

- 新增

| 位置 | 描述 |
| --- | --- |
| 2.1.1 通过 Framework 安装 iOS 探针 | 增加 libc++.tbd |

## 版本 V2.6（2018-07-24）

- 新增

| 位置 | 描述 |
| --- | --- |
| 2.1.1 通过 Framework 安装 iOS 探针 | 增加 WebKit.framework |

## 版本 V2.5（2018-05-24）

- 新增

| 位置 | 描述 |
| --- | --- |
| 简介 | 更新内容，新增 SDK 支持采集数据类型 |
| FAQ | 新增常见问题解答 |

## 版本 V2.4（2018-04-26）

- 新增

| 位置 | 描述 |
| --- | --- |
| iOS SDK 部署 | 增加 URL Scheme 配置 |

## 版本 V2.3（2018-01-12）

- 新增

| 位置 | 描述 |
| --- | --- |
| 简介 | 增加 appStarts 应用启动性能数据采集限制说明 |

## 版本 V2.2（2017-09-28）

- 新增

| 位置 | 描述 |
| --- | --- |
| 简介 | 增加 webview 页面性能数据采集系统限制说明 |

## 版本 V2.1（2017-03-30）

- 新增

| 位置 | 描述 |
| --- | --- |
| 2.1.1 通过 Framework 安装 iOS 探针 | 增加 CFNetwork |

## 版本 V2.0（2016-10-27）

- 新增

| 位置 | 描述 |
| --- | --- |
| 简介 | 简介 |
| 高级功能 | 高级功能 |

## 版本 V1.0（2014-06-18）

手册初次发行

# 目录

**前言** ..... **1**
产品版本 ..... 1
内容介绍 ..... 1
读者对象 ..... 1
约定 ..... 1
*符号约定* ..... *1*
修改记录 ..... 2
*版本 V2.6（2018-07-24）* ..... *2*
*版本 V2.5（2018-05-24）* ..... *2*
*版本 V2.4（2018-04-26）* ..... *2*
*版本 V2.3（2018-01-12）* ..... *3*
*版本 V2.2（2017-09-28）* ..... *3*
*版本 V2.1（2017-03-30）* ..... *3*
*版本 V2.0（2016-10-27）* ..... *3*
*版本 V1.0（2014-06-18）* ..... *3*

**目录** ..... **4**

**1 简介** ..... **5**
1.1 IOS SDK 目录结构 ..... 7
1.2 IOS SDK THEORY ..... 7
1.3 IOS SDK 体积增量 ..... 7

**2 IOS SDK 部署** ..... **8**
2.1 安装 SDK ..... 8
*2.1.1 通过 Framework 安装 SDK* ..... *8*
*2.1.2 通过 CocoaPods 安装 SDK* ..... *10*
*2.1.3 通过 Carthage 安装 SDK* ..... *10*
2.2 插入初始化探针代码 ..... 11
2.3 添加 URL SCHEME ..... 11
2.4 嵌码完整性校验 ..... 12

**3 创建.PCH 文件** ..... **12**
3.1 新建 PCH 文件 ..... 12

**4 高级功能** ..... **14**
4.1 面包屑 ..... 14
4.2 自定义 EVENT ..... 14
4.3 自定义 TRACE ..... 14
4.4 自定义 ID ..... 15

5 FAQ ..... 15

# 1 简介

iOS SDK 探针支持如下数据采集：

- Http/Https 协议数据收集
- 崩溃/卡顿异常数据收集
- 事件性能数据收集
- 视图性能数据收集
- 进程内存和 CPU 使用率
- 用户体验分析数据收集

iOS SDK 探针支持如下协议类库：

- NSURLConnection / NSURLSession(仅支持 iOS 8 以上的系统)
- ASI
- AFN(基于 NSURLConnection)
- webView(仅支持 UIWebView)

iOS SDK 探针支持以下系统版本：

- iOS 7.0 版本以上
- iOS 10.2 及其以上版本支持 TCP 数据采集且无法采集 Server IP

iOS SDK 通过在指定的方法中嵌码去采集下列方法中的数据。可以通过报表服务器查询各种方法的数据和调用次数。

iOS SDK 采集应用的网络数据。

- 应用响应时间
- DNS 解析时间
- TCP 建连时间
- SSL 握手时间
- 首包时间
- 访问量

iOS SDK 探针也可以采集线程堆栈，数据库、自定义参数和 HTTP 请求参数。

用户体验分析-应用启动数据。

- 启动耗时
- 启动次数
- 启动期间的崩溃、事件性能数据

用户体验分析-用户操作数据。

- 操作耗时
- 阻塞耗时
- 操作次数
- 操作期间的崩溃、事件性能数据

## 1.1 iOS SDK 目录结构

- tingyunApp.framework

## 1.2 iOS SDK Theory

```
/*
 * swizzle 过后，imageName 指向的是 nbs_imageName 函数的实现，当在调用 imageName 这个函数时，会首先调用 nbs_imageNamed 的实现，
 * 而此时 nbs_imageName 指向的是原函数 imageName 的实现，所以在 nbs_imageNamed 的实现中通过[self nbs_imageNamed:name]回调回原函数
 */
+ (UIImage *)nbs_imageNamed:(NSString *)name
{
    nbs_embedIn_start(nil, category_image, classNameOf(self), NSStringFromSelector(_cmd), currentViewController, timeNow());//SDK 开始数据采集
    id rtn= [self nbs_imageNamed:name];//调用原函数实现
    nbs_embedIn_finish(argsHelper);//SDK 完成数据采集
    return rtn;
}
```

利用 objective-c 的 runtime 特性,通过 Method swizzle 技术,可以实现在运行时替换 selector 对应的方法实现,达到 给方法挂钩的目的。也就是说,嵌入基调的 SDK 后,在程序启动之初，基调的 SDK 会对相应的方法执行 swizzle 操作，从而在调用一个被 swizzle 过后的函数时，将会首先调用 SDK 相应的自定义函数，在 SDK 的函数中会执行一些数据采集的操作，然后 SDK 的函数会再调回原函数的实现,不会影响原程序逻辑。

## 1.3 iOS SDK 体积增量

三种指令架构增加 2.71M

- Armv7 增加 0.81M
- Armv7s 增加 0.81M
- Arm64 增加 0.99M

# 2 iOS SDK 部署

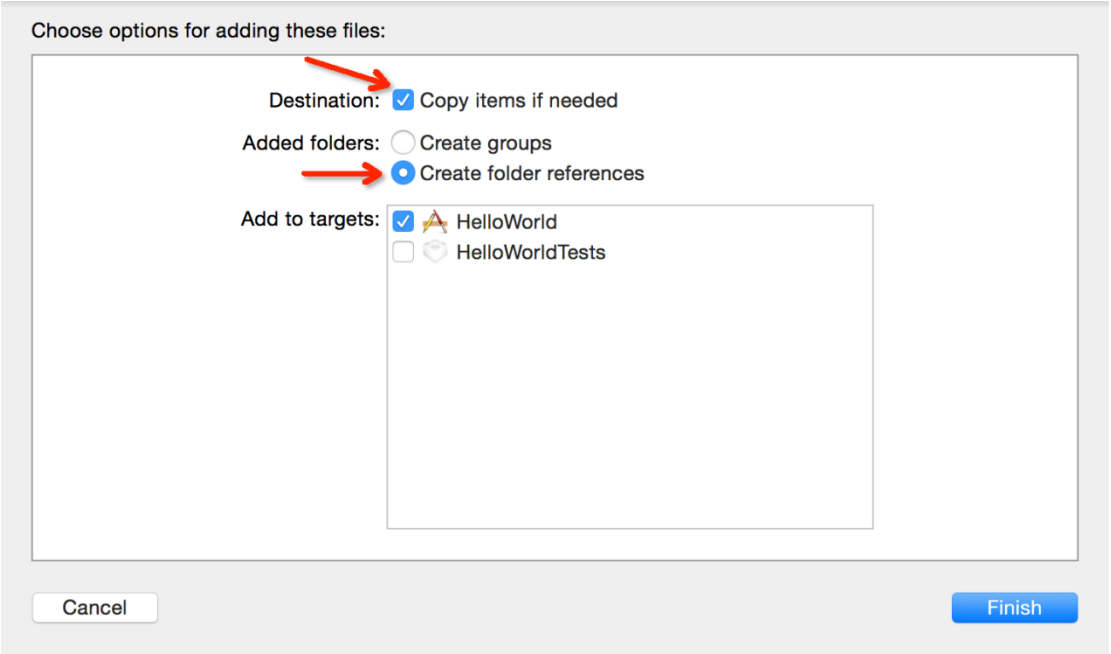
## 2.1 安装 SDK

### 2.1.1 通过 Framework 安装 SDK

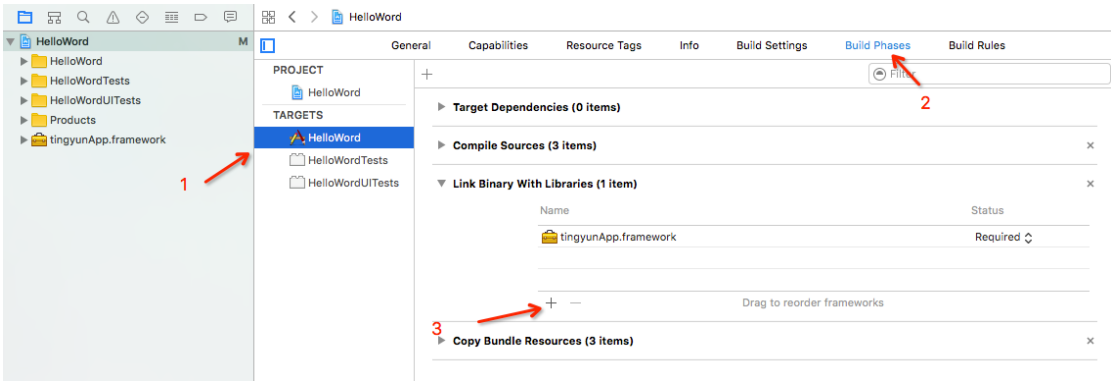
1. 下载并解压 nbs-newlens-ios.zip 探针压缩包
2. 在解开的文件夹中，选择“tingyunApp.framework”文件夹，并将其拖动到需要进行监测的 iOS App 的 Xcode 项目中



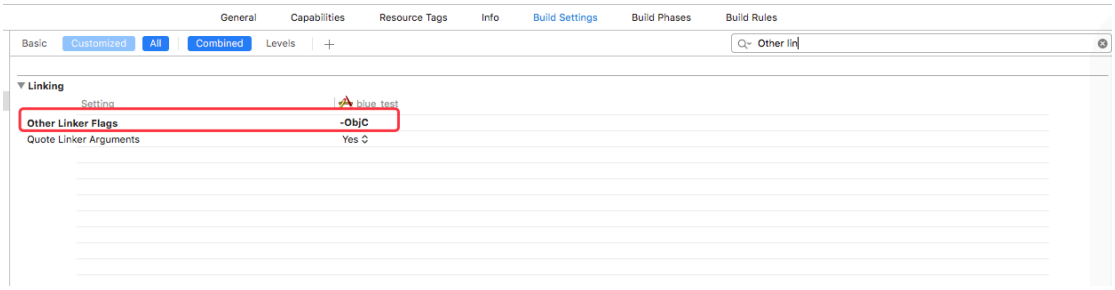

3. 在 Xcode 中，选择 Project Navigator 窗口，将拖动的文件夹图标放到“HelloWorld”文件夹组下。




4. 当 Xcode 弹出提示时，选择“Copy items into destination group’s folder”和“Create folder references for any added folders”选项，并确认。




5. 选择 Build Setting，搜索 Other Linker Flags，增加编译标志-ObjC

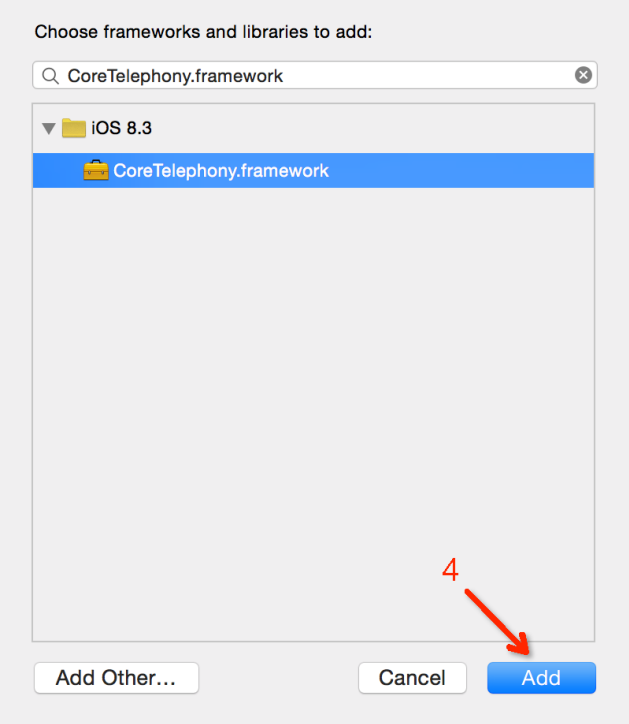



6. 在待监测的 iOS App 项目的链接设置中增加以下 6 个库，在 Xcode 的 Project Navigator 窗口中选中待监测 App 项目：

   a) 在 Target 中选中对应的 App；

   b) 切换到“Build Phases”设置页面；

   c) 展开“Link Binary With Libraries”设置项，点击“+”号进入增加库的窗口




7. 增加以下六个库：

| 库名 | 作用 |
| --- | --- |
| CoreTelephony.framework | 获取运营商 |
| Security.framework | 访问钥匙链 |
| SystemConfiguration.framework | 获取网络状态 |
| libz.tbd | 解压缩 |
| WebKit.framwork | 浏览器内核 |
| libc++.tbd | C++符号化 |

## 2.1.2 通过 CocoaPods 安装 SDK

在项目工程的 Podfile 文件中添加：pod ‘tingyunApp’

## 2.1.3 通过 Carthage 安装 SDK

在项目工程的 Cartfile 文件中添加：github ”TingYunAPM/TingYunApp”

## 2.2 插入初始化探针代码

Xcode 6 之后，创建工程时不会自动创建 pch 文件，需要自己创建。
Swift 工程请在对应的 Bridging-Header.h 文件中导入头文件。

1. 在待监测 App 的 pch 文件中引入听云 App 探针的头文件：

```
#import <tingyunApp/NBSAppAgent.h>
```

2. Swift 引入头文件

```
#ifndef testswift_Bridging_Header_h
#define testswift_Bridging_Header_h
#import <tingyunApp/NBSAppAgent.h>
#endif /* testswift_Bridging_Header_h */
```

3. 在嵌码项目工程 Supporting Files/main.m 文件的 main 函数中添加以下代码：

```
//2.1.2 以下版本需将代码添加到 application:didFinishLaunchingWithOptions 中
[NBSAppAgent startWithAppID:@"YOUR_AppKey"];
```

4. Swift 工程请在 AppDelegate.swift 文件的 application:didFinishLaunchingWithOptions 函数中添加以下代码：

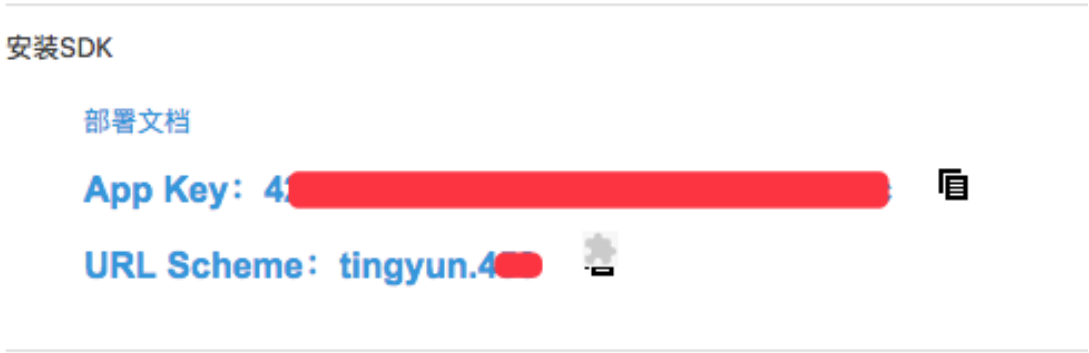
```
NBSAppAgent.startWithAppID("YOUR_APPKEY");
```

## 2.3 添加 URL Scheme

将 URL Scheme 添加到项目中，以便我们唤醒您的应用，进行点选。

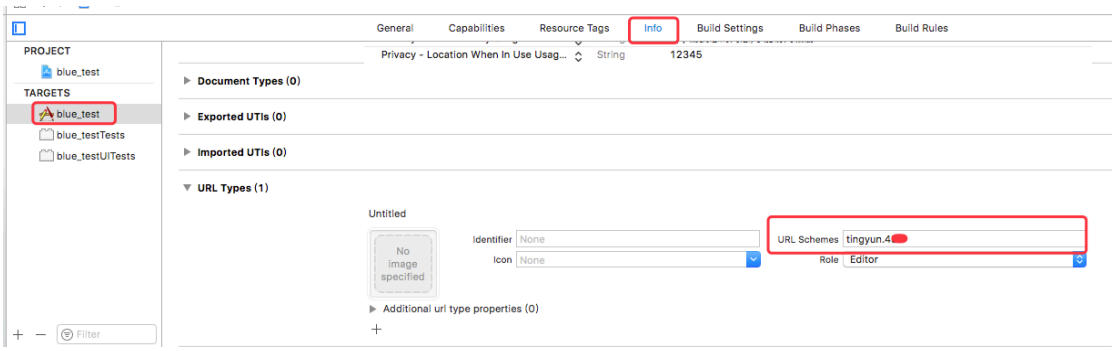
**URL Scheme 获取**

登录控制台，进入 APP 概览页 –> 点击小齿轮 –> 点击修改设置，进入到修改设置详情页面，URL Scheme 就在 appkey 下面，格式为：tingyun.appid。




**URL Scheme 配置**

将获取到的 URL Scheme 添加到 target –> info –> URL Types 。




## 2.4 嵌码完整性校验

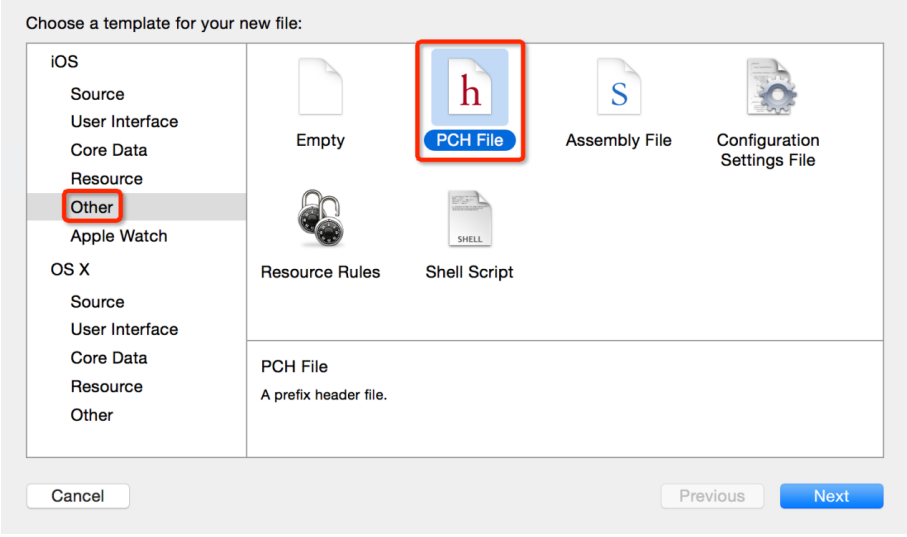
嵌码完成后可以在控制台中查看听云 SDK 日志输出结果：

```
NBSAppAgent  SDK_Version
---->start!
Success to connect to NBSSERVER
```

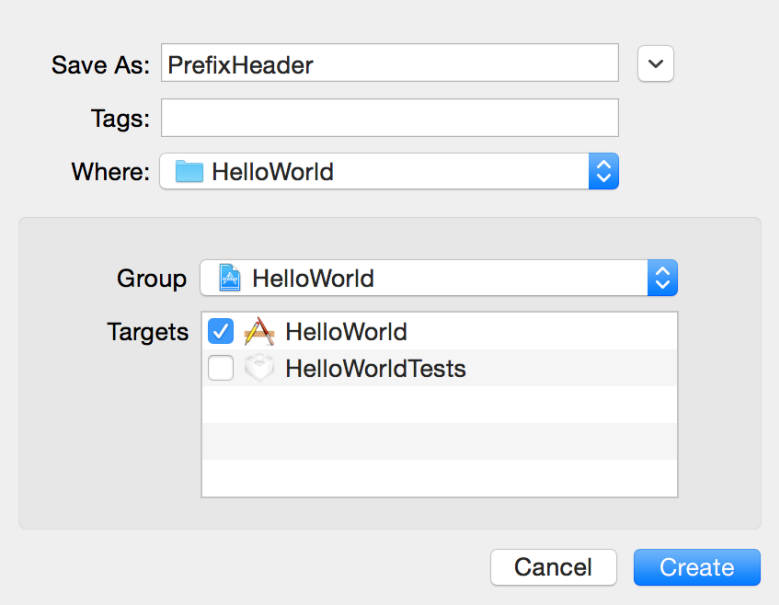
# 3 创建.pch 文件

## 3.1 新建 pch 文件

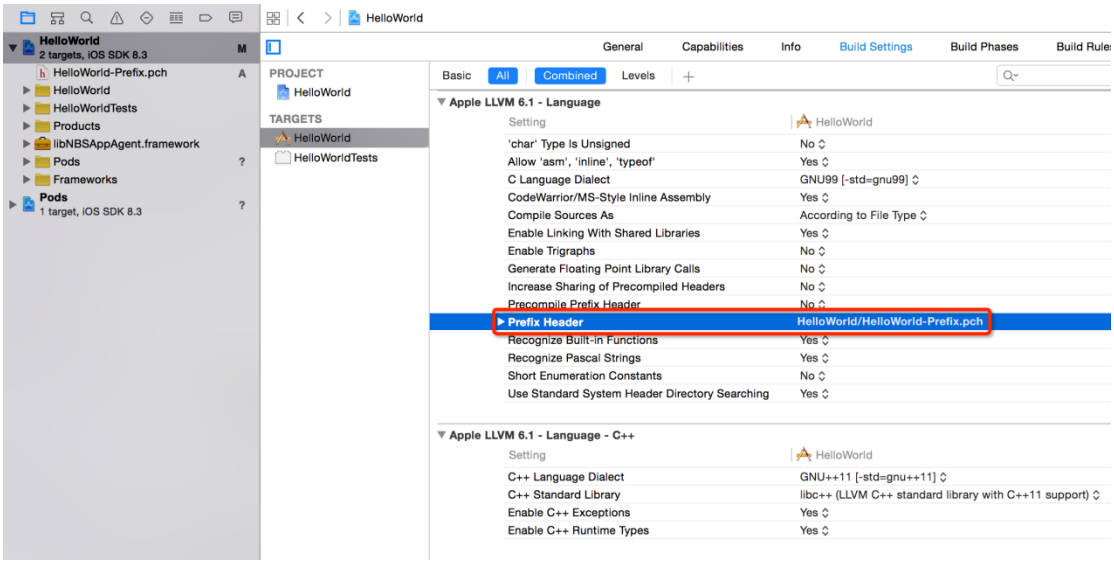
1. 打开 Xcode，依次选择 File→New→File（或者⌘N）。




2. 在弹出的对话框中选择 Other，然后选择 PCH File。




3. 点击 Next，然后输入“pch”的文件名，格式一般为“工程名-Prefix”。




4. 选中工程→TARGETS→Build Settings，找到 Apple LLVM 6.0-Language，其中有一项 Prefix Header，双击它，然后输入“工程名/pch 文件名”，到这里 pch 文件就设置好了。

# 4 高级功能

## 4.1 面包屑

1. 功能说明
   面包屑能够更好的协助用户调查崩溃发生的原因，可以知晓用户发生崩溃之前的代码逻辑与崩溃轨迹结合使用能够更好的复现用户崩溃场景。

2. 相关接口

```
//最多包含 100 个字符，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
[NBSAppAgent leaveBreadcrumb:@"keyPressed”];
[NBSAppAgent leaveBreadcrumb:@"loginDone"];
```

3. 代码示例

```
-(BOOL)application:(UIApplication *)application
didFinishLaunchingWithOption:(NSDictionary *)launchOptions
{
[NBSAppAgent leaveBreadcrumb:@“didFinishLaunchingWithOptions”];设置面包屑
return YES;
}
```

## 4.2 自定义 Event

1. 功能说明
   自定义事件用于统计 App 中的任意事件，开发者可以在 SDK 初始化后的任意位置添加自定义事件，并设置对应上传参数。如：真实用户操作时候点击某个功能按钮或触发了某个功能事件等。

2. 相关接口

```
//EVENT_ID 最多包含 32 个字符，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
[NBSAppAgent trackEvent:@”EVENT_ID”];
```

3. 代码示例

```
-(void)purchaseFailedCallback
{
[NBSAppAgent trackEvent:@“购买失败”];
}
```

## 4.3 自定义 Trace

由于自定义 Trace 是成对出现的，请勿跨方法、跨进程以及在异步加载和递归调用中使用该接口。

1. 功能说明
   听云 SDK 默认采集系统类和方法的性能数据，无法采集开发者自定义类和方法

的性能数据。使用“自定义 Trace”接口就可以帮助开发者时刻了解所写代码的健壮性及其性能数据。如：开发者想要了解某个自定义方法的初始化耗时及性能消耗情况，就可以在该自定义方法前后添加“自定义 Trace”接口即可。

2. 相关接口

```
//Name 为当前方法所在方法名或自定义名称，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
beginTracer(@"String Name")
endTracer(@"String  Name")
```

3. 代码示例

```
//用户可以在 SDK 初始化后的任意方法前后添加自定义 Trace
- (void)doSomething
 {
  beginTracer(@"String Name")
 //write you code here

 endTracer(@"String Name")
}
```

## 4.4 自定义 ID

1. 功能说明

用户自定义 ID 为当前用户设置唯一标示码，在任意位置均可设置 UserID。

2. 相关接口

```
[NBSAppAgent startWithAppID:@"Appkey"];
[NBSAppAgent setUserIdentifier:@"userID"];
```

3. 代码示例

[NBSAppAgent setUserIdentifier:@"test_uid"];//空串代码示例

```
-(void)viewDidLoad{
 [super viewDidLoad];
 [NBSAppAgent setUserIdentifier:"UserIdentifier"];//设置自定义 Id,可为邮箱等标识用户身份的信息，如 xxx@tingyun.com
 }
```

# 5 FAQ

1. 引入 framework 后找不到头文件？

```
(1).检查 Framework Search Paths 和 Head Search Paths 中的路径是否填写正确
(2).确认是否将 tingyunApp.framwork 文件“Copy”到工程中
```

2. 使用 CocoaPods 导入 SDK 报错(The dependency ’ tingyunApp’ is not used in any concrete target)

```
出错原因是没有用到指定的 target，解决办法：在 Podfile 中添加如下内容，再执行 pod install(或者 pod  update)：
            target ’MyApp’  do
```

```
pod ‘tingyunApp’
end (MyApp 请替换为自己工程里面的 target)
```

3. 执行 carthage update 报错

进入到~/Library/Caches/org.carthage.CarthageKit/dependencies/目录下，将对应的 tingyunApp 文件夹删除，然后在工程的根目录下执行 carthage update

4. 嵌码成功后，运行时无 log 日志？

(1).确认是否添加初始化代码
(2).确认是否重写过 NSLog，如果重写的话可能会导致 SDK 打印的日志无法显示

5. sdk 正常启动，但是总连接不到服务器

(1).检查报表端是否禁用了当前应用
(2).确认当前月活是否超过套餐购买月活数
(3).确认安全验证里是否设置了 bundleID，并检查其正确性

6. 嵌码成功后，进入报表”情报汇总”后发现，报表只展示活跃设备

用户使用的网络协议库是否支持，iOS SDK 支持的协议库：NSURLConnection 、ASIHttpRequest、AFNetworking、UIWebview、 NSURLSession(iOS 8 及其以上版本)

7. 报表网络数据中无法采集 TCP 数据

(1).SDK 2.5.4 版本之后，SDK 去除 socket 的模块，iOS8 及其以下系统的无法采集 TCP 数据
(2).SDK 2.5.4 版本之前，iOS 9、10、11 无法采集 TCP 数据
(3).SDK 2.5.0 版本之后，只有 iOS 10.2 及其以上设备的 NSURLSession 请求（设置 delegate 方法）、Webview 请求以及 AFNetworking 请求才可以采集到 TCP 数据

8. 无法采集 NSURLSession 网络数据

确认是否为 iOS 8 及其以上的设备，听云 SDK 在 iOS 8 以后才支持 NSURLSession

9. URL/错误过滤未生效

(1).一般是正则表达式配置问题，可以通过在线正则工具检查，需注意报表配置正则的前后需要加斜杠
(2).是否配置完毕之后立即查看？（保存配置后需要等待一两分钟，生效后即可登录报表查看）

10. 为什么崩溃没有采集到？

(1).听云 SDK 不支持对模拟器的崩溃采集
(2).是否嵌入其他 SDK？2.3.2 以后 SDK 已经兼容第三方 SDK 的崩溃采集，需将 SDK 初始化代码放置于其他第三方 SDK 初始化代码之后，以保证听云 SDK 可以正常采集到崩溃数据且第三方 SDK 也能采集到崩溃数据，故建议将听云 SDK 放在其他第三方 SDK 后进行初始化（注：如果启动位置正确还是无法采集到崩溃，可能是由于其他 SDK 做了特殊处理使得他们 SDK crash 采集的 handler 在听云 SDK crash 采集的 handler 之后启动）
(3).如果通过 Xcode 安装到手机后发生崩溃，听云 SDK 仅支持 Exception 类型崩溃采集
(4).若应用启动之后立即崩溃，则崩溃采集的 handler 不能及时启动，导致听云 SDK 无法采集崩溃数据
(5).如果为被系统杀掉或者是由于内存警告导致的闪退，则听云 SDK 无法采集这类数据
(6).听云 SDK 的崩溃采集逻辑是本次崩溃，需要下次启动之后才会上传崩溃数据
(7).确认报表是否开启崩溃采集开关

11. 为什么 WebView 性能数据采集不到？

(1).检查报表是否开启 WebView 模块开关
(2).确认 WebView 的 delegate 是否为 ViewController
(3).如果页面加载过程中没有调用 onload 或者 unload 事件，则听云 SDK 无法采集该 WebView 性能数据
(4).如果页面加载过程中调用了 onload 或者 unload 事件，但是访问的页面 url 却为”about:blank”，听云 SDK 无法采集该 WebView 性能数据
(5).确认设备是否为 iOS 10.2 及其以上系统

12. 无法采集启动时间数据?

(1).确认用户是否在 main 函数中嵌码，如果是其他位置嵌码，听云 SDK 无法采集启动时间数据

(2).检查报表是否开启 UI 模块开关，用户启动时间模块是由 UI 开关控制的，关闭之后听云 SDK 无法采集启动时间数据

13. 为什么上传 dsym 文件后，crash 堆栈解析不出来？

确认上传的 dsym 文件是否正确，步骤如下：

1、登录报表进入到崩溃详情界面，去掉格式化，查看 binaryImage 下第一行信息

2、查看 dsym UUID：打开终端进入到 dsym 文件的目录下， 执行 dwarfdump -u xxx （xxx 为 dsym 名字），执行后会显示出 dsym 的 UUID

3、校验步骤 1 和步骤 2 的 UUID 是否一致，若一致则上传的 dsym 文件匹配，但若结果不一致，则会导致报表崩溃堆栈不能解析