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 Framework 部署说明

Chengmy
2016/10/27

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

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 V2.0 （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

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

iOS SDK 探针支持如下协议类库：

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

iOS SDK 探针支持如下系统版本：

- iOS 6.0 版本以上
- socket 支持 iOS 6 ~ iOS 8 版本

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。

- 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。

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 2.3M

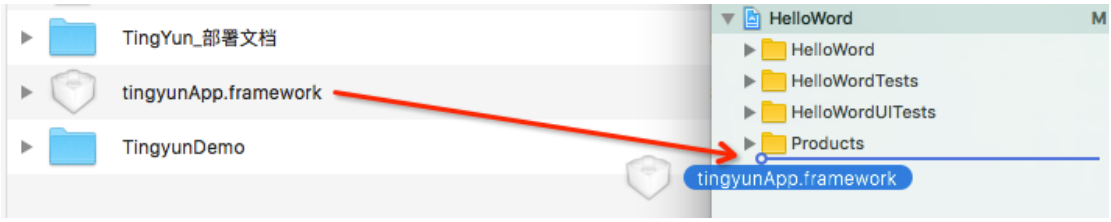
- Armv7 增加 0.7M
- Armv7s 增加 0.7M
- Arm64 增加 0.9M

2 iOS SDK 部署

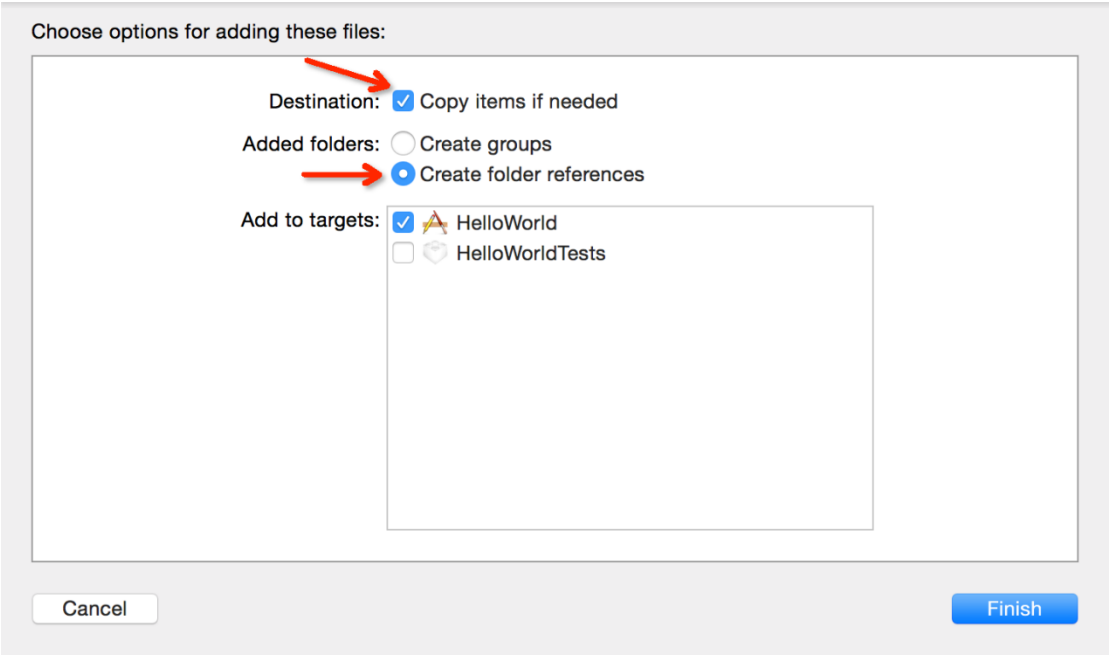
2.1 安装 iOS 探针

2.1.1 通过 Framework 安装 iOS 探针

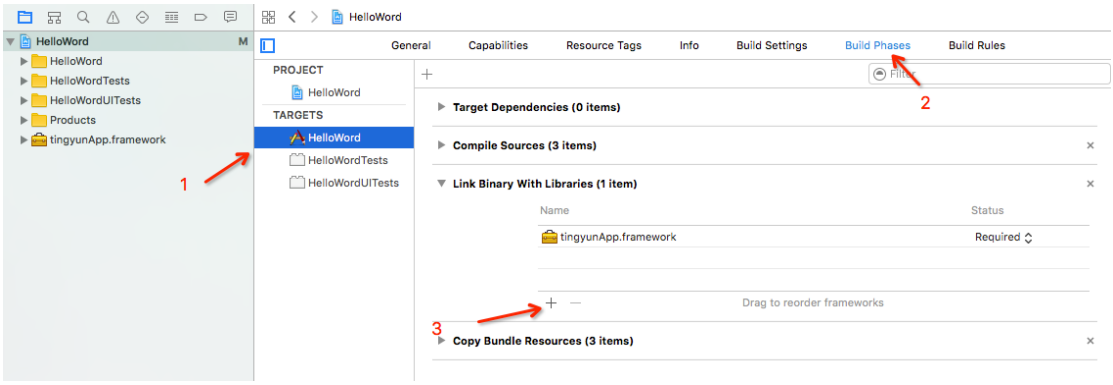
1. 下载并解压 nbs-newlens-ios.zip 探针压缩包
2. 在解开的文件夹中，选择“tingyunApp.framework”文件夹，并将其拖动到需要进行监测的 iOS App 的 Xcode 项目中



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。

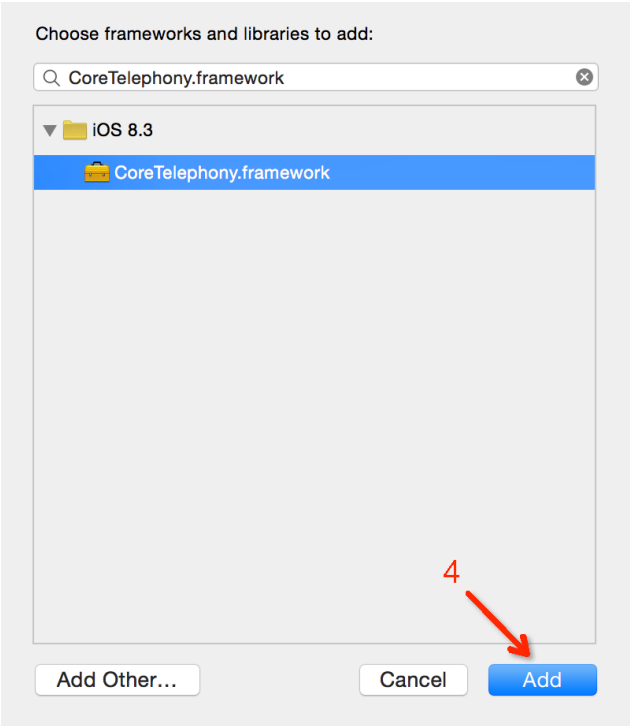


5. 在待监测的 iOS App 项目的链接设置中增加以下 4 个库，在 Xcode 的 Project Navigator 窗口中选中待监测 App 项目：

   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



6. 增加以下四个库：CoreTelephony.framework、Security.framework、SystemConfiguration.framework、libz.tbd

2.1.2 通过 CocoaPods 安装 iOS 探针

听云在 CocoaPods 里的文件名称为：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"];
```

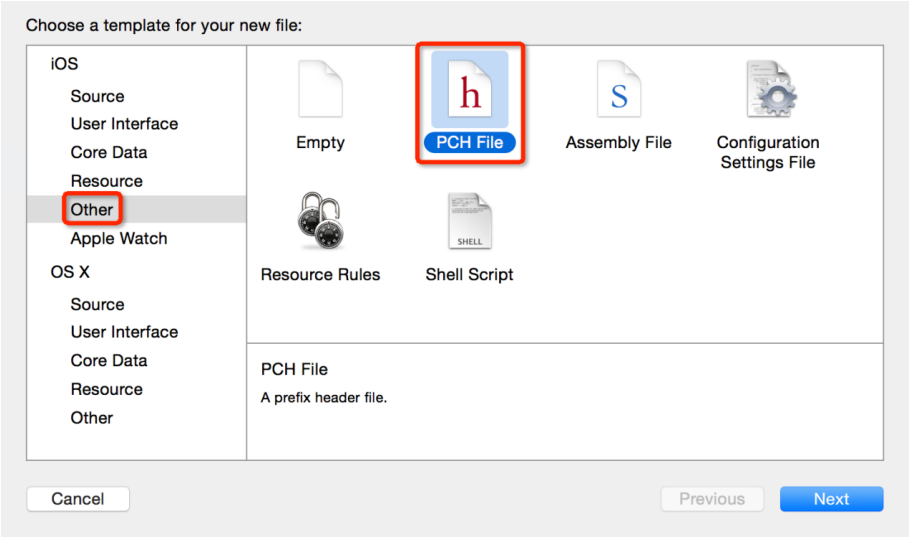
4. Swift 工程请在 AppDelegate.swift 文件的 application:didFinishLaunchingWithOptions 函数中添加以下代码：

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

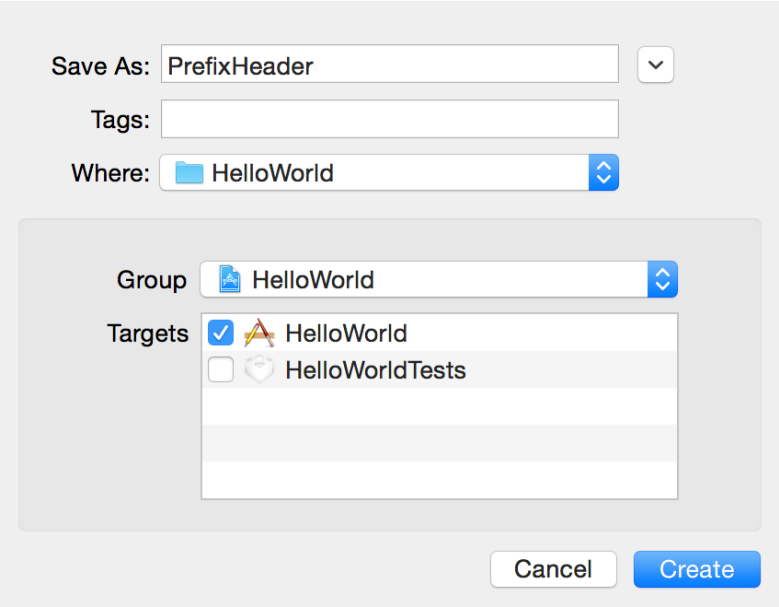
3 创建.pch 文件

3.1 新建 pch 文件

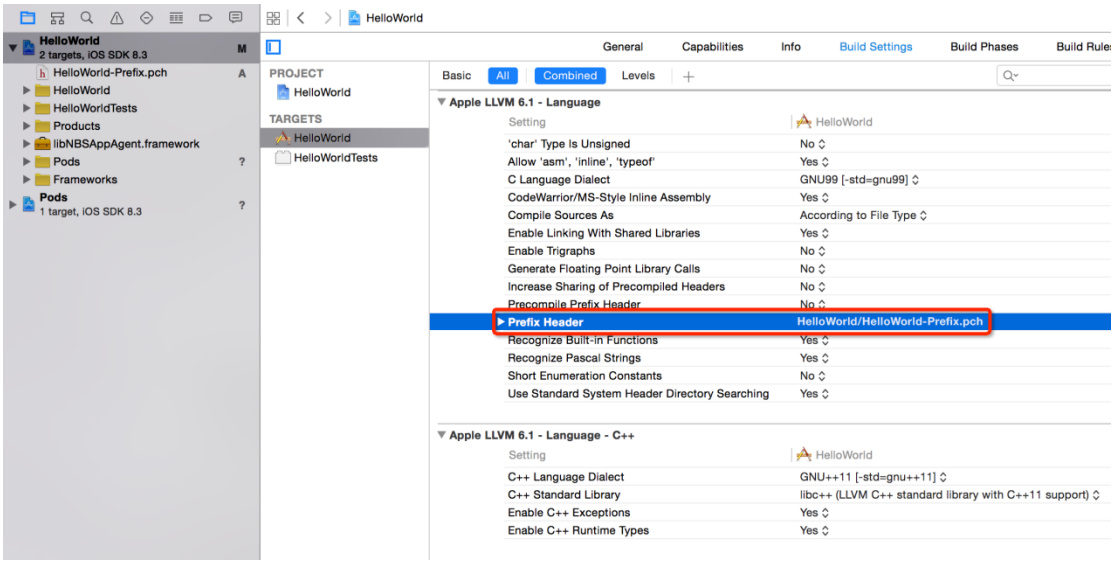
1. 打开 Xcode，依次选择 File→New→File（或者⌘N）。



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 traceEvent:@”EVENT_ID”];
```

3. 代码示例

```
-(void)purchaseFailedCallback
{
[NBSAppAgent traceEvent:@“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 MainActivity 的 onCreate 里用初始化方法传入 UserID。

2. 相关接口

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

3. [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
 }
```