

百度

百度移动统计 SDK

开发者手册 IOS-4.1 版本

MTJ

2016/6/20

修改记录

版本号	变更内容
4.1.0	(1)BUG FIX
4.0	(1)启用 HTTPS 日志传输
3.9	(1)IPV6 ONLY 适配 (2)日志中添加细分的网络类型
3.8	(1)BUG FIX (2)加强设备标识准确性。
3.7	(1)加强日志传输安全性。 (2)BUG FIX (3)该版本以后需要引用系统库 JavaScriptCore.framework
3.6	(1)新增自动监控模式。 (2)修复部分 BUG。 (3)支持收集完整详细的 CRASH 日志。
3.5	(1)适配 IWATCH 应用统计。 (2)存储效率提升，性能大幅优化。 (3)修复部分类型 CRASH 无法收集 BUG 。 (4)DEBUG 模式下 LOG 信息优化。
3.4	(1)去除了不支持 ARM64 的包，只留下支持 ARM64 的一个包。 (2)去除对 TOUCHJSON 的引用，使用 iOS 自带的 JSON 解析函数。 (3)支持版本改为 iOS 5.0+。 (4)新增对内嵌 WEBVIEW 页面的统计。 (5)DEMO 程序中新增 WEBVIEW 统计相关文件。 (6)修正部分 BUG。
3.3	(1)修正 DEMO 因静态库引用路径造成的编译错误问题 (2)修正部分注释文字 (3)错误日志中添加版本信息 (4)新增支持 ARM64- 64 位编译的库
3.22	(1)去除广告标识符的引用 (2)开发者可以根据需求来设置广告标识符
3.2	(1)优化错误信息采集部分的代码，采集错误更细化 (2)增加调试功能接口 (3)修复单页面启动时页面丢失的问题 (4)增加广告的依赖库
3.1	(1)增加统计事件发生时间 (2) 增加自动生成序列 CUID 自定义设备标示符 (3) 增加统一的 SDK 库，模拟器和真实设备共用一个库 (4) 优化升级本地缓存自定义事件部分日志的合并 (5) 增加 SHORTVERSION 接口，设置应用版本号接口 (6) 优化启动时发送策略，使发送更及时
3.0	(1) 增加统计事件发生时长

	(2) 设置应用进入后台再回到前台为同一次 SESSION 的间隔时间的方法
--	--

目录

前言.....	4
1. 关于调试(注意事项).....	4
第一章 简介.....	5
第二章 阅读对象.....	5
第三章 版本支持.....	5
第四章 集成使用.....	5
第一节 添加 SDK 到项目.....	5
第二节 参数申请.....	6
第五章 代码集成.....	6
第一节 启动功能.....	6
第二节 自定义事件统计.....	8
第三节 页面统计.....	9
第四节 渠道统计.....	10
第五节 应用版本统计.....	10
第六节 日志发送策略.....	11
第七节 调试模式.....	11
第八节 广告标识符设置.....	11
第九节 内嵌 WebView 页面统计.....	12
第十节 自动监控模式 (3.6 新增)	12
第十一节 错误日志解析.....	14
第六章 包的目录结构.....	14
第七章 FAQ.....	15
第八章 联系我们.....	15

前言

1. 关于调试(注意事项)

- (1) **保证客户端时间的准确性。**因为我们 `mtj.baidu.com` 网站上面的统计数据是根据客户端时间来统计的，所以如果您需要测试统计数据，那么请保证您的设备的时间是正确的，如果不正确，有可能会造成统计数据丢失或者延迟展现。
- (2) **保证自定义事件的 ID 已经注册。**如果您需要使用自定义事件，那么需要在 `mtj.baidu.com` 网站上自定义事件中定义您需要的自定义事件的 ID，然后按照自定义事件的规范写入代码。
- (3) **保证您已经替换了 AppKey。**Appkey 需要在 `mtj.baidu.com` 网站上注册应用，那么就会生成一个 Appkey，该 key 是该应用的唯一标识。如果没有该 key，那么，统计数据将不会展现。
- (4) 如果不知道怎么接入代码，请参考 `mtj.baidu.com` 网站上下载的 SDK 包中的 demo 实例程序。
- (5) **如果没有数据怎么办？**请确认以上五点都已经确认，网站统计数据会在 15 分钟左右展示出来，如果没有数据有可能是您的时区设置或者时间出现暂时异常，可以选择换台设备来测试，或者联系我们，具体参见第七章。
- (6) **关于渠道统计结果为 appstore，**说明您设置渠道在某种情况下丢失。请保证用户可能进入的页面已经设置了渠道，否则如果出现小型的崩溃或者内存不足时有可能导致起初设置的参数丢失。
- (7) **SDK 调试相关。**为了方便调试，从 3.2 版本起，新增了 `enableDebug` 接口来方便开发者通过 Log 来调试 SDK。
- (8) **怎样快速的发送统计日志。**可以设置 `session` 失效时间为 1S（默认为 30S，一般设置在 1 到 600S 之间）。这样每次启动间隔 1S 以上都会发送日志（前提是设置启动时发送而不是定时发送）。
- (9) **怎样查看发送日志。**打开调试开关，每次发送时在 `xcode` 工具中都会有 `sdk` 的 Log 发出。
- (10) **使用 Crash 版本的 SDK，日志怎么解析？**需要使用移动统计提供的 `Symbol` 工具，将 `dSYM` 文件信息提取后的数据包上传到 web 服务端，在服务端实时解析。
参见第五章第十一节。

第一章 简介

百度移动统计 SDK(iOS)是百度为 iOS 平台提供的统计工具(以下简称 SDK)。该文档提供了对如何使用 SDK 的一个详细说明。建议阅读时下载我们的 API 用例，按照用例设置自己的工程。SDK 下载地址：<https://mtj.baidu.com/web/sdk/index>，包的详细介绍见第六章。

如有其他问题可以参考网站的 FAQ，或者与我们联系:(apptongji@baidu.com)

第二章 阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员。

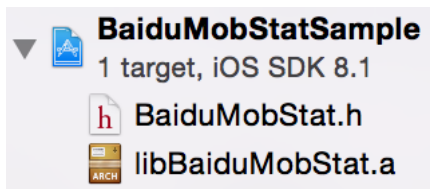
第三章 版本支持

iOS 5.0+

第四章 集成使用

第一节 添加 SDK 到项目












- 添加 sdk 静态库和.h头文件



- 添加系统依赖库

需要添加的系统依赖库如下：

▼ Linked Frameworks and Libraries

Name	Status
 JavaScriptCore.framework	Optional ⬆⬇⬆
 libstdc++.dylib	Required ⬇
 libz.1.2.5.dylib	Required ⬇
 Security.framework	Required ⬇
 CoreLocation.framework	Required ⬇
 SystemConfiguration.framework	Required ⬇
 CoreTelephony.framework	Required ⬇
 libBaiduMobStat.a	Required ⬇
 CoreGraphics.framework	Required ⬇
 UIKit.framework	Required ⬇
 Foundation.framework	Required ⬇

(注意：如果您的工程要兼容 iOS 7.0 以下的系统，将 JavaScriptCore.framework 的 status 状态设置为“Optional”)

第二节 参数申请

在百度移动统计平台(<https://mtj.baidu.com>) 申请应用 ID（APP KEY）用于标识您的应用程序。

在百度移动统计平台(<https://mtj.baidu.com>)的应用配置功能中创建 EventId 来定义您的自定义事件。

第五章 代码集成

第一节 启动功能

►启动代码

在应用启动函数(didFinishLaunchingWithOptions)中调用如下代码即可完成启动功能；启动的过程中同时您可以配置一些可选参数，具体有哪些参数，可详见 BaiduMobStat.h 文件。代码详见 Demo 程序。

（若应用程序是基于 iOS 9 系统开发，需要在应用的 info.plist 文件中添加如下配置，才能让日志正常发送，详情参考 Demo 程序的配置。如下：）

▼ NSAppTransportSecurity	Dictionary	(1 item)
NSAllowsArbitraryLoads	Boolean	YES

```
BaiduMobStat* statTracker = [BaiduMobStat defaultStat];
// 此处(startWithAppId之前)可以设置初始化的可选参数，具体有哪些参数，可详见BaiduMobStat.h文件，例如：
statTracker.shortAppVersion = [[[NSBundle mainBundle] infoDictionary]
    objectForKey:@"CFBundleShortVersionString"];

[statTracker startWithAppId:@"3af6b6bd84"]; // 设置您在mtj网站上添加的app的appkey,此处AppId即为应用的appKey
```

►初始化可选配置参数

如上图所示，在 `startWithAppId` 接口之前，可配置 SDK 初始化可选属性参数，参数列表如下，更多详情可见 `BaiduMobStat.h` 文件：

可选属性参数	说明
<code>shortAppVersion</code>	设置app的版本号 由于兼容历史Xcode3工程的原因，默认值 取 <code>CFBundleVersion</code> 中的版本号 若要统计与AppStore上一致的版本号（即 <code>CFBundleShortVersionString</code> 中的版本号），可自行获取后传入
<code>channelId</code>	设置渠道Id 默认值为 "AppStore"
<code>enableExceptionLog</code>	是否启用Crash日志收集 默认值 YES
<code>logSendWifiOnly</code>	是否仅在wifi网络状态下才发送日志 默认值 NO
<code>sessionResumeInterval</code>	设置应用进入后台再回到前台为同一次启动的最大间隔时间，有效值范围0~600s 例如设置值30s，则应用进入后台后，30s内唤醒为同一次启动 默认值 30s
<code>enableDebugOn</code>	设置是否打印SDK中的日志，用于调试 默认值 NO
<code>adid</code>	设置设备adid 若有需要，开发者可自行获取到adid后传入，使统计更精确 默认值 空字符串:@""
<code>monitorStrategy</code>	自动监控策略 详见第九节 默认值 不启动自动监控 <code>BaiduMobStatMonitorStrategyNone</code>

➤WatchKit 手表应用统计启动代码

对于手表应用 WatchKitApp 统计，需要在入口 `InterfaceController` 的

`-(void)awakeWithContext:(id)context;`

函数中，执行初始化代码块。初始化代码块与标准 App 统计的一致。

（若 watch 应用程序是基于 iOS 9 系统开发，需要在

`WatchKit Extension` 工程的 `info.plist` 文件中添加 `NSAppTransportSecurity` 相关配置，才能让日志正常发送，详情参考 Demo 程序的配置。如下：）

▼ NSAppTransportSecurity	Dictionary	(1 item)
NSAllowsArbitraryLoads	Boolean	YES

第二节 自定义事件统计

自定义事件分两种类型，无时长自定义事件、时长自定义事件。

自定义事件统计使用的 EventID 需要预先在 web 端配置，并且 eventLabel 不能为 nil 或者空字符串。

➤ 无时长事件统计

```
/**
 * 记录一次事件的点击，eventId请在网站上创建。未创建的evenId记录将无效。
 *
 * @param eventId 自定义事件Id，提前在网站端创建
 * @param eventLabel 自定义事件Label，附加参数，不能为空字符串
 */
- (void)logEvent:(NSString *)eventId eventLabel:(NSString *)eventLabel;
```

➤ 自定义事件时长

事件时长统计新增两种方法：

1. 在事件开始的时候调用 eventStart 方法，在事件结束时调用 eventEnd 方法。Sdk 自动计算事件发生时长。

```
/**
 * 记录一次事件的开始，eventId请在网站上创建。未创建的evenId记录将无效。
 *
 * @param eventId 自定义事件Id，提前在网站端创建
 * @param eventLabel 自定义事件Label，附加参数，不能为空字符串
 */
- (void)eventStart:(NSString *)eventId eventLabel:(NSString *)eventLabel;

/**
 * 记录一次事件的结束，eventId请在网站上创建。未创建的evenId记录将无效。
 *
 * @param eventId 自定义事件Id，提前在网站端创建
 * @param eventLabel 自定义事件Label，附加参数，不能为空字符串
 */
- (void)eventEnd:(NSString *)eventId eventLabel:(NSString *)eventLabel;
```

2. 调用 logEventWithDurationTime 方法，并将事件时长作为第三个参数 duration 传入。

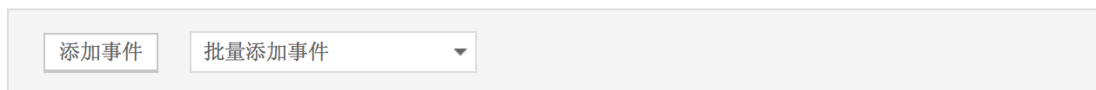
```
/**
 * 记录一次事件的时长，eventId请在网站上创建。未创建的evenId记录将无效。
 *
 * @param eventId 自定义事件Id，提前在网站端创建
 * @param eventLabel 自定义事件Label，附加参数，不能为空字符串
 * @param duration 已知的自定义事件时长，单位为毫秒 (ms)
 */
- (void)logEventWithDurationTime:(NSString *)eventId eventLabel:(NSString *)eventLabel durationTime:
(unsigned long)duration;
```


➤ 自定义事件如何设置与使用？

- 1.登录 mtj.baidu.com，点击应用进入查看报告。
- 2.左侧导航栏找到设置—自定义事件管理。



- 3.点击“添加事件”，输入“事件 ID”(与代码中的 ID 一致)，事件名称（事件的字面意义），点击“添加”，完成一个事件的添加。也可以按照模版批量上传。



事件Id	事件名称	操作
<input type="text"/>	<input type="text"/>	添加 取消

- 4.在代码中调用 event 相关 Api 时，将在 web 端添加的“事件 ID”当作 eventId 参数传入。

第三节 页面统计

统计 App 中的 ViewController 页面访问情况。

建议在各个 ViewController 中都添加 Api 调用统计，以增强对 App 的监控，同时提升对自定义事件、用户类型等数据的绑定准确度。

➤ 页面统计

页面统计两个接 Api 如下：

```
/**
 * 记录某个页面访问的开始，请参见Example程序，在合适的位置调用。
 * 建议在ViewController的viewDidAppear函数中调用
 *
 * @param name 页面名称
 */
- (void)pageviewStartWithName:(NSString *)name;

/**
 * 记录某个页面访问的结束，与pageviewStartWithName配对使用，请参见Example程序，在合适的位置调用。
 * 建议在ViewController的viewWillDisappear函数中调用
 *
 * @param name 页面名称
 */
- (void)pageviewEndWithName:(NSString *)name;
```

调用样例如下：

```

- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];
    [[BaiduMobStat defaultStat] pageviewStartWithName:@"pageName"];
}

- (void)viewDidDisappear:(BOOL)animated {
    [super viewDidDisappear:animated];
    [[BaiduMobStat defaultStat] pageviewEndWithName:@"pageName"];
}

```

➤ WatchKit应用页面统计

在自定义的WKInterfaceController的子类Controller的

- (void)willActivate函数中调用pageviewStartWithName

- (void)didDeactivate函数中调用pageviewEndWithName

如下：

```

- (void)willActivate {
    [super willActivate];
    [[BaiduMobStat defaultStat] pageviewStartWithName:@"WatchPage1"];
}

- (void)didDeactivate {
    [super didDeactivate];
    [[BaiduMobStat defaultStat] pageviewEndWithName:@"WatchPage1"];
}

```

第四节 渠道统计

渠道设置可参照第一节，可选属性参数配置中的 channelId。

您可以在startWithAppId 之前调用statTracker.channelId = @"你的渠道名”，设置不同的渠道名。

如果channelId 属性未设置，系统默认会采用AppStore 为您的应用渠道。

第五节 应用版本统计

版本设置可参照第一节，可选属性参数配置中的 shortAppVersion。

注意：由于兼容历史Xcode3工程的原因，默认值 取CFBundleVersion中的版本号

若要统计与AppStore上一致的版本号（即CFBundleShortVersionString中的版本号），可自行获取后传入

如Demo中的设置：

```

statTracker.shortAppVersion = [[[NSBundle mainBundle] infoDictionary]
    objectForKey:@"CFBundleShortVersionString"];

```

第六节 日志发送策略

日志发送策略有三种：

➤启动时发送（默认值，推荐使用）： 每次启动并联网时会把上一次启动期间的本地的日志发送。

BaiduMobStatLogStrategy = BaiduMobStatLogStrategyAppLaunch

➤每日发送： 设置每日单次发送后，启动时发送数据时会做一次检查，距离上次成功发送数据的时间间隔是否超过24 小时，如果超过24 小时，则将之前保存在本地的日志全部发送。如果还不到24 小时，则不发送。

BaiduMobStatLogStrategy = BaiduMobStatLogStrategyAppDay;

➤自定义发送间隔：

开发者可以自定义发送间隔（单位为小时，1-24 都可以设置），如果开发者设为N 小时，启动时发送数据时会做一次检查，距离上次成功发送数据的时间间隔是否超过 N 小时，如果超过 N 小时，则将之前保存在本地的日志全部发送。如果还不到 N 小时，则不发送。

**BaiduMobStatLogStrategy = BaiduMobStatLogStrategyCustom;
logSentInterval = N;**

➤仅在WIFI 发送

对上述所有发送间隔设置均有效

是：必须在wifi 联网方式下才能发送数据

否：不论联网方式都会发送数据（默认值）。

logSendWifiOnly = YES/NO;

第七节 调试模式

为了方便开发者进行调试，自 V3.2 版本起增加了 SDK 的调试接口，通过以下代码可以打开调试。运行时会有 SDK 的 Log 打印。

```
statTracker.enableDebugOn = YES;
```

关闭调试，可以去除该代码或者设置为 NO。

需要在startWithAppId 之前调用，可参照第一节，可选属性参数配置。

第八节 广告标识符设置

由于苹果禁用了没有使用广告业务的app使用广告标识符，所以V3.22版本的SDK去除了该广告标识符的使用，开发者如果使用到了广告，可以自己设置该字段，可以提高统计精度。获取 adid 需要引用系统依赖库 AdSupport.framework.

```

/*如果有需要，可自行传入adid
NSString *adId = @"";
if([[UIDevice currentDevice] systemVersion] floatValue] >= 6.0f){
    adId = [[ASIdentifierManager sharedManager] advertisingIdentifier] UUIDString];
}
}
statTracker.adid = adId;
*/

```

需要在 `startWithAppId` 之前调用，可参照第一节，可选属性参数配置。

第九节 内嵌 WebView 页面统计

如果您的页面中使用了 WebView 嵌入了 Html、JS 代码，并且希望统计 html 页面的页面访问与自定义事件，可以通过一下两个步骤实现：

1.实现 WebView 的代理方法，并再代理方法中调用 SDK 的

`webViewStartLoadWithRequest:`接口，传入 `request` 参数。

如下所示，详情见 Demo：

```

//实现WebView的代理方法，并在此函数中调用SDK的webViewStartLoadWithRequest:传入request参数，进行统计
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:(NSURLRequest *)request
    navigationType:(UIWebViewNavigationType)navigationType
{
    [[BaiduMobStat defaultStat] webViewStartLoadWithRequest:request];
    return YES;
}

```

2.在 WebView 的 Html、JS 代码中，添加对应的接口，并在合适的时候调用，从而使 SDK 获取到该行为，进一步调用 native 代码，实现统计。具体的 Html、JS 对应代码，详见 Demo 程序中的 `mobstat.html`、`mobstat.js` 两个文件。

第十节 自动监控模式（3.6 新增）

为避免用户繁琐的在程序中埋点，SDK 新增自动监控模式，可帮助用户自动监测部分行为，具体如下：

一．如何开启

SDK 默认状态下时不启动自动监控功能，可通过如下方法开启：

```
statTracker.monitorStrategy = BaiduMobStatMonitorStrategyAll;
```

在初始化模块中，`startWithAppId` 之前调用以上属性设置，可开启相应模式的监控。详见 `BaiduMobStat.h` 中的声明。

```

typedef enum _BaiduMobStatMonitorStrategy {
    BaiduMobStatMonitorStrategyNone = 0,        //不启用自动监控
    BaiduMobStatMonitorStrategyPageView = 1,    //只启动页面统计自动监控
    BaiduMobStatMonitorStrategyButton = 2,       //只启动 button 统计自动监控
    BaiduMobStatMonitorStrategyAll = 3,          //启动页面统计与 button 统计自
动监控
} BaiduMobStatMonitorStrategy;

```

二. 支持的行为类型

1. 页面统计

默认监控所有 `UIViewController`，取其 `title` 作为 `PageName` 进行统计。

（若用户在 `ViewController` 中重新声明实现了 `viewDidAppear` 与 `viewDidDisappear` 函数，则需要在这两个函数中调用相应的 `super` 的 `viewDidAppear` 与 `viewDidDisappear` 函数，自动监控方可生效。若没有重新定义这两个函数，则不需要任何操作）。

2. UIButton 点击统计（只支持无时长自定义事件）

默认监控所有 `UIButton` 的点击事件，取其 `TitleLabel` 的 `text` 作为 `EventID`（同样需要在 Web 端进行配置），默认 `Label` 值 `@“defaultLabel”`，进行统计。

三. 特殊处理

1. 在自动监控模式下，若不希望统计部分页面或 `UIButton` 事件

可以通过设置对应 `Controller` 或 `UIButton` 的属性 `baiduMobStatHandleRecord` 为 `YES` 即可。

```
viewController.baiduMobStatHandleRecord = YES;  
button.baiduMobStatHandleRecord = YES;
```

2. 在自动监控模式下，若不希望使用默认值进行统计

a) `ViewController` 可以通过设置属性 `titleForBaiduMobStat` 修改监控所用的 `Name`。

```
viewController.titleForBaiduMobStat = @"MyPageNameForBaiduMobStat";
```

该语句需要在 `ViewController` 的 `init` 相关方法中执行。

Ps: `UIViewController` 的 `init` 方法有三种：

`init`、`initWithNibName:bundle:`、`initWithCoder:`

根据不同初始化方式执行不同的方法，请在该 `Controller` 所执行的 `init` 方法中修改 `titleForBaiduMobStat` 属性（不知道具体哪一个的开发者可以都试一下，看执行的是哪一个）。

b) `UIButton` 可以通过初始化时设置属性修改统计使用的 `EventID` 与 `Label`。

```
button.titleForBaiduMobStat = @"EventID1";  
button.labelForBaiduMobStat = @"Label1";
```

四. 使用建议

1. 已经使用旧版 SDK 实现埋点统计的用户，可以不用开启监控模式。若要开启，需要删除之前的手动埋点代码，避免重复统计。

2. 由于 `UIButton` 的自动监控默认将按钮触发纪录为无时长自定义事件，若开发者要统计其为时长类自定义事件，则需要设置该 `UIButton` 的属性

`baiduMobStatHandleRecord` 为 YES，并与老版本一样进行手动调用 API 统计。

所以若有较多时长类自定义事件，或较多不使用默认值进行统计的 UIButton，建议不开启 Button 监控。

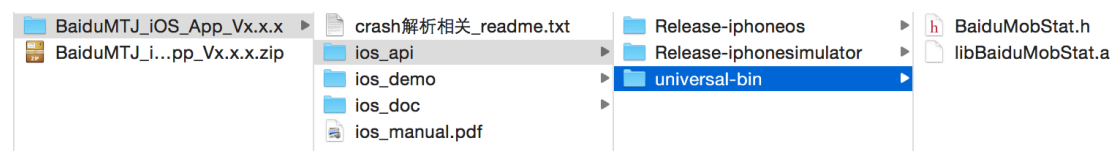
第十一节 错误日志解析

对收集详细错误 Crash 日志有需求的开发者，建议选择 Crash 版的移动统计 iOS SDK 集成（在 SDK 下载中心，选择“基础分析+Crash”）。同时需要使用移动统计提供的“Symbol 提取工具”，将 dSYM 文件中的信息提取，并将生成的数据 zip 包通过 web 网站端上传。详细步骤见“Symbol 提取工具”中的《百度移动统计 Symbol 工具使用手册》。

Symbol 提取工具，可以在登陆 mtj.baidu.com 后，进入应用，在左侧导航栏—设置—文件管理中下载。

第六章 包的目录结构

从 <https://mtj.baidu.com/web/sdk/index> 下载下来的 zip 包中解压会有以下目录结构：



- `ios_manual.pdf` 文件即为本文件，是详细的说明文档。
- `ios_api` 目录：此目录是要开发者包含到工程目录的文件，
 - a) 其中 `Release-iphoneos` 中的两个文件是给真机使用的，需要导入到工程中；
 - b) `Release-iphonesimulator` 目录是给模拟器使用的，需要导入到工程中；
 - c) `Universal-bin` 目录是前两个目录的.a文件的合体，如果开发者怕麻烦，可以直接导入此文件夹下的.a文件，这样就不用在模拟器和真机设备之间切换导入不同的.a文件了。
- `ios_demo` 目录是集成了统计里面的所有功能的一个实例，用户可以在其中参考使用统计的各种功能。
- `ios_doc` 目录是生成的类目录说明，用户可以直接查看类的用法
- `crash 解析相关_readme.txt`，是介绍 Symbol 工具的下载使用方法。

第七章 FAQ

1. 开通与设置百度移动统计步骤？

- 1) 在 mtj.baidu.com 注册并开通百度移动统计。
- 2) 登录后点击“新增应用”按钮，按步骤提示输入信息，创建应用。
- 3) 创建应用后会获得应用唯一的 AppKey (代码中初始化必传参数，后续可在进入查看应用详情页的左侧导航—设置—应用信息查看 AppKey)。
- 4) 下载 iOS SDK，按照文档导入头文件与库文件，并按照步骤集成代码。

2. 什么是渠道 ID？

- 1) 在集成代码初始化时设置的 channelId，即为渠道 ID。详见文档[第四节](#)。
此处 ChannelID 为用户自定义，无标准值，用户自己查看报告时能区别即可。
- 2) 用户在 web 端“设置—渠道管理”中，可以暂停某个渠道，或者重命名渠道 ID 展现的名称。

3. 什么是渠道来源细分？

渠道来源细分在渠道分布的基础上，帮助开发者更细粒度监测渠道质量。对于 Android 应用，可以细分统计到渠道中不同推广位置的下载激活效果；对于 iOS 应用，可以细分统计到 AppStore 上游渠道来源的下载激活效果。使用方法如下：

- 1) 请先集成新版 SDK 3.1 及以上版本
- 2) 点击【渠道来源设置】—【添加渠道来源】，对应用初始下载 URL（应用程序包的下载链接，例如：www.baidu.com/baidu.apk）进行封装。如果没有该功能请联系本应用的管理员账号获取
- 3) 将封装后的 URL 提供给渠道商，通知渠道商将初始 URL 替换为封装后的 URL 再进行推广，日志回传之后即可在【渠道来源细分】中查看相关报告了

注：由于用户从点击到激活不可避免的会产生数据丢失，存在一定误差，因此本功能不建议作为与渠道商结算的依据，主要帮助开发者衡量各渠道之间的推广效果

4. 为什么接入正常，网站却无数据？

- 1) 请按照文档的 [前言—关于调试](#) 进行排查。确保 AppKey 正确设置，网络状态正常等。
- 2) 初始化时将属性参数 enableDebugOn 设置为 YES，通过 Xcode 运行程序，在启动时，看 Xcode 中的 Log 信息中是否有移动统计的 Log 信息，进行进一步排查。
- 3) 行为数据上传到服务端到展现，有 5-10 分钟的延迟，请知晓。

5. 用户的去重标准是什么？

- 1) 百度移动统计以 OpenUDID 为基础，生成设备唯一标识 ID，用于标识用户。
- 2) 若用户的 App 应用中有集成广告功能，可获取到 IDFA（详见文档[第八节](#)），并在初始化 SDK 时传入，可提高用户识别准确度。

第八章 联系我们

感谢您的阅读,如果有问题请 email 我们。 邮箱: apptongji@baidu.com