

IOS 统计 SDK 开发者使用指南

腾讯移动分析出品

目录

IOS 统计 SDK 开发者使用指南	1
1 开始嵌入 SDK	3
安装和部署	3
升级 SDK	5
2 基础指标统计	6
页面统计	6
会话统计	7
错误统计	7
3 自定义事件	8
【次数统计】Key-Value 参数的事件	8
【次数统计】字符串参数的事件	9
【时长统计】的 Key-Value 参数的事件	9
【时长统计】字符串参数的事件	10
4 接口监控	11
5 网速监控	12
6 云标签	12
活动页面曝光事件	12
活动页面按钮点击事件	12

用户付费事件.....	13
7 高级功能	13
用户画像.....	13
游戏统计.....	13
用户反馈.....	14
8 数据上报	15
数据上报策略.....	15
数据上报相关的设置.....	16
9 APP 设置	18
10 更新用户配置参数.....	19

1 开始嵌入 SDK

安装和部署

欢迎使用腾讯移动分析 (简称 MTA) iOS 统计 SDK , 您可以按照下面 5 步完成 SDK 的安装和部署。

Step 1 获取 AppKey

登陆腾讯移动分析移动统计前台, 按照步骤提示注册应用, 可获得 AppKey。

腾讯内部用户: <http://mta.oa.com>

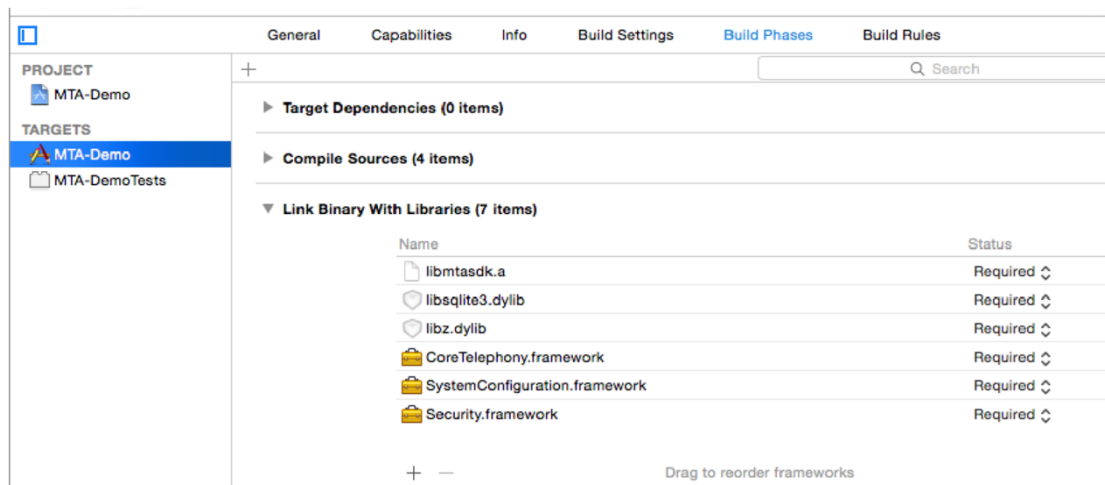
腾讯外部用户: <http://mta.qq.com>

Step 2 向工程中导入 SDK

下载统计 SDK 压缩包 解压至本地目录 将其中的 SDK 库 ,SDK 头文件导入到您的 XCode 应用工程中。

- SDK 库, SDK 头文件(MTA.h 和 MTAConfig.h)在解压开的 sdk 文件夹下
- Xcode 添加依赖系统库。依赖的系统库包括：
 - Libz.dylib
 - libsqlite3.dylib
 - SystemConfiguration.framework
 - CoreTelephony.framework
 - Security.framework

添加依赖库后 Xcode 工程中”Linked Frameworks and Libraries”中如下图：



iOS9 配置:

由于 iOS9 中的安全性做了增强, 需要修改工程的 plist 允许 http 连接.

打开工程的 plist 文件, 在 dict 字段增加以下内容

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
    <key>NSExceptionDomains</key>
</dict>
```

Step 3 在代码中添加 SDK 的引用

```
#import "MTA.h"
```

```
#import "MTAConfig.h"
```

MTAConfig.h: MTA配置相关接口, 需要在MTA.h接口前被调用才能及时生效

MTA.h: MTA统计功能相关接口, 需要开发者主动调用才能完成某项功能的统计。

在应用启动结束函数 `didFinishLaunchingWithOptions` 中调用 SDK 提供的启动方法

`[MTA startWithAppkey:@"myappkey"]`完成统计 SDK 启动。注意 `startWithAppkey`

的参数为申请的统计 ID 标识 AppKey, 如下图:

```
#import "AppDelegate.h"
#import "MTA.h"
#import "MTAConfig.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [MTA startWithAppkey:@"ios@testapp"];
    return YES;
}
```

注意: 其它SDK内置启动MTA情况下需要调用下面方法, 传入`MTA_SDK_VERSION`, 并检

查返回值。

```
+(BOOL) startWithAppkey:(NSString*) appkey checkedSdkVersion:(NSString*)ver
```

如下图:

```
if ([MTA startWithAppkey:@"DemoApp@MTA" checkedSdkVersion:MTA_SDK_VERSION]){
    NSLog(@"Start success!");
}
```

启动结束后, 应用可以开始调用SDK提供的其它统计方法。

Step 4 添加 SDK 的统计

在代码处调用类[MTA]提供的函数（见章节2、3、4、5），开始嵌入MTA的统计功能。

Step 5 验证数据上报是否正常

当您完成的MTA嵌入工作后，启动app，触发MTA统计接口，经过10秒左右，正常情况下，在您的app首页就能看到实时指标在更新，说明您已成功嵌入MTA，可继续深入的统计开发。

如果经过几分钟后，尚未看到实时指标更新，请检查以下事项：

- 1、设备是否正常联网；
- 2、APPKEY、渠道等设置是否正确；
- 3、确保已触发MTA统计接口；
- 4、打开MTA的debug开关，日志是否报错。

升级 SDK

新版本 SDK 兼容老版本接口，升级时只需要替换旧文件即可。

2 基础指标统计

基础指标包括页面统计，活动时长统计，会话统计，错误统计 4 个部分。

页面统计

使用类[MTA]提供的函数统计某个页面的访问情况：

- 标记一次页面访问的开始。

+(void) trackPageViewBegin:(NSString*) page;

参数：page 页面名

```
-(void) viewDidLoad:(BOOL)animated
{
    NSString* page = @"Page1";
    [MTA trackPageViewBegin:page];
}
```

- 标记一次页面访问的结束

+(void) trackPageViewEnd:(NSString*) page

参数：page 页面名

```
-(void) viewWillDisappear:(BOOL)animated
{
    NSString* page = @"Page1";
    [MTA trackPageViewEnd:page];
}
```

(注意：trackPageViewBegin 和 trackPageViewEnd 要成对匹配使用才能正常统计页面情况)

会话统计

以下 3 种情况下，会视为用户打开一次新的会话：

- 1) 应用第一次启动，或者应用进程在后台被杀掉之后启动
- 2) 应用退到后台超过 X 秒钟之后再次回到前台

X 秒通过 MTAConfig 类的属性 `sessionTimeoutSecs (int)` 函数设置，默认为 30s

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    .....
    [MTA startWithAppkey:@"DemoApp@MTA"];
    [[MTAConfig getInstance] setReportStrategy:INSTANT];
    [[MTAConfig getInstance] setSessionTimeoutSecs:60];
    return YES;
}
```

- 3) 调用 SDK 提供的 `startNewSession()` 函数

+(void) startNewSession

错误统计

收集应用程序的异常信息可以帮助您完善自己的程序，有下面两种方式上报异常信息。

- 上报错误和异常

+(void) trackError:(NSString*)error

参数：`error` 出错信息字符串

+(void) trackException:(NSEException*)exception

参数：`exception` 抛出的异常

```
@try{
} @catch (NSEException *e){
    NSLog(@"Caught % @ % @", [e name], [e reason]);
    [MTA trackException:e];
}
```

➤ 上报未捕获异常

SDK 默认捕获 app 未捕获的异常，如果需要关闭，可调用以下接口。

```
[[MTAConfig getInstance] setAutoExceptionCaught:FALSE];
```

3 自定义事件

可以统计某些用户自定义事件的发生次数，时间，变化趋势，例如广告点击，短信数量等等。

自定义事件分为 2 大类：

- 1、统计次数：统计指定行为被触发的次数
- 2、统计时长：统计指定行为消耗的时间，单位为秒。需要 begin 接口与 end 接口成对使用才生效。

其中每类事件都有 Key-Value 参数类型和不定长字符串参数类型，由于 Key-Value 参数类型的接口能表达更丰富的内容，我们**推荐优先使用 key-value 参数接口**。另外，如果代码同时使用了这 2 种参数类型，event_id 最好不一样。

注意：event_id 需要先在腾讯移动分析网站上面注册，才能参与正常的数据统计。event_id 不能包含空格或转义字符。

【次数统计】Key-Value 参数的事件

+(void) trackCustomKeyValueEvent:(NSString*)event_id

props:(NSDictionary *) kvs;

参数： event_id 事件标识

kvs 事件参数

```
-(IBAction) clickNormaltButton:(id)sender{
    NSDictionary* kvs=[NSDictionary dictionaryWithObject:@"Value" forKey:@"Key"];
    [MTA trackCustomKeyValueEvent:@"KVEvent" props:kvs];
    .....
}
```


【次数统计】字符串参数的事件

+(void) trackCustomEvent:(NSString*)event_id args:(NSArray*) array

参数： event_id 事件标识

args 事件参数

```
-(IBAction) clickNormaltButton:(id)sender{
    [MTA trackCustomEvent:@"NormalEvent" args:[NSArray
arrayWithObject:@"arg0"]];
}
```

【时长统计】的 Key-Value 参数的事件

可以指定事件的开始和结束时间，来上报一个带有统计时长的事件。

+(void) trackCustomKeyValueEventBegin:(NSString*)event_id

props:(NSDictionary *) kvs

+(void) trackCustomKeyValueEventEnd:(NSString*)event_id

props:(NSDictionary *) kvs

参数： event_id 事件标识

Kvs 事件参数

```
-(IBAction) clickStartButton:(id)sender{
    NSDictionary* kvs = [NSDictionary dictionaryWithObject:@"Value"
forKey:@"TimeKey"];
    [MTA trackCustomKeyValueEventBegin :@"KVEvent" props:kvs];.....
}
-(IBAction) clickEndButton:(id)sender{
    NSDictionary* kvs = [NSDictionary dictionaryWithObject:@"Value"
forKey:@"TimeKey"];
    [MTA trackCustomKeyValueEventEnd :@"KVEvent" props:kvs];
    .....
}
```

注意：trackCustomKeyValueEventBegin 和 trackCustomKeyValueEventEnd 必须成对出现，且参数列表完全相同，才能正常上报事件。

【时长统计】字符串参数的事件

可以指定事件的开始和结束时间，来上报一个带有统计时长的事件。

+(void) trackCustomEventBegin:(NSString*)event_id args:(NSArray*) array

+(void) trackCustomEventEnd:(NSString*)event_id args:(NSArray*) array

参数： `event_id` 事件标识

`args` 事件参数

```
-(IBAction) clickStartButton:(id)sender{
    [MTA trackCustomEventBegin:@"TimeEvent" args:nil];
    .....
}
-(IBAction) clickEndButton:(id)sender{
    [MTA trackCustomEventEnd:@"TimeEvent" args:nil];
    .....
}
```

4 接口监控

统计应用对某个外部接口(特别是网络类的接口,如连接、登陆、下载等)的调用情况。

当开发者用到某个外部接口,可调用该函数将一些指标进行上报,MTA 将统计出每个接口的调用情况,并在接口可用性发生变化时进行告警通知;对于调用量很大的接口,也可以采样上报,云监控统计将根据 sampling 参数在展现页面进行数量的还原。

+(void) reportAppMonitorStat: (MTAAppMonitorStat *)stat;

参数：stat 监控对象,需要根据接口情况设置接口名称、耗时、返回值类型、返回码、请求包大小、响应包大小和采样率等信息,详见 MTA.h

```
-(IBAction) clickNormaltButton:(id)sender{
    MTAAppMonitorStat* stat = [[[MTAAppMonitorStat alloc] init] autorelease];
    [stat setInterface:@"interface1"];

    // 被监控的接口

    ...
    [stat setRetsultType: SUCCESS];
    ...
    [MTA reportAppMonitorStat:stat];
}
```

5 网速监控

开发者在前台配置待监控的域名和端口列表,由服务器下发到 SDK,然后 SDK 在 app 启动时会主动测速,会对配置的所有域名进行测速监控。

(注意:本功能会产生网络 I/O)

6 云标签

云标签事件用于 mta 做数据挖掘并对挖掘后的用户贴标签,若需要使用 mta 云标签功能的应用需要上报此类事件,自定义事件 ID 和参数在配置时有特殊要求。

事件 ID	必填参数	说明
mta_tag_activity_open	aty、gid	活动页面曝光事件
mta_tag_activity_click	aty、btn、gid	活动页面曝光事件
mta_tag_user_pay	scene、amount	用户付费事件

活动页面曝光事件

活动页面曝光事件,事件 ID 为 "mta_tag_activity_open"。

```
NSMutableDictionary *dictionary = [NSMutableDictionary dictionaryWithCapacity:2];
[dictionary setObject:@"your_activity" forKey:@"aty"];
[dictionary setObject:@"1" forKey:@"gid"];
[MTA trackCustomKeyValueEvent:@"mta_tag_activity_open" props:dictionary];
[dictionary release];
```

活动页面按钮点击事件

活动页面曝光事件,事件 ID 为 "mta_tag_activity_click"。

```
NSMutableDictionary *dictionary = [NSMutableDictionary dictionaryWithCapacity:3];
[dictionary setObject:@"your_activity" forKey:@"aty"];
[dictionary setObject:@"OK" forKey:@"btn"];
[dictionary setObject:@"1" forKey:@"gid"];
[MTA trackCustomKeyValueEvent:@"mta_tag_activity_click" props:dictionary];
[dictionary release];
```

用户付费事件

用户付费事件，事件ID为 “mta_tag_user_pay”。

```
NSMutableDictionary *dictionary = [NSMutableDictionary dictionaryWithCapacity:2];  
[dictionary setObject:@"ipad4" forKey:@"target"];  
[dictionary setObject:@"350" forKey:@"amount"];  
[MTA trackCustomKeyValueEvent:@"mta_tag_user_pay" props:dictionary];  
[dictionary release];
```

7 高级功能

用户画像

用户画像统计需要app开发者主动上报QQ号码，若没有上报QQ号码，则无法使用用户画像及QQ登录数等特色功能。

+(void) reportQQ:(NSString*) qq;

参数： qq qq 号码

```
-(void) loginSuccess:(BOOL)animated  
{  
    NSString* qq = @"45284547";  
    [MTA reportQQ:qq];  
}
```

游戏统计

统计游戏用户需要调用下面接口方法上报游戏用户ID，分区，等级相关信息。若在App用户使用过程中，相关信息发生改变，需要重新调用接口上报。

+(void) trackGameUser:(NSString*)uid world:(NSString*)wd level:(NSString*)lev;

参数： uid 游戏用户 ID

world 游戏用户分区

level 游戏用户等级

```
-(void) loginSuccess:(BOOL)animated
{
    [MTA trackGameUser:@"g123" world:@"sz1" level:@"10"];
}
```

用户反馈

发送用户反馈

```
+(void) postFeedbackFiles:(NSString*)strContent screenshot:(UIImage*)screenshot callback:(void(^)(BOOL bSuccess, NSString* msg))cb;
```

参数： *strContent* 反馈内容

screenshot 屏幕截图

cb 回调

回调函数中 *bSuccess* 为 YES 表示操作成功, NO 为操作失败

msg 为服务器返回的相关信息

获取用户反馈

```
+(void) getFeedbackMessage:(uint32_t)offset numLine:(uint32_t)numLine callback:(void(^)(BOOL bSuccess, NSString* msg, NSArray<MTAFeedBack*>*datas))cb;
```

参数： *offset* 获取的偏移量, 0 表示从最新发送的那一条开始获取

numLine 获取条数

cb 回调

回调函数中 *bSuccess* 为 YES 表示操作成功, NO 为操作失败

msg 为服务器返回的相关信息

如果获取成功, *datas* 为获取到的用户反馈

回复用户反馈

```
+(void) replyFeedBackMessage:(NSNumber*) fbId content:(NSString*) content  
callback:(void(^)(BOOL bSuccess, NSString* msg)) cb;
```

参数：*fbId* 回复的反馈 id

content 回复的内容

cb 回调

回调函数中 bSuccess 为 YES 表示操作成功, NO 为操作失败

msg 为服务器返回的相关信息

8 数据上报

数据上报策略

设置数据上报策略,可以有效节省流量。使用以下 3 种方式调整 app 的数据上报策略:

- 1) app 启动时指定上报策略(默认为 MTA_STRATEGY_APP_LAUNCH)¹

```
@property MTAStatReportStrategy reportStrategy
```

腾讯移动分析目前支持的上报策略包括 6 种:

编号	策略名称	说明
1	MTA_STRATEGY_INSTANT	实时发送, app 每产生一条消息都会发送到服务器。
2	MTA_STRATEGY_ONLY_WIFI	只在 wifi 状态下发送, 非 wifi 情况缓存到本地。
3	MTA_STRATEGY_BATCH	批量发送, 默认当消息数量达到 30 条时发送一次。
4	MTA_STRATEGY_APP_LAUNCH	只在启动时发送, 本次产生的所有数据在下次启动时发送。
5	MTA_STRATEGY_DEVELOPER	开发者模式, 只在调用+(void) commitCachedStats:(int32_t) maxStatCount 时发送, 否

¹ property 定义在 MTAConfig.h, 调用方式为[[MTAConfig getInstance] setPropertyName:value]

		则缓存消息到本地。
6	MTA_STRATEGY_APP_LAUNCH + wifi MTA_STRATEGY_PERIOD	间隔一段时间发送，每隔一段时间一次性发送到服务器。

SDK 默认为 MTA_STRATEGY_APP_LAUNCH + wifi 下实时上报，对于响应要求比较高的应用，比如竞技类游戏，可关闭 wifi 实时上报，并选择 MTA_STRATEGY_APP_LAUNCH 或 MTA_STRATEGY_PERIOD 上报策略。

- 2) 考虑到 wifi 上报数据的代价比较小，为了更及时获得用户数据，SDK 默认在 WIFI 网络下实时发送数据。可以调用下面的接口禁用此功能(在 wifi 条件下仍使用原定策略)。

@property BOOL smartReporting

- 3) 通过在 Web 界面配置，开发者可以在线更新上报策略，替换 app 内原有的策略。app 下次启动时会自动生效并存储该策略。

上面 3 种方式的优先级依次递增。例如，wifi 下转为实时发送会优先于第 1 种方式中选定的任何策略执行；在 Web 界面上配置的策略会覆盖 app 本地已经生效的策略。

数据上报相关的设置

- 1) 设置最大缓存未发送消息个数（默认 1024）

@property uint32_t maxStoreEventCount

缓存消息的数量超过阈值时，最早的消息会被丢弃。

- 2) （仅在发送策略为 MTA_STRATEGY_BATCH 时有效）设置最大批量发送消息个数（默认 30）

@property uint32_t minBatchReportCount

- 3) （仅在发送策略为 PERIOD 时有效）设置间隔时间（默认为 24*60，即 1 天）

@property uint32_t sendPeriodMinutes

9 APP 设置

使用 MTAConfig 单例对象属性设置可以动态调整 APP 和 SDK 的相关设置，调用形式为：

```
[[MTAConfig getInstance] setPropertyName: value];
```

- 会话时长（默认 30s，离开应用 30 秒之后再回来，视为一次新的会话）

@property uint32_t sessionTimeoutSecs

- 消息失败重发次数（默认 3）

@property uint32_t maxSendRetryCount

- 用户自定义时间类型事件的最大并行数量（默认 1024）

@property uint32_t maxParallelTimingEvents

- 设置安装渠道（默认为 “appstore”）

@property (nonatomic, retain) NSString* channel

- 设置 app key

@property (nonatomic, retain) NSString* appkey

- 设置统计功能开关（默认为 true）

@property BOOL statEnable

如果为 false，则关闭统计功能，不会缓存或上报任何信息。

- 设置 session 内发送消息限制（默认为 0，即无限制）

@property int32_t maxSessionStatReportCount

如果为 0，则不限制 session 内发送消息的个数；若大于 0，每个 session 内发送的消息不会超过此值，若超过了，新产生的消息将会被丢弃。

10 更新用户配置参数

开发者在腾讯移动分析网站上设置 Key-Value 值之后，可以调用下面的接口动态获取线上最新的参数值。

何时更新本地参数：用户在前台配置在线参数，并不是实时下发的，而是当 SDK 上报会话统计日志时才会更新。调试时，可在配置参数 10 分钟后，让 app 退到后台超过 30 秒发生超时，产生一个会话，便会更新。

-(NSString*)getCustomProperty:(NSString*) key default:(NSString*) v