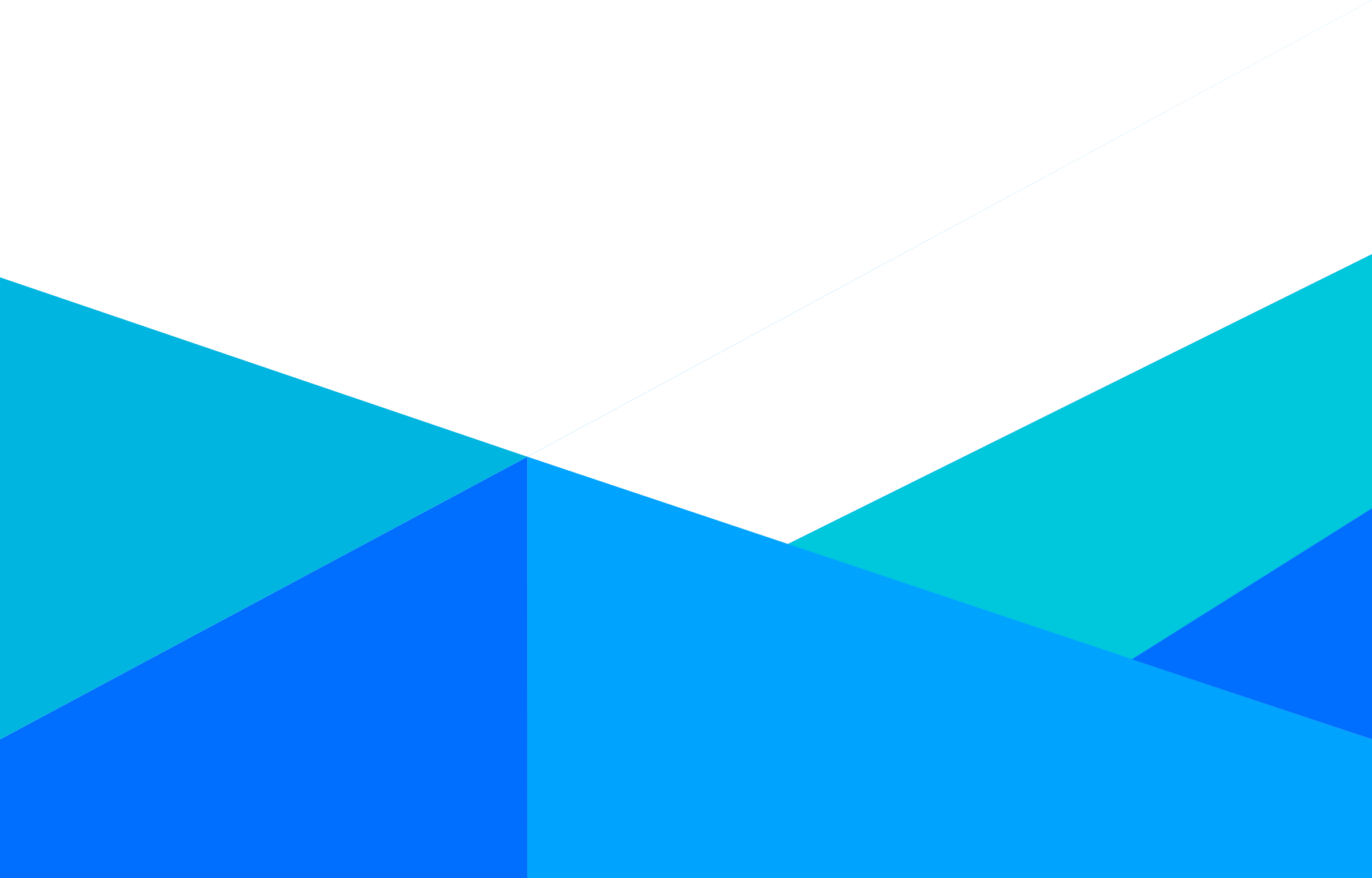
****

**腾讯云移动金融开发平台TMF**

**QAPM（iOS）用户手册**



腾讯云计算（北京）有限责任公司

**版权声明**

本文档著作权归腾讯云计算（北京）有限责任公司（以下简称“腾讯云”）单独所有，未经腾讯云事先书面许可，任何主体不得以任何方式或理由使用本文档，包括但不限于复制、修改、传播、公开、剽窃全部或部分本文档内容。

本文档及其所含内容均属腾讯云内部资料，并且仅供腾讯云指定的主体查看。如果您非经腾讯云授权而获得本文档的全部或部分内容，敬请予以删除，切勿以复制、披露、传播等任何方式使用本文档或其任何内容，亦请切勿依本文档或其任何内容而采取任何行动。

**商标声明**



“腾讯”、“腾讯云”及其它腾讯云服务相关的商标、标识等均为腾讯云及其关联公司各自所有。若本文档涉及第三方主体的商标，则应依法由其权利人所有。

**免责声明**

本文档旨在向客户介绍本文档撰写时，腾讯云相关产品、服务的当时的整体概况，部分产品或服务在后续可能因技术调整或项目设计等任何原因，导致其服务内容、标准等有所调整。因此，本文档仅供参考，腾讯云不对其准确性、适用性或完整性等做任何保证。您所购买、使用的腾讯云产品、服务的种类、内容、服务标准等，应以您和腾讯云之间签署的合同约定为准，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

**目 录**

[1 SDK 集成 5](#_Toc107598523)

[1.1 手动集成 5](#_Toc107598524)

[1.2 cocoaPods 集成 5](#_Toc107598525)

[2 初始化 SDK 及 Web 端环境配置 6](#_Toc107598526)

[3 功能配置 8](#_Toc107598527)

[3.1 卡顿检测功能 8](#_Toc107598528)

[3.2 Crash 监控功能 10](#_Toc107598529)

[3.3 Normal Crash 自定义上传业务日志功能 10](#_Toc107598530)

[3.4 启动耗时监控功能 12](#_Toc107598531)

[3.5 Web 监控功能 13](#_Toc107598532)

[4 用户隐私协议 14](#_Toc107598533)

[4.1 功能配置 14](#_Toc107598534)

[5 符号表配置 15](#_Toc107598535)

[6 查看 QAPM 工作日志 16](#_Toc107598536)

[6.1 设置查看工作日志 16](#_Toc107598537)

[6.2 上报日志分析 17](#_Toc107598538)

**前 言**

文档目的

本文档为QAPM（iOS）用户手册。

目标读者

本文档主要适用于如下对象群体：

* 开发工程师

阅读约定

本文档中可能采用的符号约定如下：

| 符号 | 说明 |
| --- | --- |
|  | 表示是正文的附加信息，是对正文的强调和补充。 |
|  | 表示有低度的潜在风险，主要是用户必读或较关键信息，若用户忽略注意消息，可能会因误操作而带来一定的不良后果或者无法成功操作。 |
|  | 表示有中度的潜在风险，例如用户应注意的高危操作，如果忽视这些信息，可能导致人身轻微或中度伤害、或设备损坏、数据丢失、设备性能降低以及不可预知的结果。 |
|  | 表示有高度的潜在危险，例如用户应注意的禁用操作，如果不能避免，可能导致人身严重伤害、死亡、或系统崩溃、数据丢失且无法修复等严重问题。 |

## 1 SDK 集成

iOS SDK 最低兼容系统版本 iOS 8.0。

### 1.1 手动集成

1. 下载 SDK。
2. 拖拽 QAPM.framework 文件到 Xcode 工程内（请勾选 Copy items if needed 选项）。
3. 在 **TARGETS** > **Build Phases-Link Binary Libraries** 添加依赖库。
   * libc++.dylib (libc++.tbd)
   * libc.dylib (libc.tbd)
   * CoreLocation.framework
4. 在工程的 Other Linker Flags 中添加-ObjC参数。
5. 将framework里面的js\_sdk.js文件导入到工程根目录。

fig:注意：如果需要使用证书校验，需要把上报移动监控环境的证书放在QAPM.framework > QAPM.bundle文件下（如果没有请新建），然后工程右键添加add Files(copy items if needed)将bundle证书添加到项目中。

### 1.2 cocoaPods 集成

暂时只支持本地的 cocoaPods 集成。

1. 将带有 QAPM 字样的整个文件夹，拖入到工程根目录带有 Pods 字样的文件夹下面。
2. 在工程根目录 Podfile（不是 Podfile.lock）文件中，加入如下内容（注意路径的正确性）。

pod 'QAPM', :path => './Pods/QAPM'

以上步骤完成之后，cd 到 Podfile 目录，然后执行pod install 指令即可。

fig:注意：如果需要使用证书校验，需要把上报移动监控环境的证书放在QAPM.framework > QAPM.bundle文件下(如果没有请新建)，然后执行pod install命令。

## 2 初始化 SDK 及 Web 端环境配置

进入 QAPM 页面的【配置】>【产品配置】，可以查看到 AppKey，该 key 在初始化接入中需要使用。  
图形用户界面, 应用程序, Teams

描述已自动生成

在工程的 AppDelegate.m 文件导入头文件

#import <QAPM/QAPM.h>

如果是 Swift 工程，请在对应 bridging-header.h 中导入。

初始化 QAPM 在工程AppDelegate.m 的application:didFinishLaunchingWithOptions:方法中初始化：

void loggerFunc(QAPMLoggerLevel level, const char\* log) {  
  
#ifdef RELEASE  
 if (level <= QAPMLogLevel\_Event) { ///外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
   
#ifdef GRAY  
 if (level <= QAPMLogLevel\_Info) { ///灰度和外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
   
#ifdef DEBUG  
 if (level <= QAPMLogLevel\_Debug) { ///内部版本、灰度和外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
}  
  
- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {  
  
 /// 设置QAPM 日志输出  
NSLog(@"qapm sdk version : %@", [QAPM sdkVersion]);   
[QAPM registerLogCallback:loggerFunc];  
   
 // 设置QAPM 开启的监控功能

[[QAPMModelStableConfig getInstance] setupModelStale:1];

//私有云用户根据实际情况配置。  
 [QAPMConfig getInstance].host =@"自定义host";  
 // appconfigHost内外网用户为默认值，私有云用户需要自定义设置。

[QAPMConfig getInstance].appconfigHost = @"私有云设置";  
 [QAPMConfig getInstance].customerAppVersion = @"设置app自定义版本号";

//根据实际情况设置userid 和deviceID

//设备唯一标识deviceID，例如IDFV配合Keychain使用

[QAPMConfig getInstance].userId = @"设置userId";  
 [QAPMConfig getInstance].deviceID = @"自定义deviceId";  
 /// 启动QAPM  
 [QAPM startWithAppKey:@"产品唯一的appKey"];  
 return YES;  
}

## 3 功能配置

### 3.1 卡顿检测功能

QAPMMoniterType: QAPMMoniterTypeBlue卡顿检测将在卡顿时，卡顿时间超过200ms阈值则采集堆栈进行上报。卡顿监控分滑动场景监控与非滑动场景监控。卡顿和掉帧监控默认1‰的抽样，接入时可以在产品设置—白名单中添加测试的userid

* **滑动场景的卡顿监控**

在相关页面进行如下代码的打点：

- (void)viewWillDisappear:(BOOL)animated {

[super viewWillDisappear:animated];  
   
 [QAPMBlueProfile stopTrackingWithStage:NSStringFromClass([self class])];  
}  
  
#pragma mark - TableView Delegate  
  
- (void)scrollViewWillBeginDragging:(UIScrollView \*)scrollView {  
 [QAPMBlueProfile beginTrackingWithStage:NSStringFromClass([self class])];  
}  
  
- (void)scrollViewDidEndDragging:(UIScrollView \*)scrollView willDecelerate:(BOOL)decelerate {  
 if(!decelerate){  
 [QAPMBlueProfile stopTrackingWithStage:NSStringFromClass([self class])];  
 }  
}  
  
- (void)scrollViewDidEndDecelerating:(UIScrollView \*)scrollView {  
 [QAPMBlueProfile stopTrackingWithStage:NSStringFromClass([self class])];  
}

滑动场景卡顿监控除了会上报该滑动场景的卡顿数据外，还会统计出该场景的一个流畅度指标。

#### 3.1.1 功能配置

@interface QAPMBlueProfile: NSObject   
/\*\*  
 开始记录掉帧，建议滑动开始时调用  
   
 \* @param stage 用来标识当前页面(一般为当前VC类名）  
 \*/  
+ (void) beginTrackingWithStage:(NSString \*)stage;  
  
/\*\*  
 掉帧，滑动结束时调用  
   
 \* @param stage 用来标识当前页面(一般为当前VC类名）  
 \*/  
+ (void)stopTrackingWithStage:(NSString \*)stage;  
  
  
@end

### 3.2 Crash 监控功能

QAPMMonitorTypeCrash Crash 日志会在下次启动 SDK 后上报数据。

fig:注意：业务方如果使用了第三方SDK收集普通崩溃的功能请卸载掉第三方监控软件，避免出现上报堆栈不准的问题。

在 FOOM 与deadlock卡死退出后，将在下次启动上报上一次记录的相关堆栈信息。FOOM在debug或者非 app store全量上报，app store环境下数据会有2%抽样

### 3.4 启动耗时监控功能

* **功能说明**
  + 使用启动耗时监控功能，可以统计出 App 进程创建时间到 App 第一帧UI上屏的时间。
  + 当启动时间超过阈值（默4000ms），则会上报个例详情。个例详情包括启动耗时、自动打点区间、自定义打点区间和启动过程堆栈。
* **相关接口**

**@interface** QAPMLaunchProfile : NSObject

/\*\*  
 设置自定义打点区间开始，该区间需要在启动时间区间内。begin与end的scene需要一致。  
   
 @param scene 场景名  
 \*/  
- (void)setBeginTimestampForScene:(NSString \*)scene;  
  
/\*\*  
 设置自定义打点区间结束，该区间需要在启动时间区间内。begin与end的scene需要一致。  
 @param scene 场景名  
 \*/  
- (void)setEndTimestampForScene:(NSString \*)scene;  
  
@end

* **代码示例**
  + 在 main 函数进行启动启动监控组件:

int main(int argc, char \* argv[]) {

@autoreleasepool {  
   
 [QAPMLaunchProfile didEnterMain];  
  
 return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));  
 }  
}

### 3.5 Web 监控功能

目前 WKWebView 支持 iOS >= 11 。

该功能能够监控 Web 网络资源加载耗时、jserror 监控。Web监控默认1‰的抽样，接入时可以在产品设置—白名单中添加设置的userid

#### 3.5.1 功能配置

* **Web 端配置**
* fig:注意：如果使用手动集成的方式，需要将 framework 里面的 js\_sdk 以 Add Files to 方式引入到工程中；如果是用的 cocoaPods 集成方式则不需要。
* 如果用到 TMFWebOffline 离线包功能，工程里面的 wkwebview 相关页面的头文件需要引入#import <TMFQWebView/QBWKWebView.h>，且遵循TMFWebOfflineWebViewControllerProtocol 代理，wkwebview 继承 TMFWkWebView 如下设置：

#import <TMFQWebView/QBWKWebView.h>

@interface WKWebviewViewController ()<WKUIDelegate, WKNavigationDelegate, WKScriptMessageHandler,TMFWebOfflineWebViewControllerProtocol>  
{  
  
 TMFWkWebView \*wkWebView;  
   
}

* **iOS SDK 配置**
  + 类文件中添加 #import <QAPM/QAPM.h>导入 SDK 头文件。
  + 在 WKWebView 的代理方法 webView:didCommitNavigation:中添加如下代码，以提供 Web 获取 native 相关信息接口。

- (void)webView:(WKWebView \*)webView didFinishNavigation:(WKNavigation \*)navigation

{  
   
[webView evaluateJavaScript:[QAPMWebViewProfile qapmBaseInfo:@" "] completionHandler:nil];  
  
 [webView evaluateJavaScript:[QAPMWebViewProfile qapmJsStart] completionHandler:nil];  
  
}

## 4 用户隐私协议

### 4.1 功能配置

隐私合规政策： 因隐私合规要求，在用户同意隐私合规之前请确保不调用QAPM的任何接口，此外QAPM仍然需要设备级的唯一标识用于确定设备的唯一性，用于用户指标级的计算。

// 当用户授权后，方可正常初始化QAPM

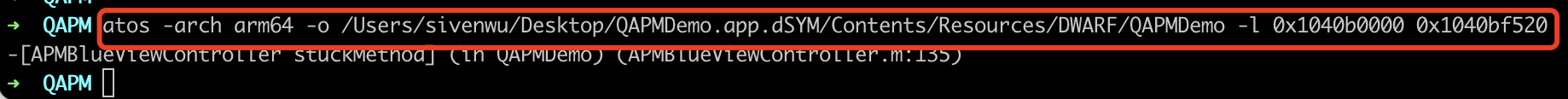
if (isAgree) { }  
//启动耗时函数的第一个打点  
// 需要传入设备的唯一标识，如IDFV配合Keychain使用   
[QAPMConfig getInstance].deviceID = @"自定义deviceId";   
//用户user ID、第三方登录账号，此接口可以多次在代码位置使用   
[QAPMConfig getInstance].userId = @"设置userId";

标识符deviceID采集方式变更背景： 当前监管要求SDK不允许直接或间接采集UDID等信息，我们只能通过让用户自行传入标识符的方法去区分不同的设备，有效的降低crash率指标数据的失真。

## 5 符号表配置

#### 5.1 上传符号表

利用已上报的数据个例上传符号表。根据个例页面的构建ID找到对应的符号表，可以使用官方atos命令翻译个例页面某行堆栈，确认符号表和翻译正常。



1. 在个例页面中有上传的入口，例如选择【卡慢】>【卡慢分析】>【卡慢问题列表】，进入详情页面。  
   图形用户界面, 文本, 应用程序, 电子邮件

   描述已自动生成
2. 在详情页面中，单击【上传dSYM文件】。  
   图形用户界面, 应用程序

   描述已自动生成
3. 进入符号表上传页面，单击【选择文件】，然后选择该构建对应的 dSYM 文件即可。  
   图形用户界面, 应用程序

   描述已自动生成

## 6 查看 QAPM 工作日志

### 6.1 设置查看工作日志

在调用[QAPM startWithAppKey:]启动 QAPM SDK 前，设置日志输出函数， 可以根据不同发布版本情况进行输出日志控制：

void loggerFunc(QAPMLoggerLevel level, const char\* log) {  
  
#ifdef RELEASE  
 if (level <= QAPMLogLevel\_Event) { ///外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
   
#ifdef GRAY  
 if (level <= QAPMLogLevel\_Info) { ///灰度和外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
   
#ifdef DEBUG  
 if (level <= QAPMLogLevel\_Debug) { ///内部版本、灰度和外发版本log  
 NSLog(@"%@", [NSString stringWithUTF8String:log]);  
 }  
#endif  
}  
  
  
- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {  
  
 /// 设置QAPM 日志输出  
 [QAPM registerLogCallback:loggerFunc];  
  
 /// ...   
 /// 设置启动QAPM SDK  
}

### 6.2 上报日志分析

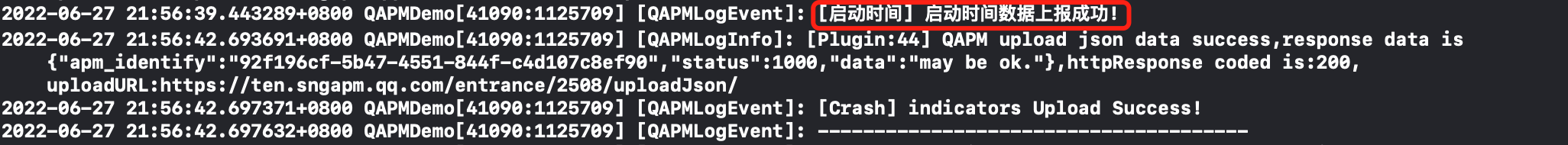
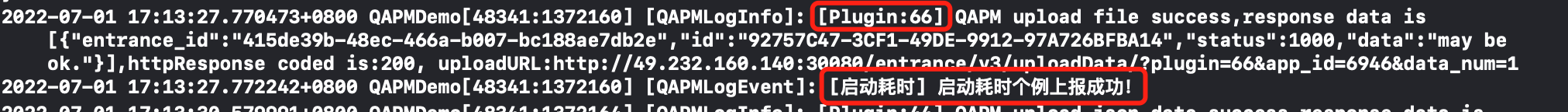
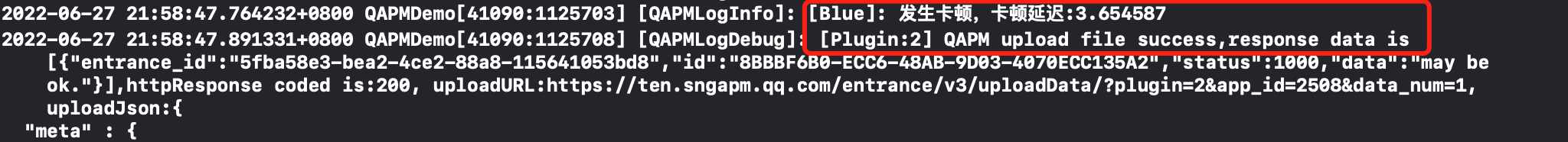
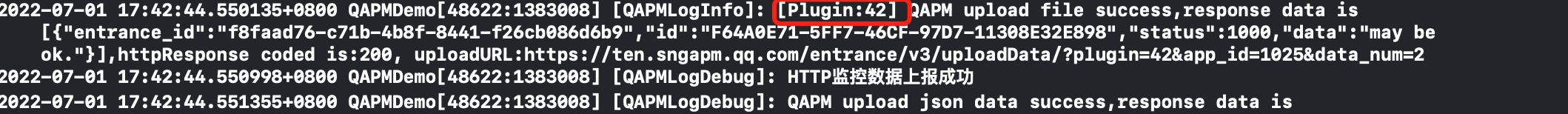
在接入完成 SDK 后，通常情况下会通过分析日志来确定监控功能是否已经开启。

监控功能未开启时，日志如下：  
手机屏幕截图

中度可信度描述已自动生成

监控功能开启时，日志如下：  
文本

描述已自动生成  
通过初始化日志，可以看到初始化成功，各个监控功能开启，然后就是各功能上报成功的验证。

* 启动耗时的上报  
  
* 
* 卡顿个例的上报  
  
* FOOM个例上报  
  
* Deadlock个例上报  
  
* HTTP 监控上报  
  
* 普通崩溃（normal crash）的上报  
  在触发 normal crash 的上报时，请不要将数据线连接 Xcode，触发完 normal crash 后，下次重启 App 的时候即可看到上报信息，该上报日志可通过 Mac 自带的控制台查看上报日志，日志如下:  
  fig:
* Webview 和 JSerror 的上报  
  Webview 和 jserror 的上报，可在xcode查看日志，以plugin:43和plugin:41为准。
* 