腾讯客户端性能分析

操作指南

腾讯云

文档目录

操作指南
Android SDK 接入

最近更新时间：2024-07-31 17:44:31

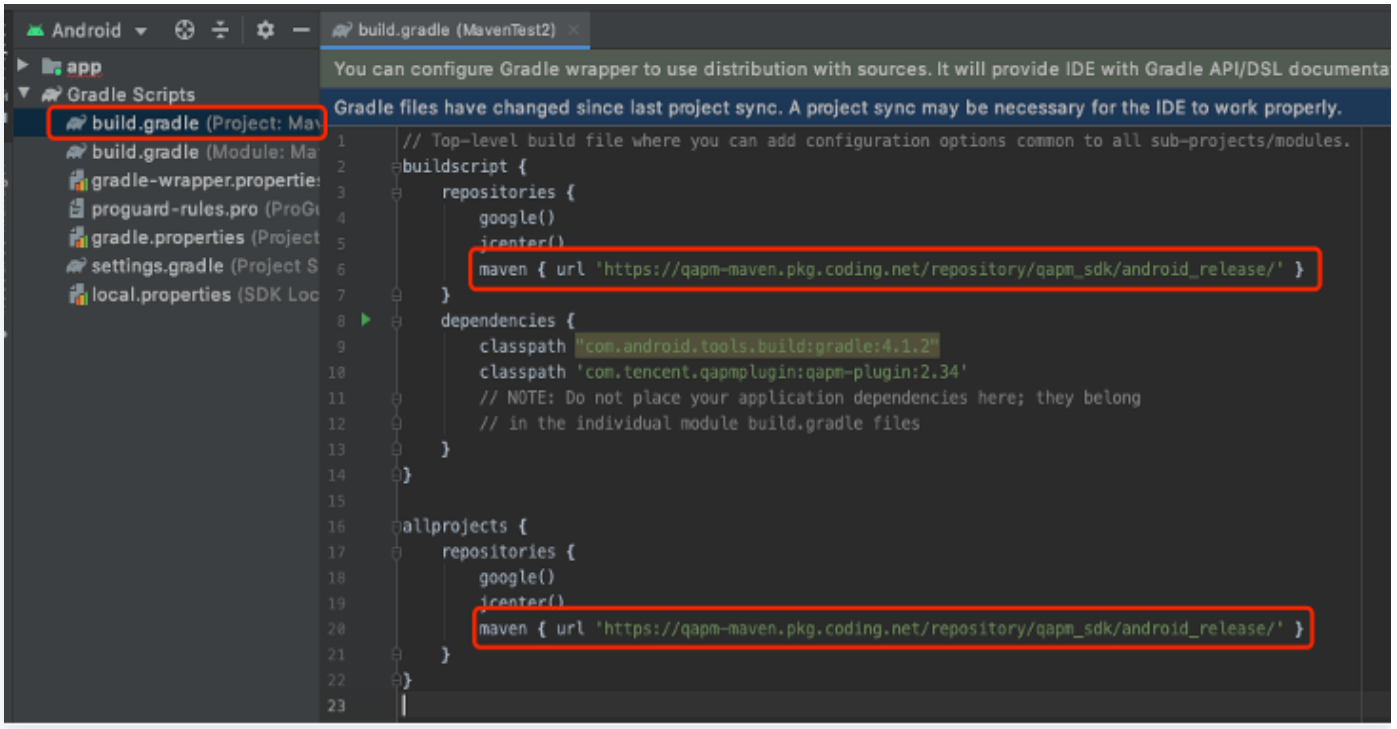
前言

1. 为确保隐私安全问题，请不要将内存功能开到线上，hprof 内包含整个 app 的信息，是个巨大的隐私安全隐患。
2. 隐私合规相关配置请重点关注 用户隐私协议 。

引入 SDK

Gradle 集成

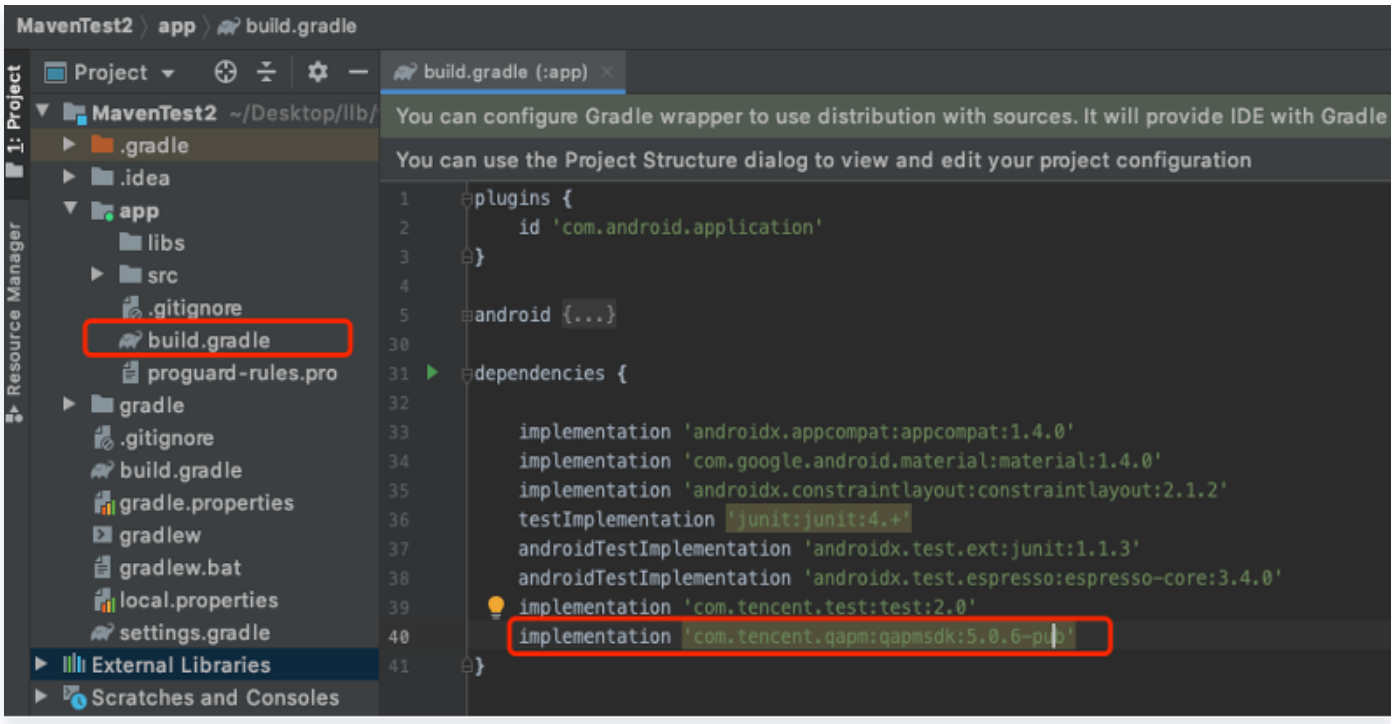
1. 在工程级的 build.gradle 中加入 maven 依赖。



参考代码：

```
  maven  {  url'https : / / qapm -
maven . pkg . coding . net / repository / qapm_sdk / android_release / ' }
```

2. 在 app 的 build.gradle 中引入模块。



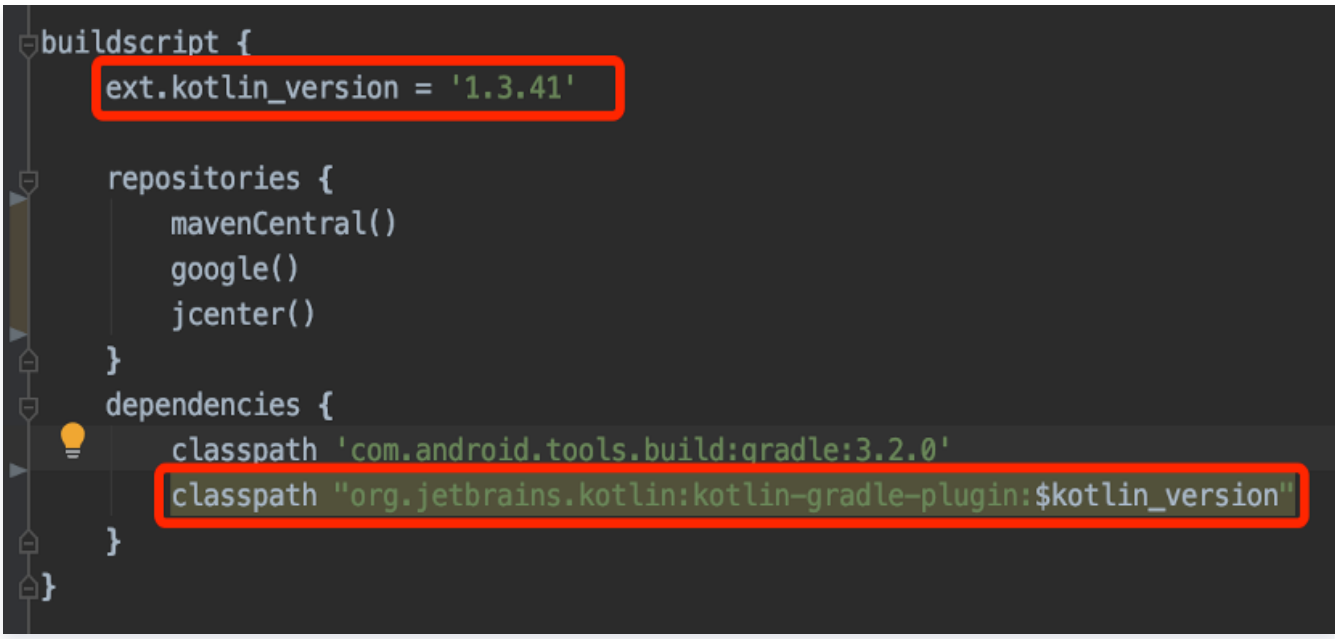
参考代码：

```
implementation'com . tencent . qapm : qapmsdk : 5.4 .3 - pub'
```

> ⚠ **注意：**
> studio 版本在3.0以下的请使用 compile 的引用头。

3. 因为 APM 工程使用了 kotlin 混合开发，需要依赖下 kotlin 插件。

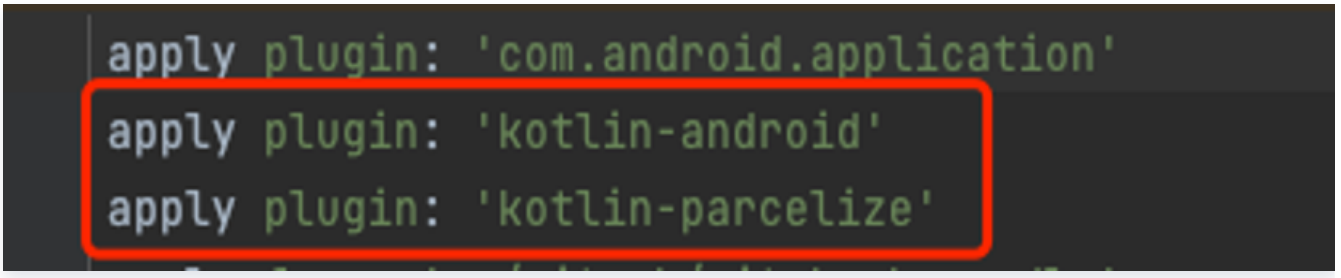
   3.1 在工程级的 build.gradle 下加入如下代码。



参考代码：

```
    ext . kotlin_version  =  '1.7.20'
    classpath  "org.jetbrains.kotlin:kotlin-gradle-
plugin:$kotlin_version"
```

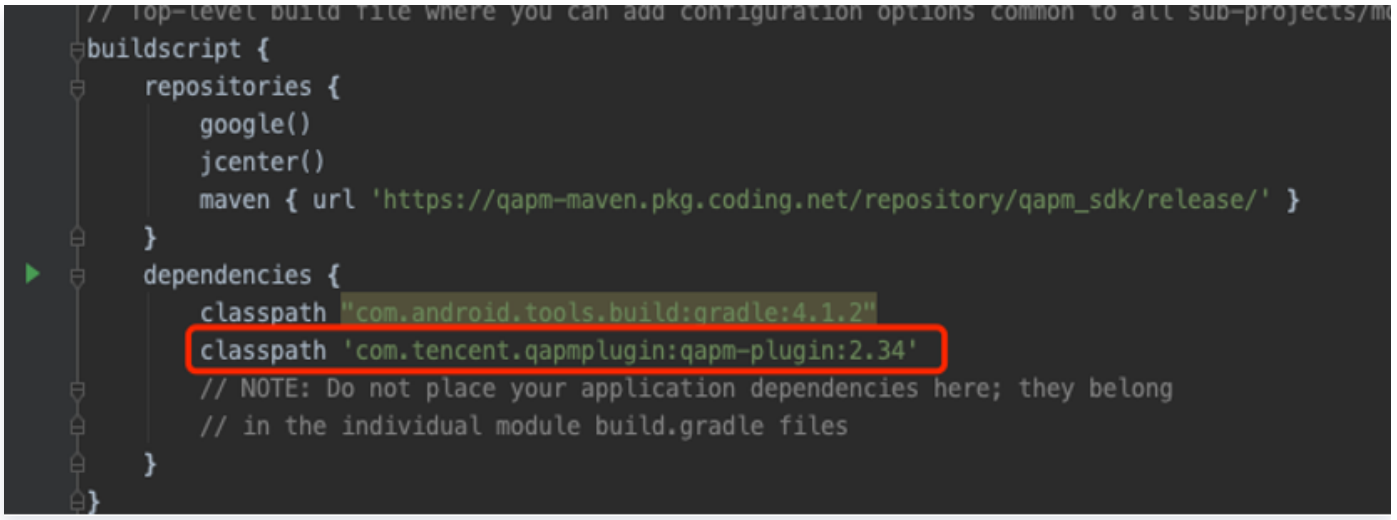
3.2 在 app 的 build.gradle 下加入如下代码。



参考代码：

```
    apply plugin :  'kotlin - android'
    apply plugin :  'kotlin - parcelize'
```

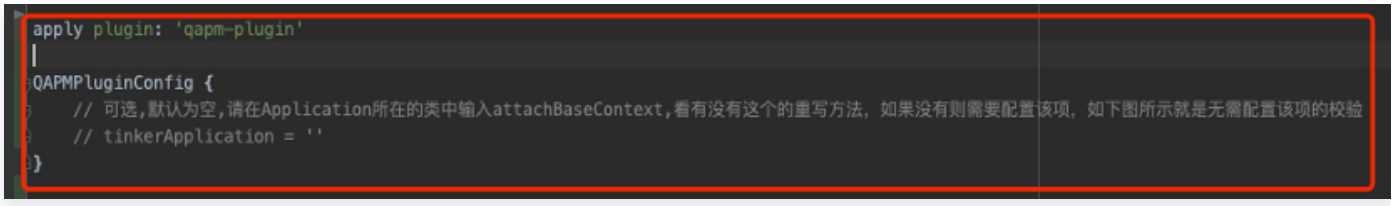
4. 在工程级的 build.gradle 文件中加入以下配置。



参考代码：

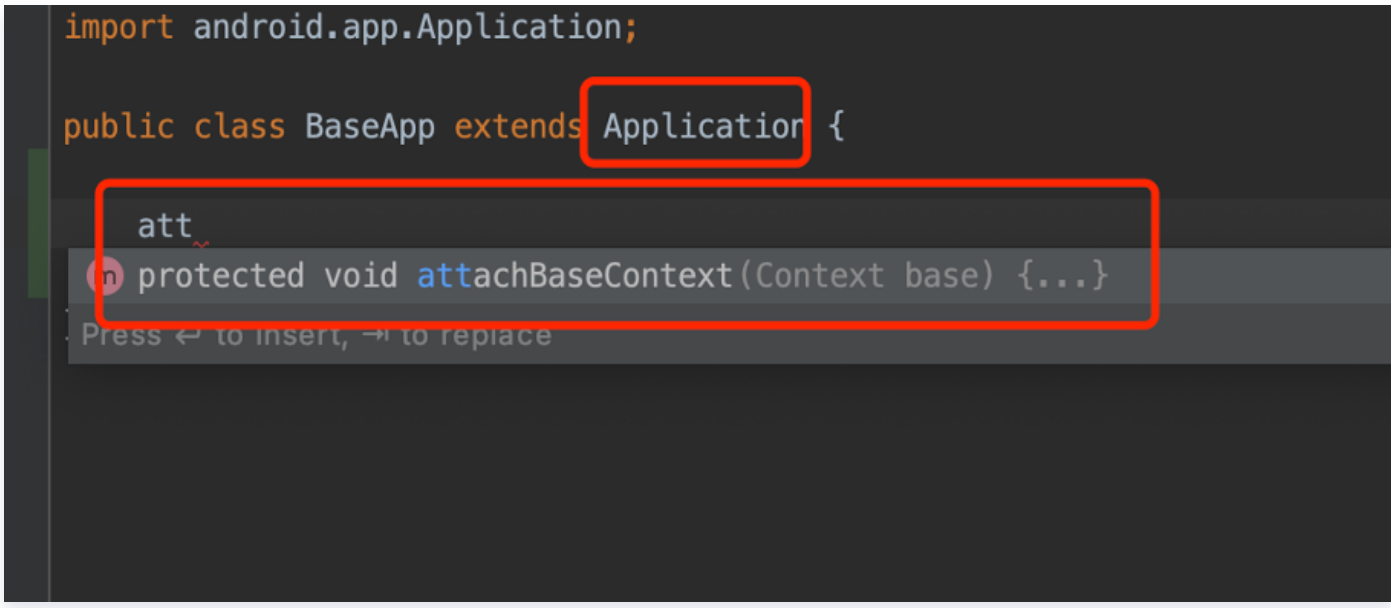
```
classpath 'com.tencent.qapmplugin:qapm-plugin:2.39'
```

5. **在 app 的 build.gradle 文件中加入以下配置。**



参考代码：

```
apply plugin: 'qapm-plugin'
QAPMPluginConfig {
    // 可选,默认为空,请在Application所在的类中输入attachBaseContext,看有没有这个的重写方法，如果没有则需要配置该项，如图所示就是无需配置该项的校验
    // tinkerApplication = 'com/tencent/qapm/demoApplication'}
```



> ⚠ **注意：**
> 如果未配置4 – 5项，则会影响启动、网络监控以及自动上传符号表功能的使用。

参数配置

1. 在 `AndroidManifiest.xml` 中添加以下权限。

```
<!-- 上报信息所需 -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- 采集信息所需 -->
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
```

2. 请避免混淆 qapm，在 app 的 `proguard-rules.pro` 文件中增加以下配置。

```
-keep class com.tencent.qapmsdk.**{*;}
# 如需要网络监控，请确保okhttp3不被混淆
-keep class okhttp3.**{*;}
```

启动SDK

SDK初始化

请拷贝下面代码（均是初始化必需的接口设置，详细的 SDK 功能接口配置请参考 监控功能启用 ），修改其中必需字段，建议在 Application 中初始化 QAPM：

```
// 为响应工信部“26号文”要求，提供该设置用于告知SDK是否可以进行可选个人信息的采集，默认可以采集。设置为false则不采集。该设置需要最先配置，一旦设置则全局生效
// 可选个人信息包括但不限于以下信息：设备制造商、系统、运营商、root状态等等，详见《QAPM SDK合规使用指南》
// 该设置置为false，部分信息将不再获取，可能会影响到控制台的搜索、展示等，请知悉！
QAPM.setProperty(QAPM.PropertyKeyCollectOptionalFields, true);
// 设置手机型号和设备ID，在隐私合规政策下，QAPM不可直接/间接获取mac地址、imei、androidid等隐私信息，但为了确保指标统计的准确性，需要用户参考4.8章节的建议，在统一获取相关信息后将信息按照既定方式处理，生成不敏感的QAPM设备唯一标识符并传参给SDK
// 需要传入设备的唯一标识（必需！！）
QAPM.setProperty(QAPM.PropertyKeyDeviceId, "设备的唯一标识");
// 需要传入手机型号（必需！！）
QAPM.setProperty(QAPM.PropertyKeyModel, "填写手机型号");
// 设置Application（必需）
QAPM.setProperty(QAPM.PropertyKeyAppInstance, getApplication());
// 设置AppKey（必需，用于区分上报的产品，该值由移动监控的产品配置页面获取 ）
QAPM.setProperty(QAPM.PropertyKeyAppId, "Your AppKey");
// 设置产品版本，用于后台检索字段（必需）
QAPM.setProperty(QAPM.PropertyKeyAppVersion, "Your App Version");
// 设置UUID，用于拉取被混淆堆栈的mapping （必需，若使用了QAPM符号表上传插件，可以直接使用该变量，否则请遵循UUID格式，自行传入该参数，请注意UUID与一次构建是一一对应关系，为了区分不同的构建版本，建议每次构建更新该参数），该变量会在编译前生成，爆红不用处理
QAPM.setProperty(QAPM.PropertyKeySymbolId, BuildConfig.QAPM_UUID);
// 设置用户ID，用于后台检索字段（必需）
QAPM.setProperty(QAPM.PropertyKeyUserId, "123456");
// 设置Log等级（可选），线上版本请设置成QAPM.LevelOff 或者 QAPM.LevelWarn
QAPM.setProperty(QAPM.PropertyKeyLogLevel, QAPM.LevelInfo);
```

⚠ 注意：

1. AppKey 请到移动监控的配置页 > 产品配置页获取。
2. 研发流程内，可以将设置的用户 ID /设备 ID 添加成白名单，确保指定设备的功能上报不会被采样影响，可在控制台中通过配置 > 产品配置 > 白名单配置 > 添加白名单完成操作。

监控功能启用

代码如下：

```
// 启动QAPM
QAPM . beginScene ( QAPM . SCENE_ALL ,  QAPM.ModeStable ) ;
```

接口说明如下：

```
public static boolean beginScene(String sceneName, int mode)
//用途：  开启监控
//参数：  sceneName  ——  场景名
//参数：  mode       ——  开启的功能
//可选项 ↓
QAPM.ModeStable              //--->开启稳定功能（包含卡顿、掉帧、用户行为、
Crash、ANR、启动，网络、WebView）
QAPM.ModeCrash               //--->开启Crash监控
QAPM.ModeANR                 //--->开启Anr监控
QAPM.ModeLaunch              //--->开启启动监控
QAPM.ModeLooper              //--->开启卡顿监控
QAPM.ModeDropFrame           //--->开启掉帧监控
QAPM.ModeHTTP                //--->开启原生层网络监控
QAPM.ModeBreadCrumb          //--->开启原生层用户行为监控
QAPM.ModeWebView             //--->开启WebView页面加载监控
QAPM.ModeJsError             //--->开启WebView的JS异常监控
QAPM.ModeHTTPInWeb           //--->开启WebView的网络监控
QAPM.ModeBreadCrumbInWeb     //--->开启WebView的用户行为监控
```

⚠ 注意：

- 使用或运算的方式自定义开启性能模块，如开启 Crash 和 Anr：`beginScene(“Crash&ANR”, QAPM.ModeCrash| QAPM.ModeANR)` 。
- 掉帧监控需先伴随其他功能启动，如 `beginScene(“Crash&DropFrame”, QAPM.ModeCrash| QAPM.ModeDropFrame)` ，之后参考 掉帧率采集 进行埋点统计。
- **为响应工信部 “26号文” 要求，我们依据工信部对性能监控类 SDK 基础功能的定义，将网络监控与用户行为监控划分为扩展业务功能，这意味着为了避免采集网络日志信息、用户操作记录等个人信息，您可以选择性开启这两个功能。操作层面您可以在自定义性能模块开启配置中避免填入 `QAPM.ModeHTTP` 、 `QAPM.ModeHTTPInWeb` 两个参数，以关闭网络监控功能；以此类推，您可以在自定义性能模块开启配置中避免填入 `QAPM.ModeBreadCrumb` 、 `QAPM.ModeBreadCrumbInWeb` 两个参数，以关闭用户行为监控功能。**

```
public static boolean endScene(String sceneName, long mode)
//用途：  结束监控（只针对掉帧采集有效）
```

```
//参数：  sceneName  ——  需要关掉的场景名（与beginScene的要相对应）
//可选项 ↓
QAPM.ModeDropFrame     //--->关闭掉帧监控
```

> ⚠ **注意：**
>
> - 多个进程需要各自初始化 QAPM。
> - Crash、Anr、启动、Jserror 功能采样率为100%，其余功能会默认采样，如需进行接入验证可以通过为验证设备添加白名单的方式规避采样影响，可在控制台中通过配置 > 产品配置 > 白名单配置 > 添加白名单完成操作。

如打印以下日志，则代表 SDK 初步接入成功，可以验证数据上报/尝试开启高级功能：
参考 TAG: QAPM_manager_QAPMPluginManager。

SDK 功能介绍

掉帧率采集

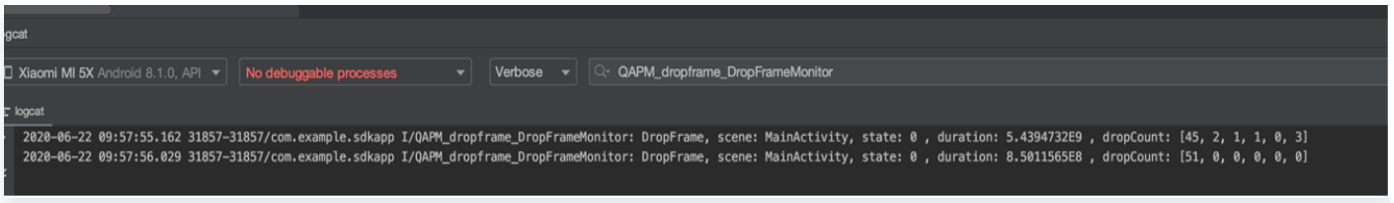
初始化需要开启掉帧监控，掉帧率的使用需要额外的埋点，建议打点在滑动列表上，如（ListView、GridView、ReclyclerView 等）。
在每次滑动前调用 `QAPM.beginScene(“xxxx滑动”, QAPM.ModeDropFrame);`
在滑动结束后调用 `QAPM.endScene(“xxxx滑动”, QAPM. ModeDropFrame);`
一般可以通过重写滑动组件的 `onScrollStateChanged` 方法来实现，示例如下：

```
@Override
public  void  onScrollStateChanged ( AbsListView  view ,  int
scrollState )  {
    if  ( scrollState  ==
AbsListView.OnScrollListener . SCROLL_STATE_IDLE ) {
        QAPM . endScene ( "xxxx滑动" ,  QAPM.ModeDropFrame ) ;
    }  else  {
        QAPM . beginScene ( "xxx滑动" ,  QAPM.ModeDropFrame ) ;
    }
}
@Override
public  void  onScrollStateChanged ( RecyclerView  view ,  int
scrollState )  {
    if  ( scrollState  ==  RecyclerView . SCROLL_STATE_IDLE )  {
        QAPM . endScene ( "xxxx滑动" ,  QAPM.ModeDropFrame ) ;
    }  else  {
        QAPM . beginScene ( "xxxx滑动" ,  QAPM.ModeDropFrame ) ;
```

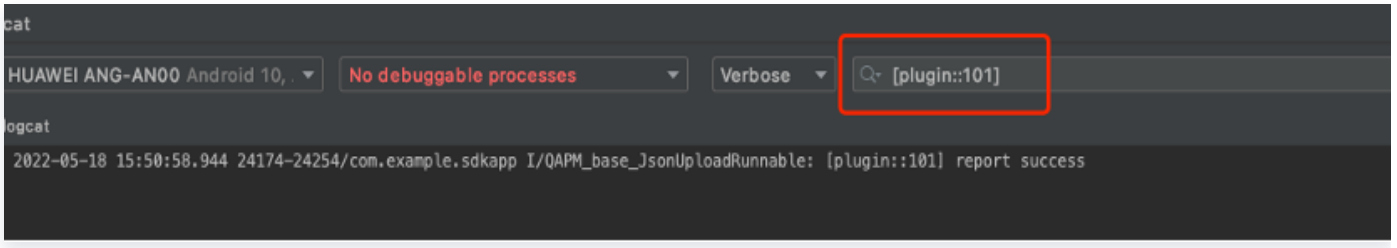
```
        }
    }
```

校验功能是否正常

- 滑动结束（调用 endScene）后,打印出以下日志则代表掉帧率数据已经存入了本地数据库：



- 检索 TAG: QAPM_dropframe_DropFrameMonitor。
- 在有埋点的情况下，启动五分钟之后，打印出以下数据，代表上报成功：



- 检索 TAG: [plugin::101]。

> ⚠ **注意：**
>
> - 不建议在进入 Activity 之后就开始，然后在退出 Activity 之后结束，掉帧是建立在页面渲染的情况下才会有数据的，如果进入的 Activity 是个静态的，则收集到的掉帧率数据无意义。
> - beginScene 与 endScene 需要成对出现，多次 beginScene 以第一次为准。
> - 掉帧数据每间隔5分钟上报一次。

WebView 性能、JsError 错误监控、Web 网络监控

初始化需要开启 WebView、JsError、Web 网络监控，如下是在 Stable 的基础上开启当前栏目的三个功能。

```
QAPM.beginScene(QAPM.SCENE_ALL,  mode: QAPM.ModeStable | QAPM.ModeWebView | QAPM.ModeJsError | QAPM.ModeHTTPInWeb);
```

除此之外，还需要配置以下代码，WebView 监控需要开启与JavaScript交互，在WebView初始化时调用如下代码开启：

```
WebSettings webSetting = webView.getSettings();
webSetting.setJavaScriptEnabled(true);
```

在 WebView 初始化完成之后加入 JAVA 与 JS 之间的调用接口通道，目的是让js层获取到 java 层的一些配置信息

```
WebView webView = new WebView(this);
webView.addJavascriptInterface(QAPMJavaScriptBridge.getInstance(), "QAPMAndroidJsBridge");
```

在 WebView 的 onPageFinished 代码里加入以下方法，用于注入 JS 脚本

```
webView.setWebViewClient(new WebViewClient() {
    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    @Override
    public void onPageFinished(WebView view) {
        super.onPageFinished(view, url);
        QAPMJavaScriptBridge.getInstance().initFileJS(view);
    }
});
```

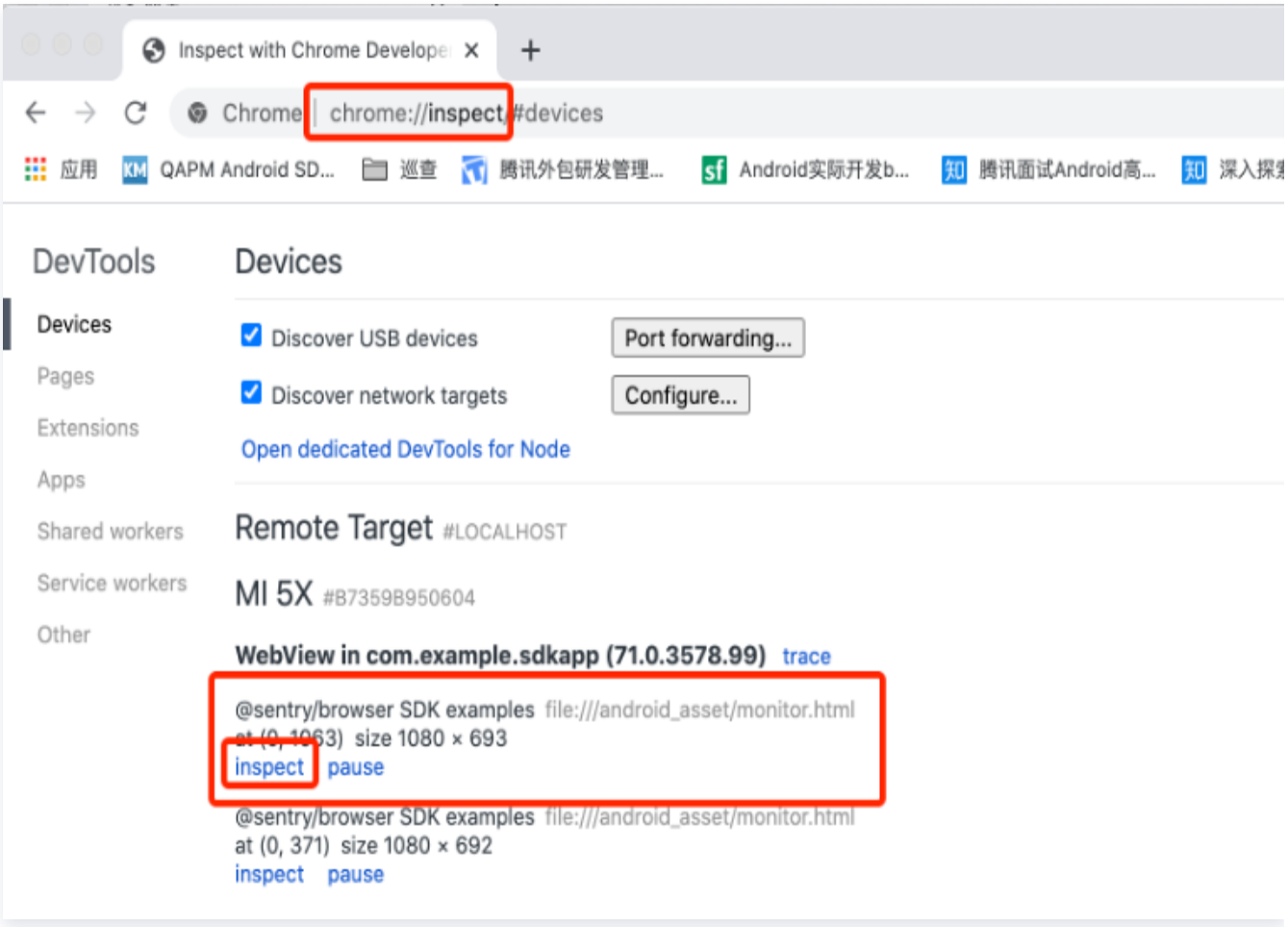
校验功能是否正常

- 原生 WebView、JsError 监控：

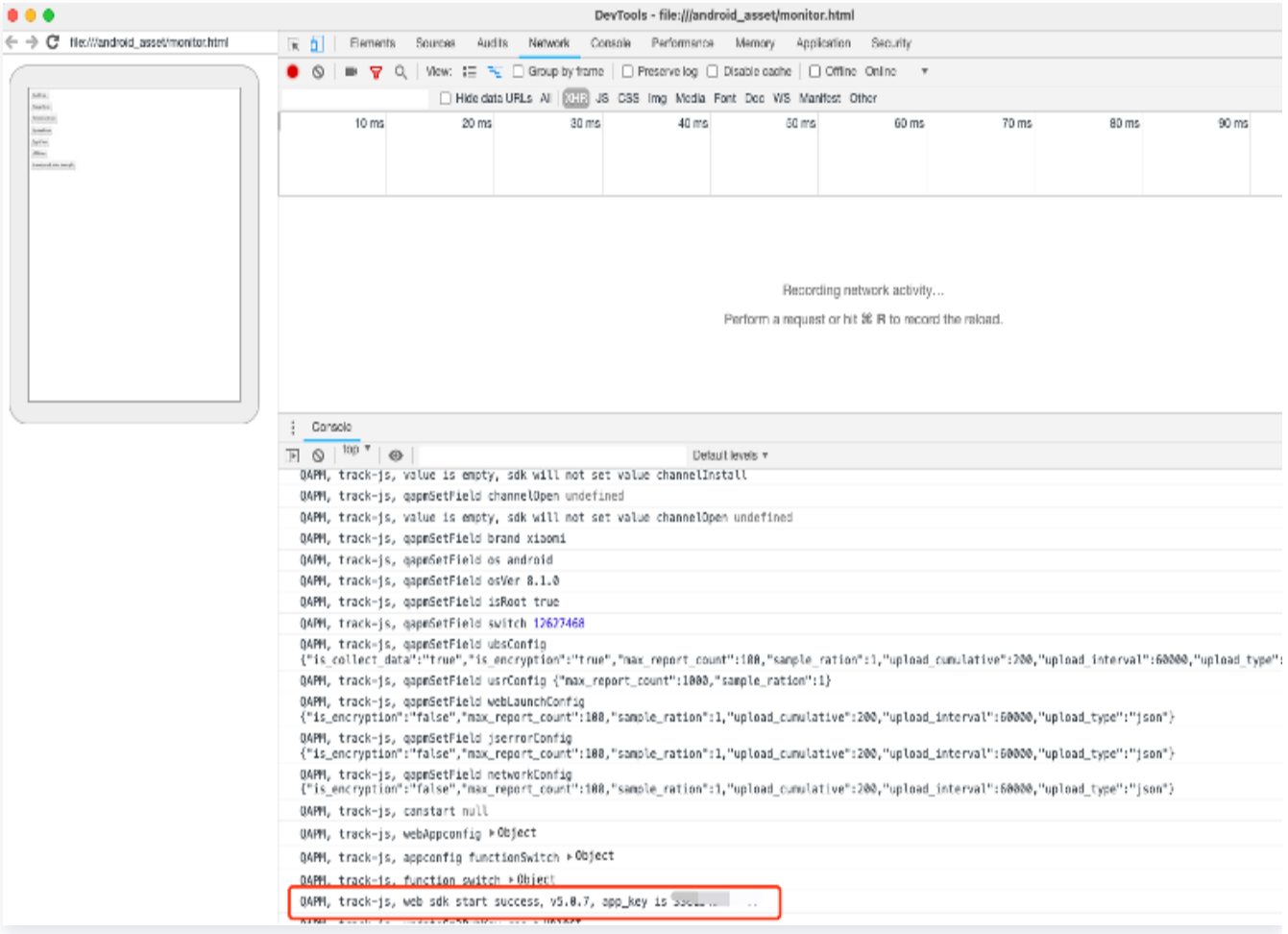
  1.1 代码中加入以下代码（用于远程调试）。

  ```
  WebView.setWebContentsDebuggingEnabled(true);
  ```

  1.2 打开谷歌浏览器，地址栏输入 `chrome://inspect` ，在出现的设备中点击 inspect。

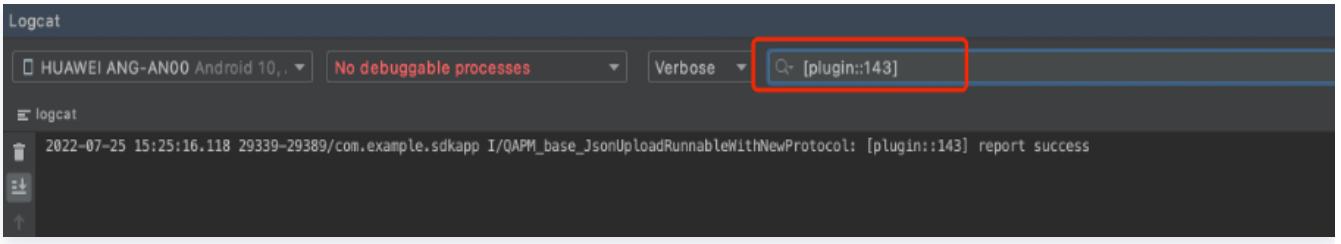


1.3 进入后找到 Console 模块查询日志，如出现 web start success ,vxxx，则代表 websdk 注入成功。



1.4 检测各个功能是否上报正常，以 jserror上报为例，如下：

每次触发 JsError 错误，如打印以下日志，则代表 JsError 数据上报成功。



- 检索 TAG: [plugin::143]。

其余检索 TAG 分别如下：

页面加载：plugin::141（每次 Web 页面加载完成后即会上报）。

网络请求：plugin::154（出现错误网络和慢请求时会上报）。

> ⚠ 注意：
>
> - 如需查看 WebView 监控是否正常需要通过 chrome 等浏览器调试查看。

- 只有页面加载时长大于3.5s的样本才可在问题列表里查看。
- 网络请求只有在网络错误和网络慢时才会上报。

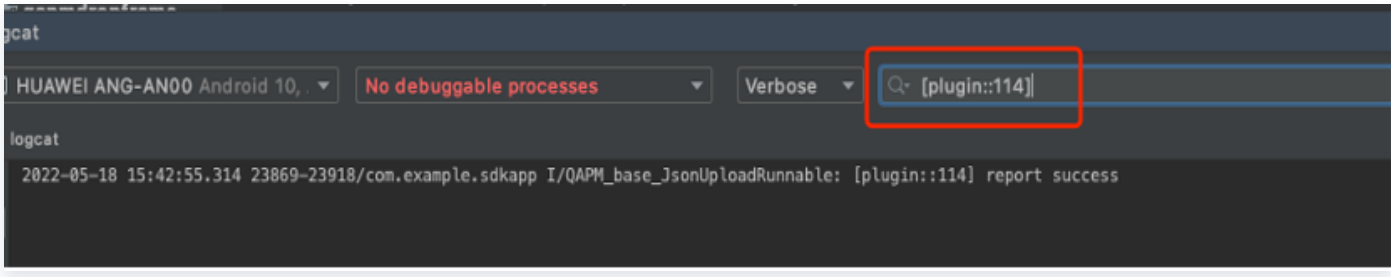
启动监控

启动监控需要使用 qapmplugin 插件在编译期间进行插桩操作，默认插桩点为 Application 与 Activity 的各个生命周期。
SDK 统计默认的启动耗时为 Application 的 attachBaseContext 到第一个 Activity 的 onResume 结束。
如果想自定义启动的结束点，则需要在第一个Activity调用onResume的20s内额外打点，示例如下：
参考代码

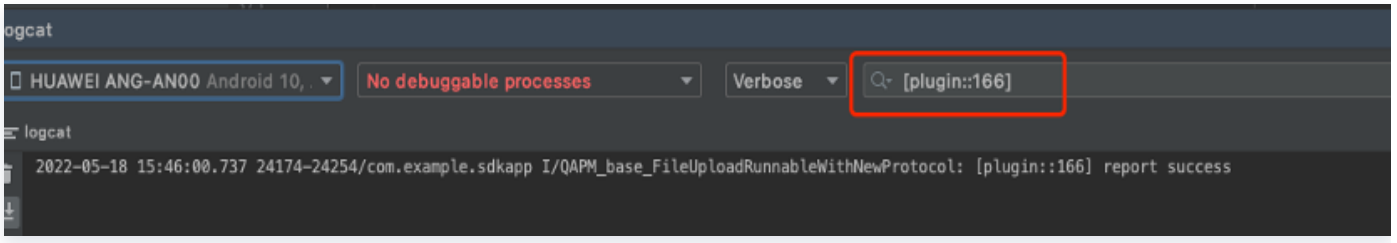
```
/**
 * 需要自定义结束点的用户需要在onResume之后的20s内，否则以APM的结束点为准
 */
QAPM . endScene ( StageConstant . QAPM_APPLAUNCH ,  QAPM.ModeResource ) ;
```

校验功能是否正常

- 每次启动后20s或者切换到后台，如打印以下日志，则代表启动指标数据上报成功。



- 检索 TAG: [plugin::114]。
- 每次启动后20s或者切换到后台，如打印以下日志，则代表启动个例数据上报成功。



- 检索 TAG: [plugin::166]。

⚠ **注意：**

- 需要使用 qapmplugin 插件进行插桩才可用，否则无效。
- 冷启动总耗时大于2.5s将会被记录为一个慢启动问题样本，而热启动的判断阈值则为1s。
- 慢启动的问题样本数据可以在移动监控的启动栏目 > 问题列表查看。

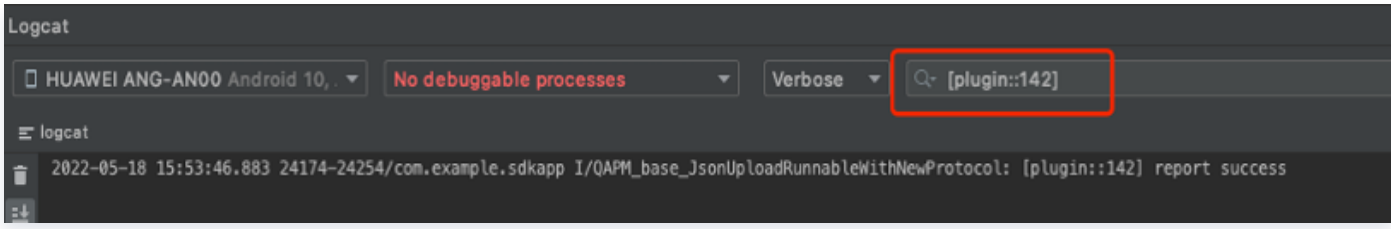
网络监控

网络监控需要使用 qapmplugin 插件进行插桩才可使用，默认会插在网络层的各个出入口。
这里可以根据使用网络通信模式选择相关的 http 库和版本，如下是 apm 在编译期间依赖的库和版本信息仅供参考。特别的如果有依赖 okhttp3 那么还必须依赖okio 1.14.0以上的库版本。

```
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okhttp3:okhttp-urlconnection:3.11.0'
implementation 'com.squareup.okhttp:okhttp:2.7.5'
implementation 'com.squareup.okhttp:okhttp-urlconnection:2.7.5'
implementation 'org.apache.httpcomponents:httpcore:4.4.11'
implementation 'org.apache.httpcomponents:httpclient:4.5.6'
```

校验功能是否正常

- 每次网络请求后1分钟，如打印以下日志，则代表网络数据上报成功。



- 检索TAG: [plugin::142]。

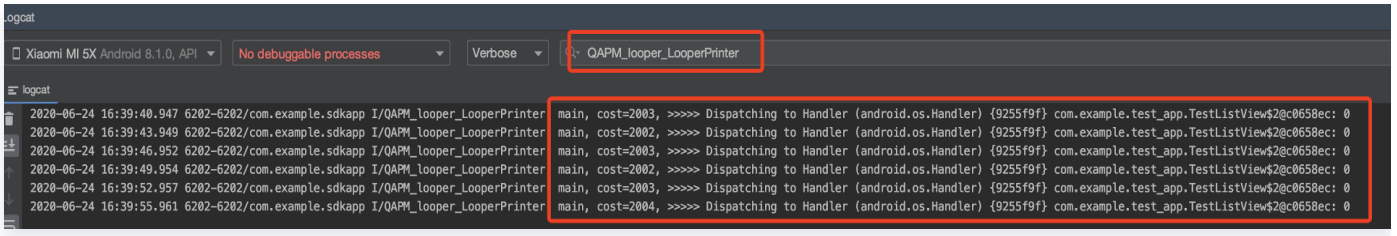
> ⚠ **注意：**
> - 需要使用 qapmplugin 插件进行插桩才可用，否则无效。
> - 问题样本数据在移动监控的网络栏目 > 慢请求/异常请求 > 问题列表查看。

卡顿监控

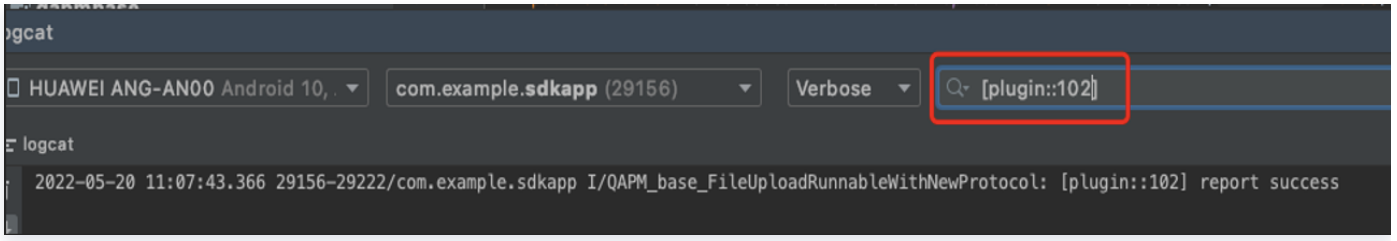
初始化需要开启卡顿监控，卡顿监控无需埋点。

校验功能是否正常

- 如打印以下日志，则代表卡顿监控正常。



- 检索TAG: QAPM_looper_LooperPrinter。
- 如打印以下日志，则代表卡顿上报正常。



- 检索TAG: [plugin::102]。

> ⚠ **注意：**
>
> 1. 卡顿有30%的默认采样配置，即不是每一次的卡顿都会上报。
> 2. 上报后的卡顿在移动监控页面 > 卡慢 > 问题列表查看。

Crash、ANR监控

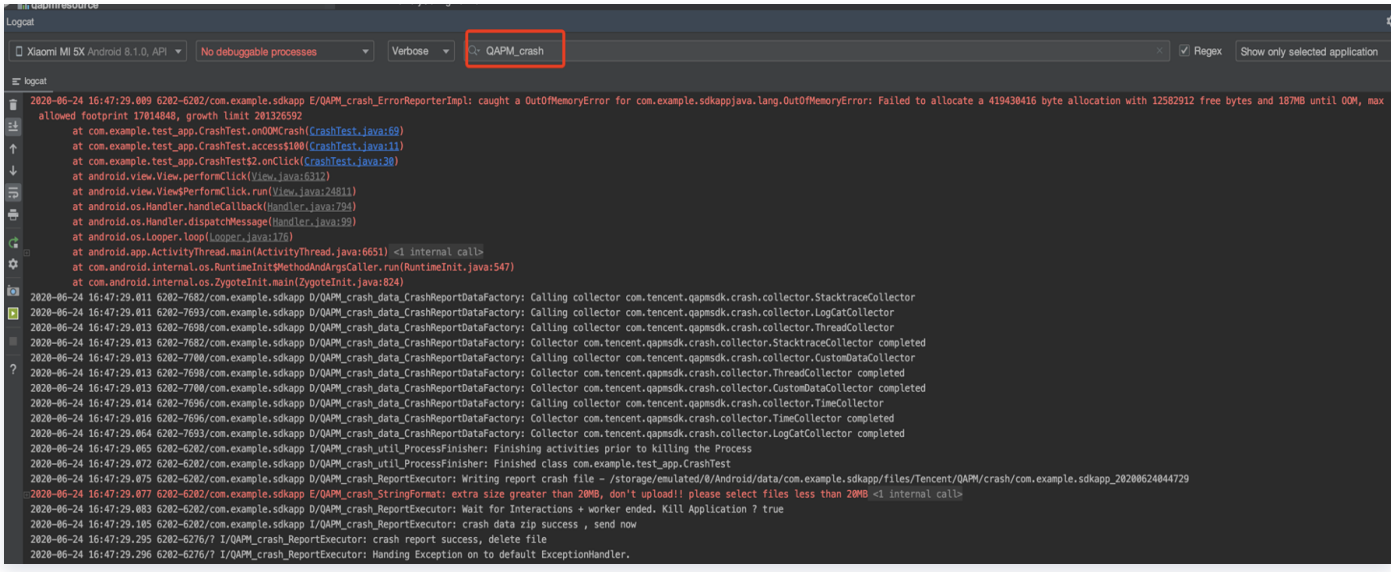
初始化需要开启 Crash、Anr 监控，该监控会默认监控 Crash 和 Anr 信息。
QAPM 提供了相关接口，可以在不幸发生了 Crash 或者 Anr 时，上传用户自定义的日志文件，示例如下：

```
QAPM . setProperty ( QAPM.PropertyExtraDataListener ,  new
IExtraDataListener ( )  {
    // 发生ANR时会走这个回调
    @Override
    public  List <String>  onAnrExtraFileHandler ( )  {
        List <String>  files  =  new  ArrayList <> ( ) ;
        File [ ]  fileArray  =  new  File ( "xxxx" ) . listFiles ( ) ;
        for  ( File  file  :  fileArray )  {
            files . add ( file . getAbsolutePath ( ) ) ;
        }
        return  files ;
```

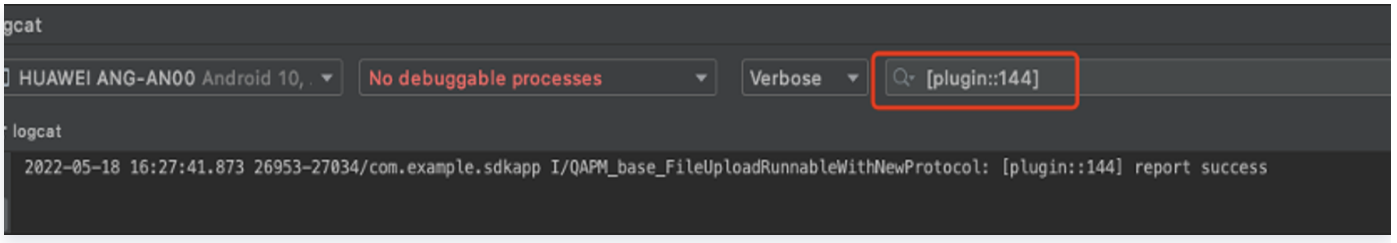
```
        }
        // 发生Crash时会走这个回调
        @Override
        public List<String> onCrashExtraFileHandler() {
            List<String> files = new ArrayList<>();
            File[] fileArray = new File("xxxx").listFiles();
            for (File file : fileArray) {
                files.add(file.getAbsolutePath());
            }
            return files;
        }
    });
```

校验功能是否正常

- 当发生 Crash、ANR 时，打印如下日志，则代表 QAPM 正常收集了此次异常。



- 检索 TAG: QAPM_crash。
- 当打印如下日志，则代表 QAPM 将此次异常上报成功，此处举例 JavaCrash 的上报情况。



- 检索 TAG 分别如下：

  ANR：[plugin::140]
  JavaCrash：[plugin::144]
  NativeCrash：[plugin::146]

> ⚠ **注意：**
> - 为避免出现卡死的情况，接口回调里的逻辑请尽量简单明了。
> - 上传的文件大小限制为20MB，大于限制则不上传，请选择认为有帮助的日志文件。
> - Crash 可以在移动监控的崩溃页面查看，ANR 可以在移动监控的ANR页面查看。

用户行为监控

初始化需要开启用户行为监控，该监控默认会收集用户的点击、滑动、页面切换等事件。
QAPM 提供了相关接口，可用于自定义用户行为事件。

1. **自定义用户行为事件。**

   接口介绍：

```
/**
 * 用户自定义用户行为操作调用,外部用户接口.该方法的所在类为`BreadCrumb`
 *
 * @param category   事件名，强烈建议全大写。示例：USER_PAY，该参数不可为空
 * @param tags       事件关联的一系列属性，为 map<string, string> 类型，该参数可为空
 * @param values     事件关联的一系列数值类属性，该参数可为空
 * @return 事件的id,如果生成失败返回null
 */
public String customEvent(String category,
                          Map<String, String> tags,
                          Map<String, Long> values)
```

   具体示例：

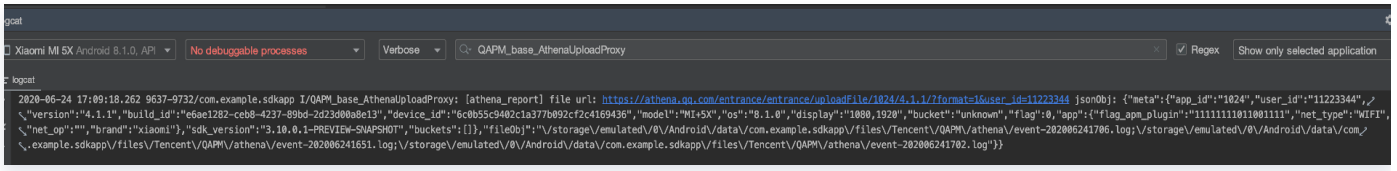
```
Map<String, String> tags = new HashMap<>();
tags.put("d1", "FUJI mini7+");
tags.put("d2", "package:1");
tags.put("d3", "color:white");
tags.put("info1", "富士新手推荐性价比之王拍立得相机mini7+一次成像男女学生款便宜胶片机");
```

```
tags . put ( "info2" ,  "套餐类型:套餐一【官方标配+20张相纸+新品大礼包+配件礼包
10件套】颜色分类:白色" ) ;

Map <String, Integer> values = new HashMap <> ( ) ;
values . put ( "v1" , 748L ) ;
values . put ( "v2" , 1L ) ;

BreadCrumb . getInstance ( ) . customEvent ( "CLICK_BUY_BUTTON" , tags ,
values ) ;
```

2. **校验功能是否正常。**

- 如打印以下日志，则代表用户行为功能开启正常。

- 检索TAG: QAPM_athena。
- 如打印以下日志，则代表用户行为上报正常。



- 检索TAG: QAPM_base_AthenaUploadProxy

用户隐私协议（重要）

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。
参考代码：

```
// 当用户授权后，方可正常初始化QAPM
if  ( isAgree )  {
    // 需要传入设备的唯一标识
    QAPM . setProperty ( QAPM.PropertyKeyDeviceId ,  "设备的唯一标识" ) ;
```

```
        // 需要传入手机型号
        QAPM . setProperty ( QAPM.PropertyKeyModel ,  "填写手机型号" ) ;

        // 正常初始化代码贴入，参考文档2.1部分
    }
```

标识符变更获取方案背景

当前监管要求 SDK 不允许直接或间接采集IMEI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。

标识符变更解决方案

推荐

1. 在5.0.7版本之前已经使用自行传入设备标识符的方式接入 QAPM，则可以继续采用传入自行生成的设备标识符。不会对任何数据造成影响。
2. 在5.0.7版本之前采用的是由 SDK 自行采集设备标识符的方式，更新到5.0.7及以上版本，可以通过自行实现以下方法：

   `Stringutil .getMD5(Build. BRAND + Build.Model + androidId + macAddress + imei)` 。

生成设备标识符并自行传入，其中 Build. BRAND、Build.Model 、 androidId、 macAddress 为必须项，获取相关字段信息的方法可参考附件中的"PhoneUtils.java"文件。若您本身不会获取“IMEI”，“IMEI”这一字段缺省即可。使用通过该方法生成的设备标识符与5.0.7及以前版本的标识符一致，不会对任何数据造成影响。

不推荐

在5.0.7版本之前采用的是由 SDK 自行采集设备标识符的方式，更新到5.0.7及以上版本，采用自行生成标识符号的方式，则会对 crash 率，用户 ANR 率指标的总体数据有影响，需要全部用户都更新到了这一个 SDK 才能消除影响，但是历史数据和新版本的版本级数据均不会受到影响。

此外为了响应隐私合规要求，App 在指定设备上的首次启动，QAPM SDK 将无法采集到首次启动耗时信息，如需要上报首次启动的耗时数据，请调用以下接口进行主动上报（请确保在用户同意隐私政策后再调用）：

```
// startTime   应用最开始的时间，建议打在Application的attachBaseContext方法的最上方
// endTime    应用结束的时间，可以自定义，QAPM打在第一个Activity的onResume方法的最下方
QAPM . sendFirstLaunch ( startTime ,  endTime ) ;
```

> ⚠ **注意：**
>
> - 请确保用户在同意隐私政策之前不调用任何 QAPM 的接口。
> - 如果未传入 deviceId，则会影响用户级指标数据的计算，如用户崩溃率。
> - 如果未传入手机型号信息，则会影响在控制台中结合手机型号维度进行指标及问题样本分析的有效性。

符号表配置

自动上传

1. 在 app 的 `build.gradle` 文件中加入如下代码。

```
QAPMPluginConfig {
    // 必选，AppKey,可在移动监控页面配置->产品配置页获取
    app_key = "YourKey"
    // 必选，符号表的上传地址,外网用户填写https://qapm.qq.com。如果是tmf下，地址为打开移动监控的web页面的host(带端口)
    upload_url = ''
    // 可选，符号表路径，默认本地 build/outputs/mapping/release/mapping.txt
    // symbol_path = ''
}
preBuild.dependsOn(UUIDGenerator) //生成UUID，用于符号表的唯一标识
tasks.matching { task ->
    task.name.startsWith('assembleRelease') //依赖任务执行
}.all { task ->
    task.dependsOn uploadSymbolFile //上传符号表的task
}
```

各项配置如下：

```
QAPMPluginConfig {
    // 必选，AppKey,可在移动监控页面配置->产品配置页获取
    app_key = "YourKey"
    // 必选，符号表的上传地址,请填写https://qapm.qq.com
    upload_url = 'https://qapm.qq.com'
    // 可选，符号表路径，默认本地
${buildDir}/outputs/mapping/release/mapping.txt
    // symbol_path = ''
}
preBuild.dependsOn(UUIDGenerator) //生成UUID，用于符号表的唯一标识
tasks.matching { task ->
    task.name.startsWith('assembleRelease') //依赖任务执行
}.all { task ->
    task.dependsOn uploadSymbolFile //上传符号表的task
}
```

2. 在 QAPM 初始化时加入如下代码。

```
QAPM.setProperty(QAPM.PropertyKeyAppVersion, "Your App Version");
// 设置UUID，用于拉取被混淆堆栈的mapping （必需，若使用了QAPM符号表上传插件，可以直接使用该变量）
QAPM.setProperty(QAPM.PropertyKeySymbolId, BuildConfig.QAPM_UUID);
// 设置用户ID，用于后台检索字段（必需）
QAPM.setProperty(QAPM.PropertyKeyUserId, "123456");
```

参考代码：

```
QAPM . setProperty ( QAPM.PropertyKeySymbolId ,
BuildConfig . QAPM_UUID ) ;
```

3. 执行编译，如看到以下日志，则自动上传符号表的配置则成功了

```
> Task :app:uploadSymbolFile
QAPM_UUID: aef3e99f-7b2e-490f-b1d3-dde17081be85
POST statusLine: HTTP/1.1 200 OK, Response: [status:ok, data:success]
mappingFile: /Users/jiangyunsheng/Desktop/workspace/QAPM_SDK/app/build/outputs/mapping/release/mapping.txt uploaded successfully
```

> ⚠ **注意：**
> 上传符号表的功能包含在 qapmplugin 插件中，如果未使用则无法使用自动上传符号表功能。

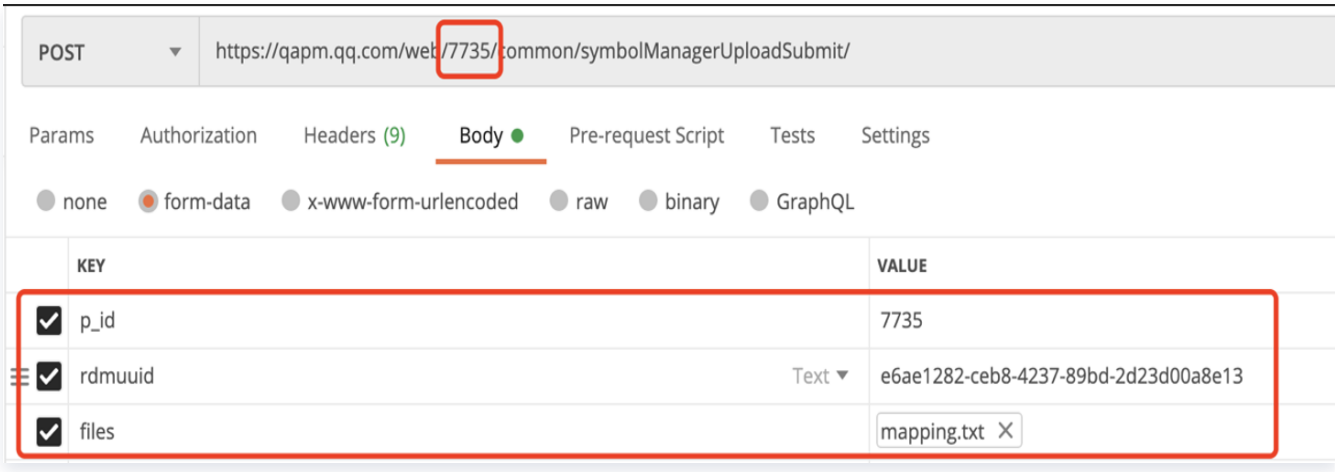
手动上传

针对无法通过打包自动上传符号表的情况这里有两个方式上传。

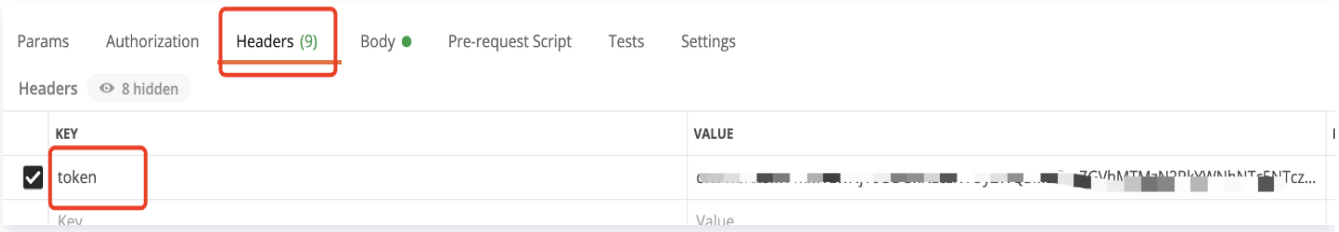
1. **利用接口上传 mapping 文件（推荐）**

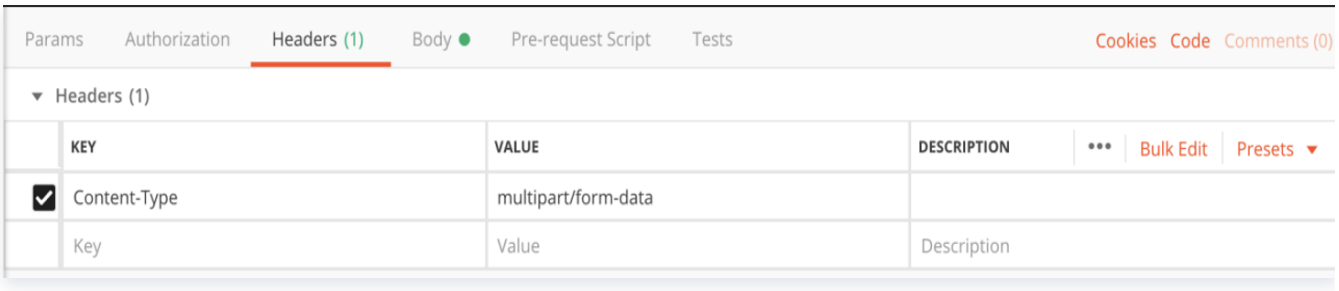
   上传符号表接口：`https://qapm.qq.com/web/common/symbolManagerUploadSubmit/` 。
   报文 header：`Content-Type = multipart/form-data` 。
   参数为：
   p_id : int 产品 pid，即 appkey 后面的数字。
   rdmuuid：string 编译 id，可以在代码总打印输出 `BuildConfig.QAPM_UUID` 变量获取。
   files： string mapping 文件路径。
   例如这里可以通过 postman 构建上报报文类似如下：

   - 设置返送方式为 Post, 并填写 url。

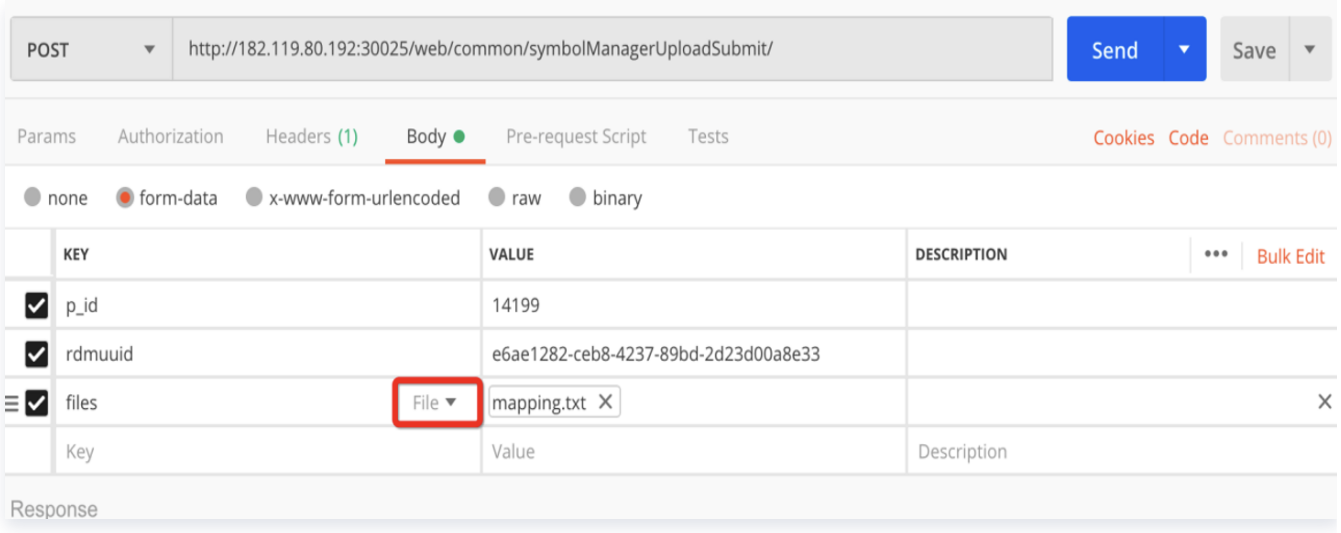


- 设置报文头为，token 通过该接口获取
  `https://qapm.qq.com/web/102/api/getToken?app_key={appkey}` 。





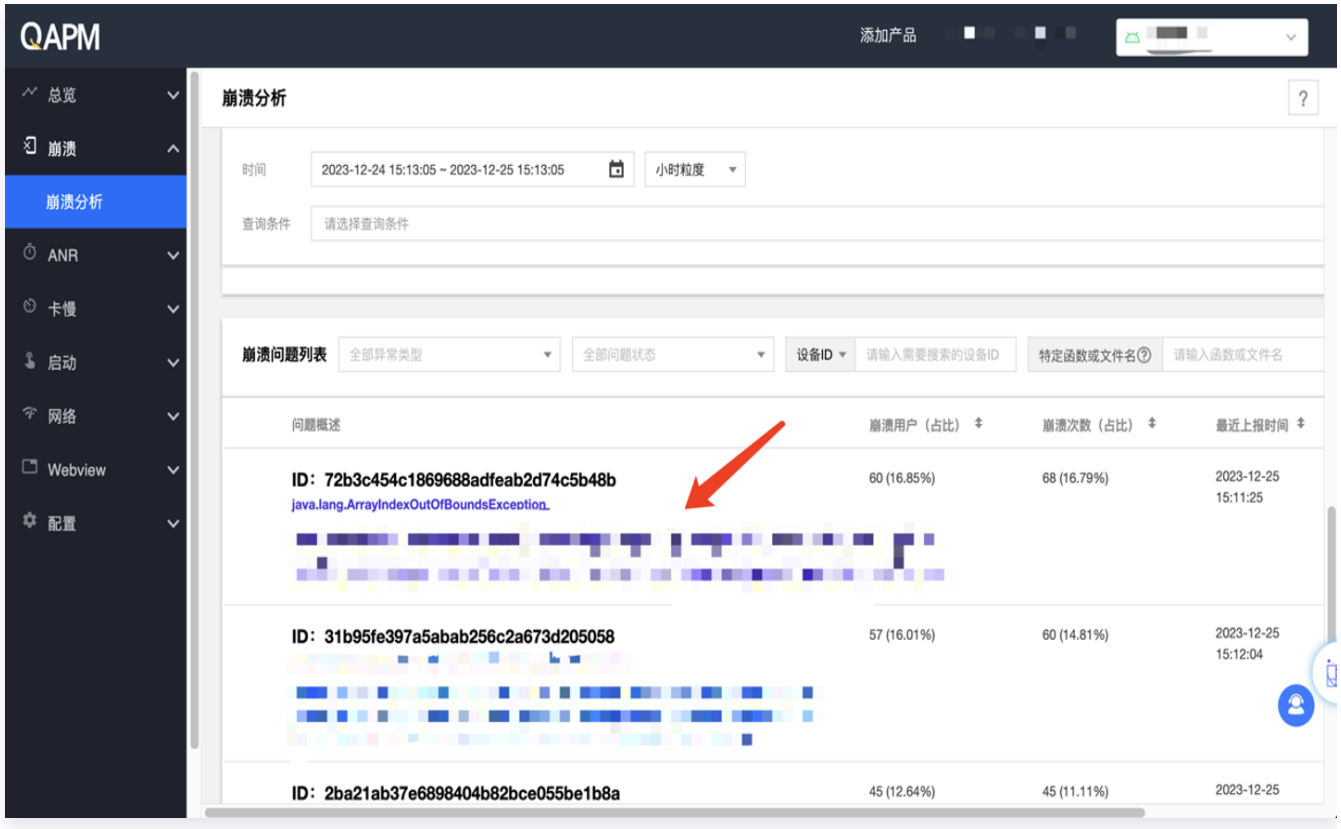
- 其参数设置如下，注意这里的 files 设置为 File 类型。



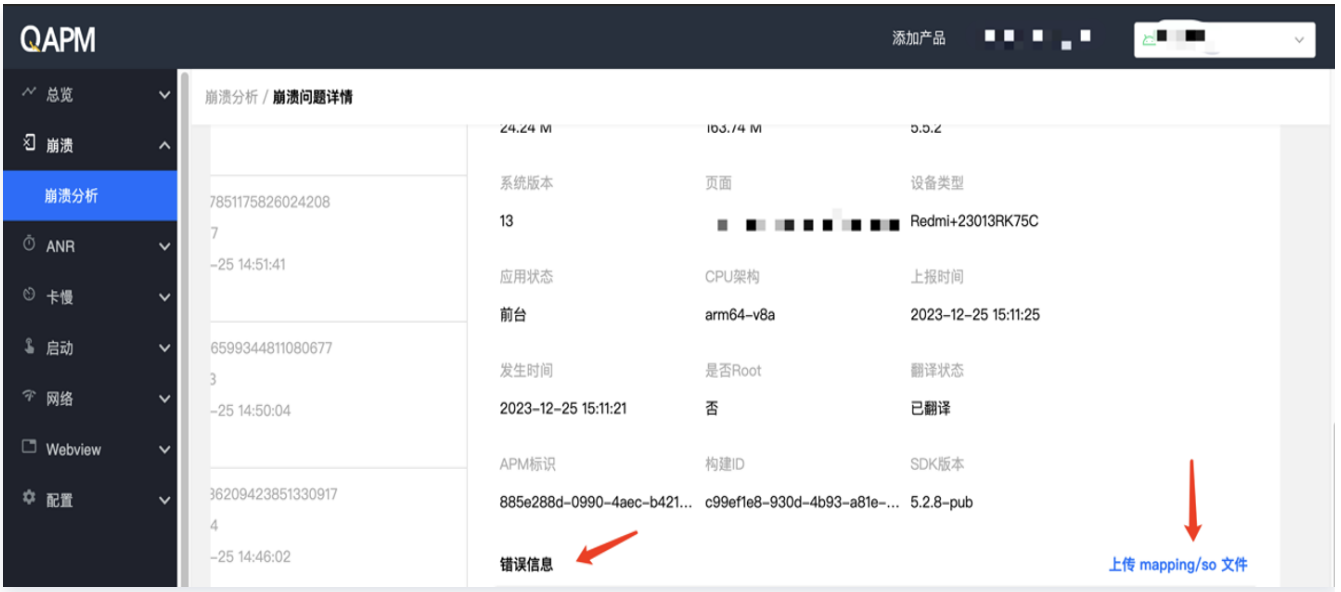
- 点击 send 按钮查看回包，如果回包为如下字段表示发送成功。

```
{"status": "ok", "data": "success"}
```

2. 利用已上报的问题样本在控制台完成符号表上传（该方法对已上报的问题样本不会再做重新翻译处理）
   - 可以通过问题列表 > 问题详情 > 错误信息 > 上传符号表文件的路径完成符号表文件的上传。



- 在问题详情页面中点击上传符号表按钮。



- 进入符号表上传页面，点击**选择文件**，然后选择该构建 ID 对应的 maping 文件&so 文件打包的zip压缩包即可。该构建 ID 为初始化时 `QAPM.setProperty(QAPM.PropertyKeySymbolId,"xxxxx")` 设置的参数，该 ID 需要遵循 UUID 格式,否则初始化设置无效。

iOS SDK 接入

最近更新时间：2024-07-26 17:28:24

SDK 集成

- **手动集成**

  1.1 下载 SDK。

  1.2 拖拽 QAPM.framework 文件到 Xcode 工程内（请勾选Copy items if needed 选项）。

  1.3 在 TARGETS > Build Phases-Link Binary Libraries 添加依赖库：

  - libc++.dylib (libc++.tbd)。
  - libz.dylib (libz.tbd)。

  1.4 在工程的 Other LinkerFlags 中添加 `-ObjC` 参数。

  1.5 将 framewor k里面的 `js_sdk.js` 文件导入到工程根目录。

- **cocoaPods 集成**

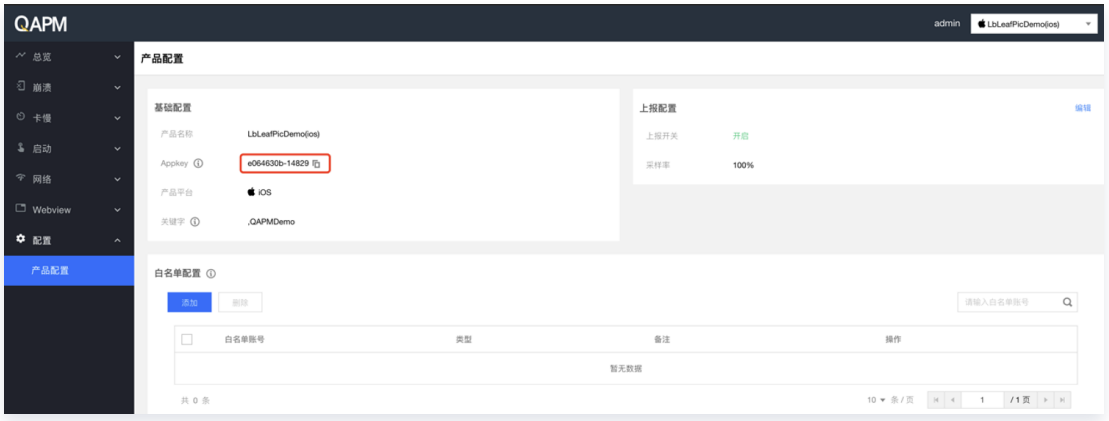
  在 podfile 文件中增加如下操作：
  `pod 'QAPM',:source => 'https://github.com/TencentCloud/QAPM-iOS-CocoaPods.git'`，
  然后执行 pod install 指令。

> **注意：**
> iOS SDK 最低兼容系统版本 iOS 8.0。

SDK 启动

SDK 初始化

1. 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。



2. 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。

3. 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 {
//为响应工信部“26号文”要求，提供该设置用于告知SDK是否可以进行可选个人信息的采集，默认可以采集。设置为NO则不采集。该设置需要最先配置，一旦设置则全局生效
//可选个人信息包括但不限于以下信息：设备制造商、系统、运营商等等，详见《QAPM SDK合规使用指南》
//该设置默认设置是YES，代表允许可采集，示例代码如下：
[QAPMConfig getInstance].collectOptionalFields = YES;

// 特别说明： 若上述该设置置为NO，部分信息将不再获取，可能会影响到前端的搜索、展示等，请知悉。

/// 设置QAPM 日志输出
NSLog(@"qapm sdk version : %@", [QAPM sdkVersion]);
[QAPM registerLogCallback:loggerFunc];

//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;
}
```

> **注意：**
>
> 1. AppKey 请到移动监控的配置页 > 产品配置页获取。
> 2. 研发流程内，可以将设置的用户 ID /设备 ID 添加成白名单，确保指定设备的功能上报不会被采样影响，可在控制台中通过配置 > 产品配置 > 白名单配置 > 添加白名单完成操作。

监控功能启用

推荐使用：

- 在 `QAPMModelStableConfig.h` 文件中的接口：
  `- (void)setupModelAll; //此接口可开启全部监控功能。`
- 在 `QAPMModelStableConfig.h` 文件中接口：
  `- (void)setupModelStable; //此接口可开启除了网络监控和用户行为监控之外的所有监控功能，网络监控与用户行为监控按照工信部26号文要求划分为扩展业务功能，您可根据自身需要选择是否开启。`

自定义使用：

如果要自定义开启 QAPM 功能（包括卡顿、crash、原生网络、原生用户行为、启动、webview），可在 `QAPMConfig.h` 文件中的 `enableMonitorTypeOptions` 接口，设置组合的枚举值，添加多个参数枚举值完成一系列功能的开启，代码如下：

```
// 设置QAPM 开启的监控：
[ QAPMConfig  getInstance ] . enableMonitorTypeOptions  =
    QAPMMonitorTypeBlue  |
    QAPMMonitorTypeCrash  |
    QAPMMonitorTypeHTTPMonitor  |
    QAPMMonitorTypeIUPMonitor |
    QAPMMonitorTypeLaunch  |
    QAPMMonitorTypeJSError  |
    QAPMMonitorTypeWebViewNetWork  |
```

```
    QAPMMonitorTypeWebViewIUPMonitor  |
    QAPMMonitorTypeWebMonitor ;
```

接口说明：

接口： `QAPMConfig.h` **类中**

`@property (nonatomic, assign) QAPMMonitorType enableMonitorTypeOptions;`

参数说明： `QAPMMonitorType` **类型为功能类型，可选值为** 下述代码 **所示。**

可选项说明：

```
///检测卡顿功能
QAPMMonitorTypeBlue
/// Crash监控功能
QAPMMonitorTypeCrash
///原生网络监控
QAPMMonitorTypeHTTPMonitor
///原生用户行为监控
QAPMMonitorTypeIUPMonitor
/// 启动个例监控功能
QAPMMonitorTypeLaunch
/// webview的JS异常监控
QAPMMonitorTypeJSError
///webview的网络
QAPMMonitorTypeWebViewNetWork
///webview 用户行为监控
QAPMMonitorTypeWebViewIUPMonitor
/// webview页面性能监控
QAPMMonitorTypeWebMonitor
```

注意：

1. **使用或运算方式自定义开启所需监控功能，如卡顿：** `QAPMMonitorTypeBlue` **。**
2. **为响应工信部** “26号文” **要求，我们依据工信部对性能监控类 SDK 基础功能的定义，将网络监控与用户行为监控划分为扩展业务功能，这意味着为了避免采集网络日志信息、用户操作记录等个人信息，您可以选择性开启这两个功能。操作层面您可以在自定义性能模块开启配置中避免填入** `QAPMMonitorTypeHTTPMonitor` **、** `QAPMMonitorTypeWebViewNetWork` **两个参数，以关闭网络监控功能；以此类推，您可以在自定义性能模块开启配置中避免填入** `QAPMMonitorTypeIUPMonitor` **、** `QAPMMonitorTypeWebViewIUPMonitor` **两个参数，以关闭用户行为监控功能，或直接使用 ModelStable 功能开启模式以在确保关闭所有扩展业务功能的情况下自动开启其他推荐功能。**

SDK功能介绍

卡顿及流畅度监控功能

- **卡顿检测功能**

  QAPMMoniterType：QAPMMonitorTypeBlue 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立即上报。

- **流畅度监控功能**

  在滑动场景下相关页面进行如下代码的打点即可开始统计流畅度，日志会在下次启动 App 后上报数据。

```
#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])];
}
```

Crash监控功能

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

> ⚠ **注意：**
>
> 1. 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。
> 2. 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%的采样抽样。

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

在工程对应的类里面导入头文件 `#import <QAPM/QAPMLaunchProfile.h>` **。**

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]));
    }
}
```

Web监控功能

> **说明：**
> 1. 目前 WKWebView 支持 iOS ⩾ 11 。
> 2. 该功能能够监控 Web 网络资源加载耗时、jserror 监控。

功能配置

- Web 端配置

> ⚠ **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/QAPMLaunchProfile.h>导入 SDK 头文件。
  - 在 WKWebView 的代理方法 webView:didFinishNavigation: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 ] ;
}
```

用户隐私协议

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。

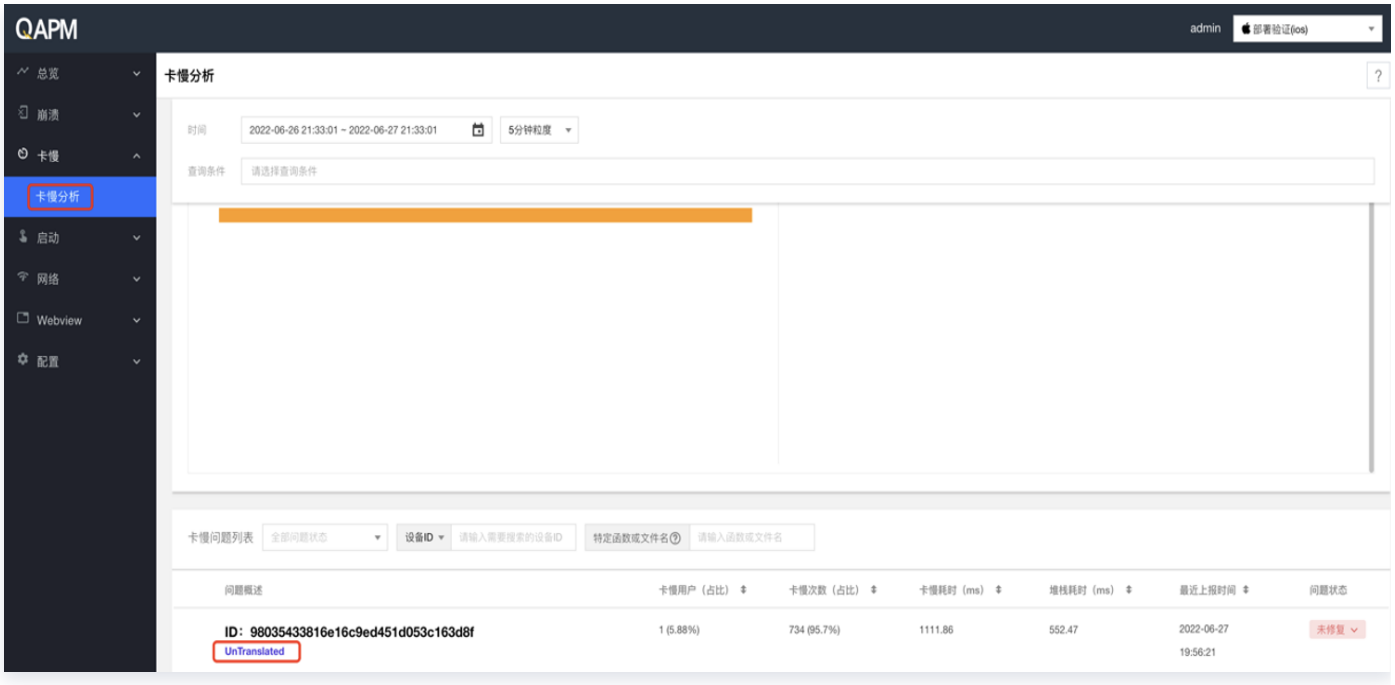
```
// 当用户授权后，方可正常初始化QAPM
if （isAgree） {        }
//启动耗时函数的第一个打点
// 需要传入设备的唯一标识，如IDFV配合Keychain使用
[QAPMConfig getInstance].deviceID = @"自定义deviceId";
//用户user ID、第三方登录账号，此接口可以多次在代码位置使用
[QAPMConfig getInstance].userId = @"设置userId";
```

说明：
标识 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。

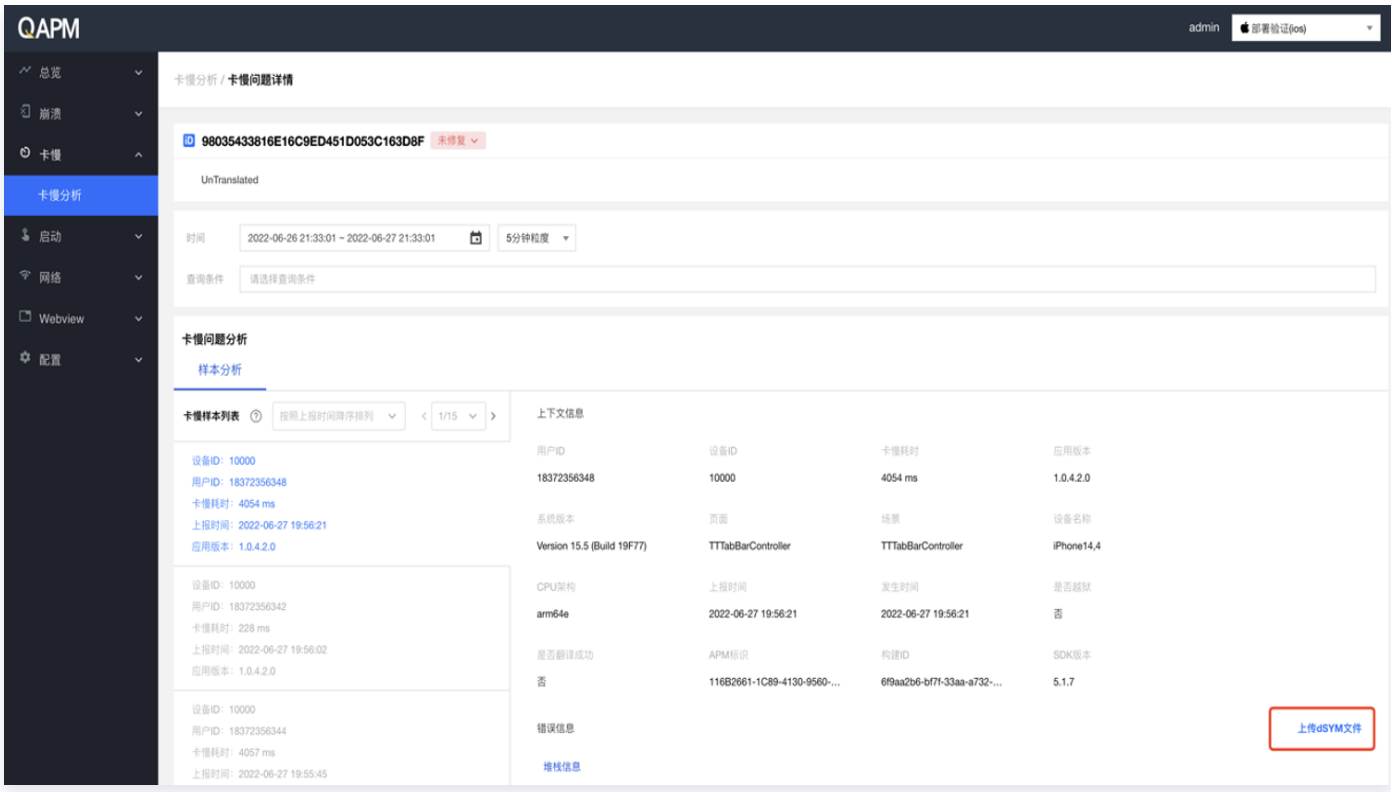
符号表配置

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。

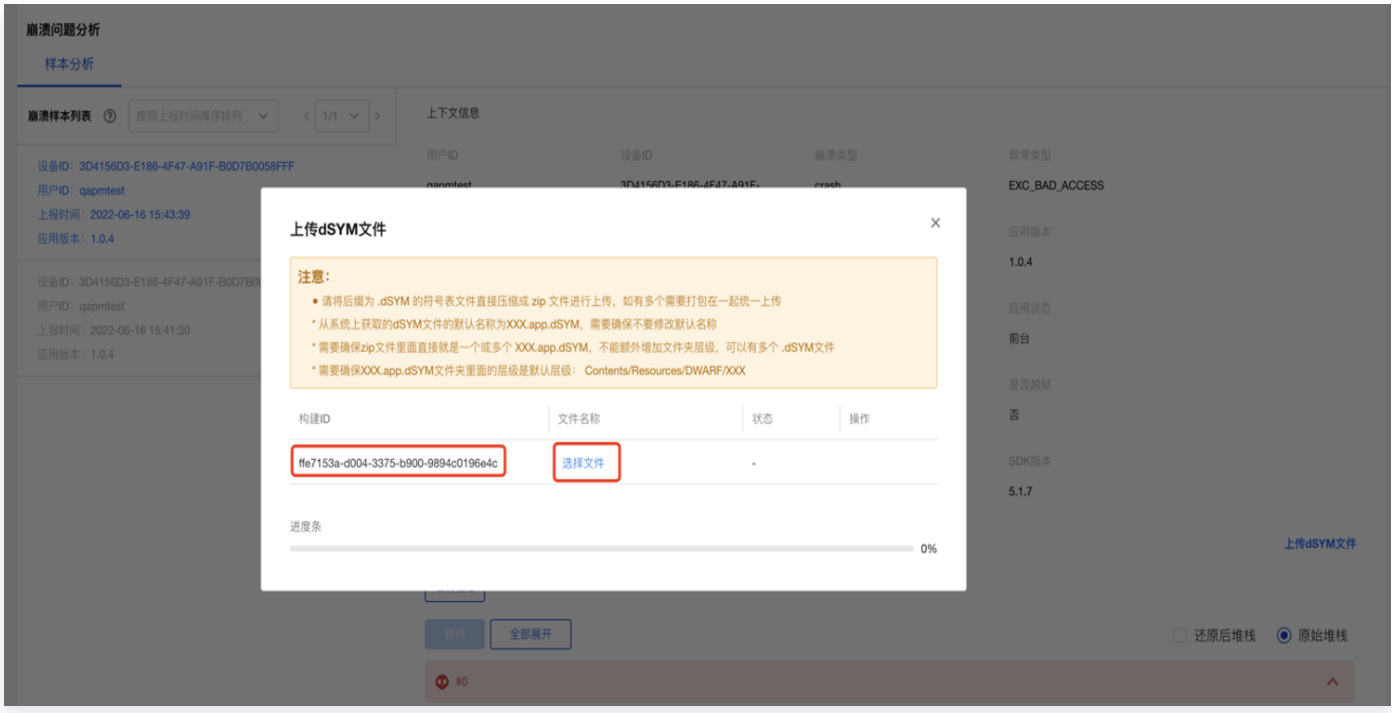
1. 在个例页面中有上传的入口，例如选择**卡慢** > **卡慢分析** > **卡慢问题列表**，进入详情页面。



2. 在详情页面中，单击**上传dSYM文件**。



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



查看QAPM工作日志

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
}
```

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：

```
[QAPM_LogEvent][QAPM.m:327][Config] [Custom lag monitoring function] The background is not allowed to be enabled
[QAPM_LogEvent][QAPM.m:331][Config] [Frame drop rate monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:335][Config] [Blue (lag monitoring) function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:339][Config] [Yellow(VC leak detection function)] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:343][Config] [Foom monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:347][Config] [Deadlock monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:351][Config] [Crash indicator] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:355][Config] [QQLeak memory leak detection function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:359][Config] [Resource usage monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:363][Config] [Memory maximum usage value monitoring (peak rate function)] is not allowed in the background
[QAPM_LogEvent][QAPM.m:368][Config] [Chunk malloc monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:373][Config] [NormalCrash monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:377][Config] [User behavior monitoring AthenaSDK function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:381][Config] [Power consumption monitoring] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:385][Config] [Network monitoring] background is not allowed to be enabled
[QAPM_LogEvent][QAPM.m:389][Config] [JSerror] background is not allowed to be enabled
[QAPM_LogEvent][QAPM.m:393][Config] [WebView Slow Request] background is not allowed to be enabled
[QAPM_LogEvent][QAPM.m:396][Config] [Launch monitoring function] is not allowed to be enabled in the background
```

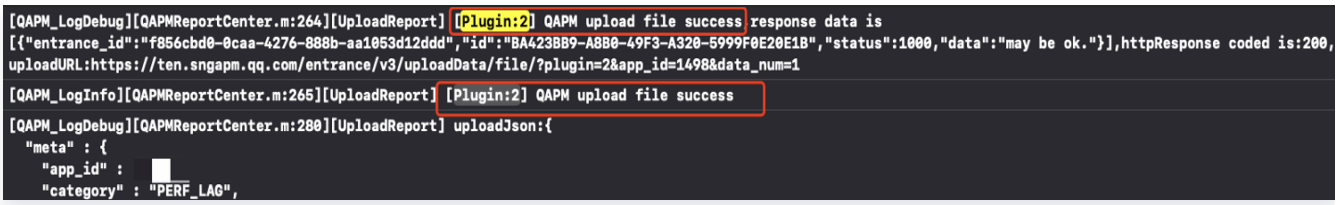
- 监控功能开启时，日志如下：

```
[QAPM_LogEvent][QAPM.m:179][Config] [Normal crash monitoring function] is enabled
[QAPM_LogEvent][QAPM.m:226][Config] [User behavior monitoring AthenaSDK function] is enabled
[QAPM_LogDebug][QAPMEventConn.m:110][Athena] appid:1498,userid:qapmcharles,buildid:unknown,version:5.3.1-qapm-sdk-debug
[QAPM_LogDebug][QAPMEventConn.m:133][Athena] The current performance event reporting logic is real-time JSON reporting, and the reporting interval is 1 minute
[QAPM_LogEvent][QAPM.m:229][Config] [ubs monitoring function] is enabled
[QAPM_LogEvent][QAPM.m:277][Config] [Blue (lag monitoring) monitoring function] is enabled
[QAPM_LogEvent][QAPM.m:284][Config] [HTTP monitoring function] is enabled
[QAPM_LogEvent][QAPM.m:292][Config] [WebMonitor] background configuration is allowed to be enabled, please click on the corresponding page
```

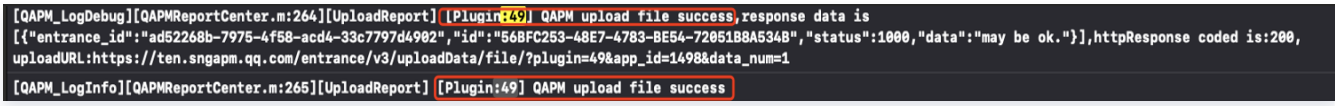
- 通过初始化日志，可以看到初始化成功，各个监控功能开启，然后就是各功能上报成功的验证。
  - 启动耗时的上报

```
[QAPM_LogDebug][QAPMReportCenter.m:264][UploadReport] [Plugin:66] QAPM upload file success,response data is
[{"entrance_id":"1ea1521a-5d4f-40b0-8c6e-366a93e4b065","id":"E0BA3A6E-4D52-4912-85B7-2B73981D9A77","status":1000,"data":"may be ok."}],httpResponse coded is:200,
uploadURL:https://ten.sngapm.qq.com/entrance/v3/uploadData/file/?plugin=66&app_id=1498&data_num=1
[QAPM_LogInfo][QAPMReportCenter.m:265][UploadReport] [Plugin:66] QAPM upload file success
[QAPM_LogDebug][QAPMReportCenter.m:280][UploadReport] uploadJson:{
  "meta" : {
```

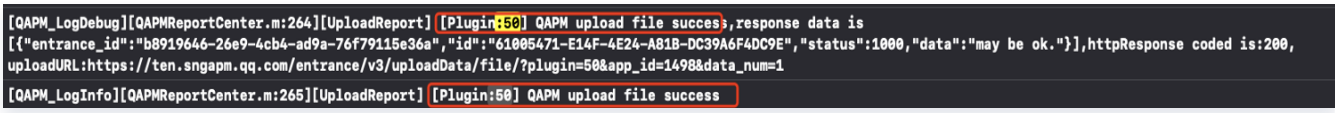
  - 卡顿个例的上报



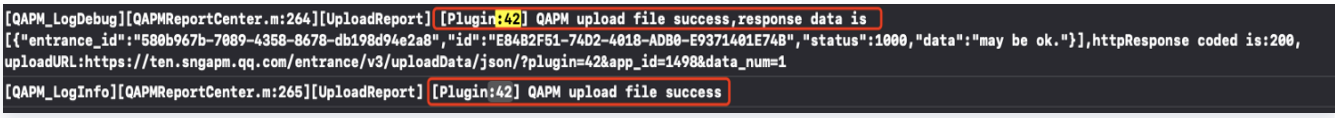
- FOOM个例上报



- Deadlock个例上报

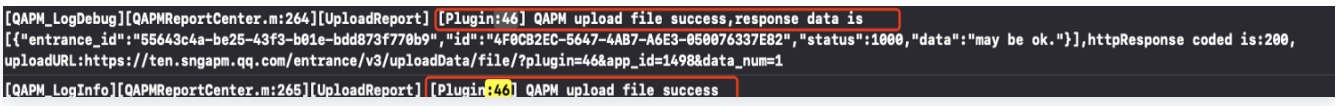


- HTTP 监控上报



- 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:



- 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。