腾讯云可观测平台

终端性能监控

腾讯云

文档目录

终端性能监控
终端性能监控概述

最近更新时间：2024-09-13 21:32:41

简介

终端性能监控是全方位定位检测 App 应用性能和用户体验的工具，多维度自动分析出各维度存在可疑的性能缺陷。帮助您精确衡量 App 应用的性能，以低成本、高效率发现 App 应用各类问题。

APP 监控功能说明

功能名称	说明
崩溃（Crash）	监听 Java 崩溃、Native 崩溃、mach 异常、OC 异常、FOOM、死锁等双端多种崩溃异常问题，并提取 issue 特征进行聚类，帮助客户精准的解决问题，便捷的管理问题。
ANR	多维度还原线上用户真实体验，通过收集用户真实使用 App 过程中碰到的 ANR 问题及问题发生时各线程堆栈信息，提取关键特征聚类。
启动	主动识别 iOS 与 Android 双端用户慢启动问题，能够区分热启动、冷启动、首次启动等不同启动场景，有效识别慢启动问题并收集代码级线索辅助问题解决。
卡慢	收集与分析 iOS 与 Android 双端用户交互流畅度与卡慢问题，能够精准识别主线程消息执行超时现象，聚类问题并提供预分析结果及火焰图等多种图表提升问题分析效率。
网络	从慢与异常两个角度收集网络性能问题，通过分段耗时、错误码、请求头等关键信息结合地区、ISP、IP、网络类型、LocalDNS 等维度定位网络性能瓶颈，保障用户体验。
WebView	• 仅通过集成一个 SDK 即可获得内嵌 Web 页面性能的监控与分析，通过趋势、分组、分布、分位、多维、瀑布图等分析工具，对 Webview 慢页面问题进行系统治理； • 提供多种图表可对 JS 异常趋势、分布进行有效分析，并提供 SourceMap 还原功能，使 JS 错误分析定位直达代码行。

数据存储说明

- 指标数据：存储 90天。
- 问题样本数据：存储90天。

接入指南
安卓应用场景
集成和初始化

最近更新时间：2024-12-30 11:40:22

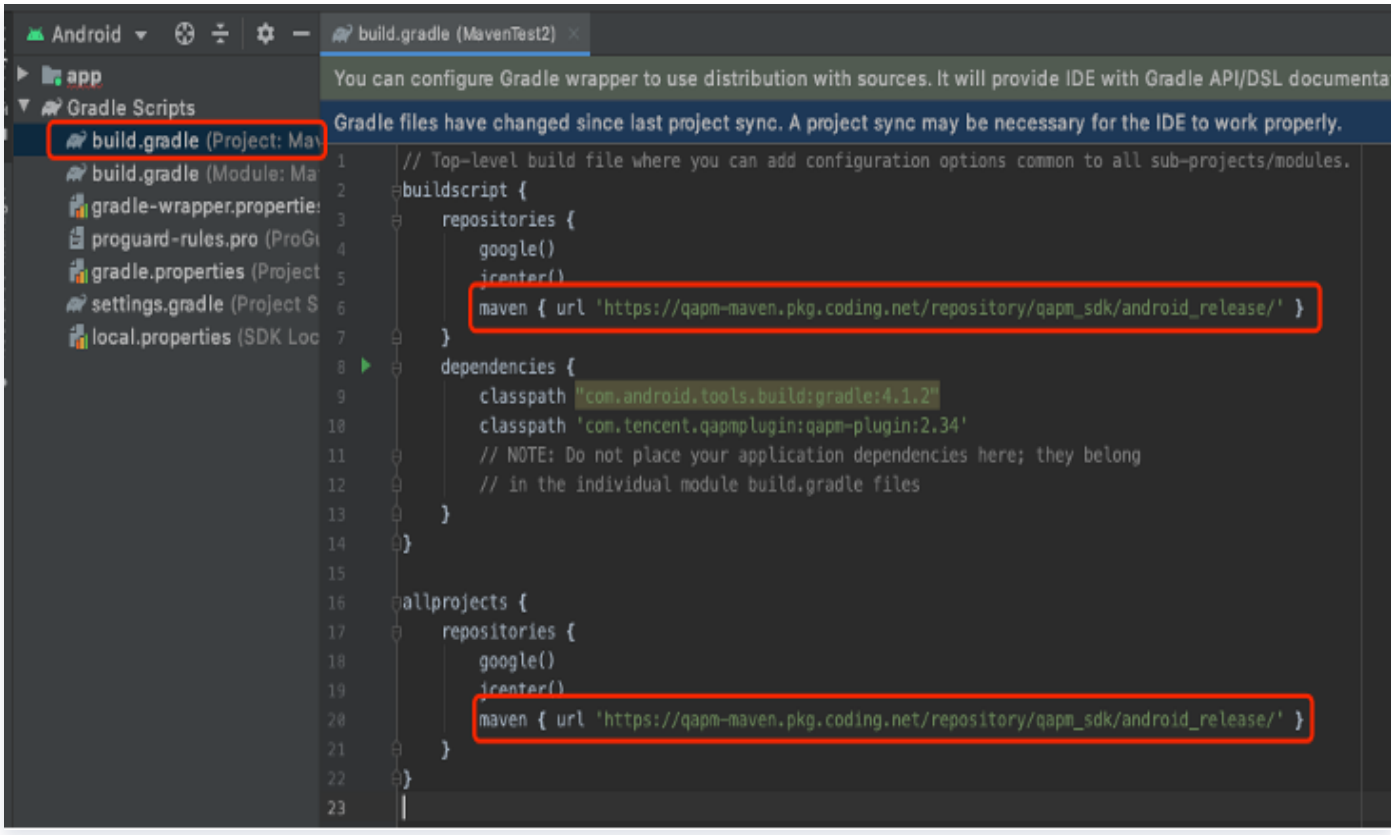
本文指导您使用 Android SDK 的集成与初始化。

> **说明：**
> 隐私合规相关配置请查看 合规使用指南 。

操作步骤

步骤1：Gradle 集成

1. 在 **settings.gradle** 中添加 maven 仓库源。

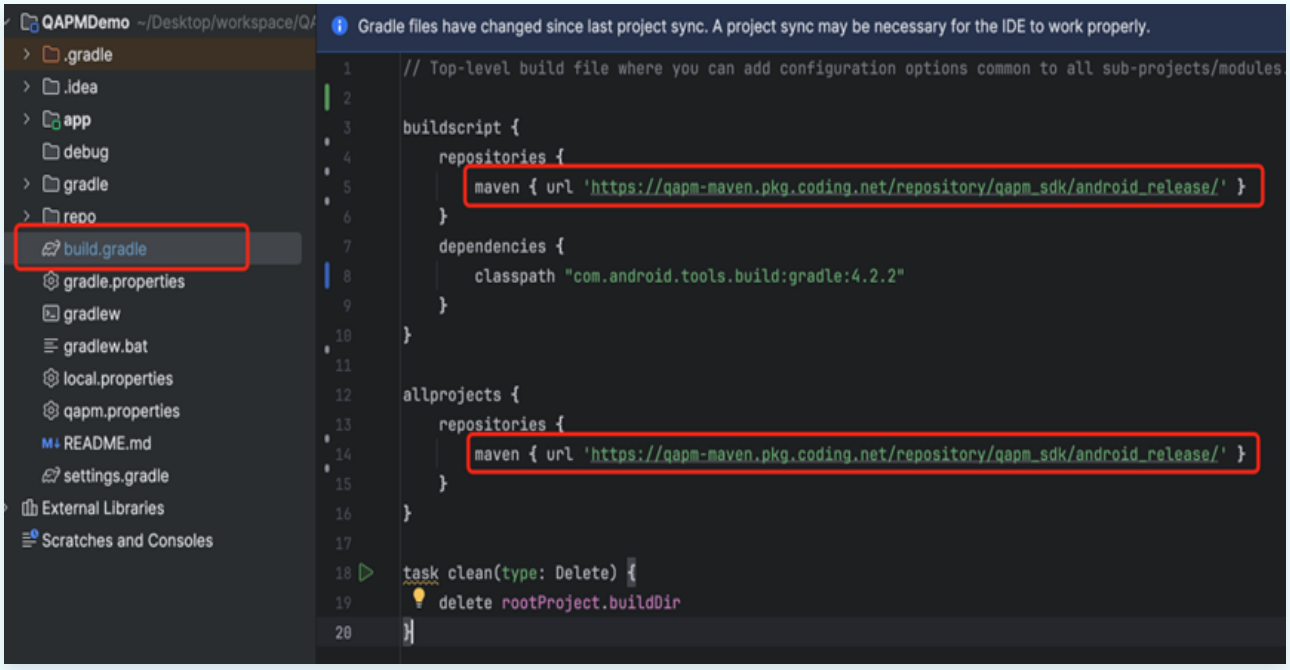


参考代码：

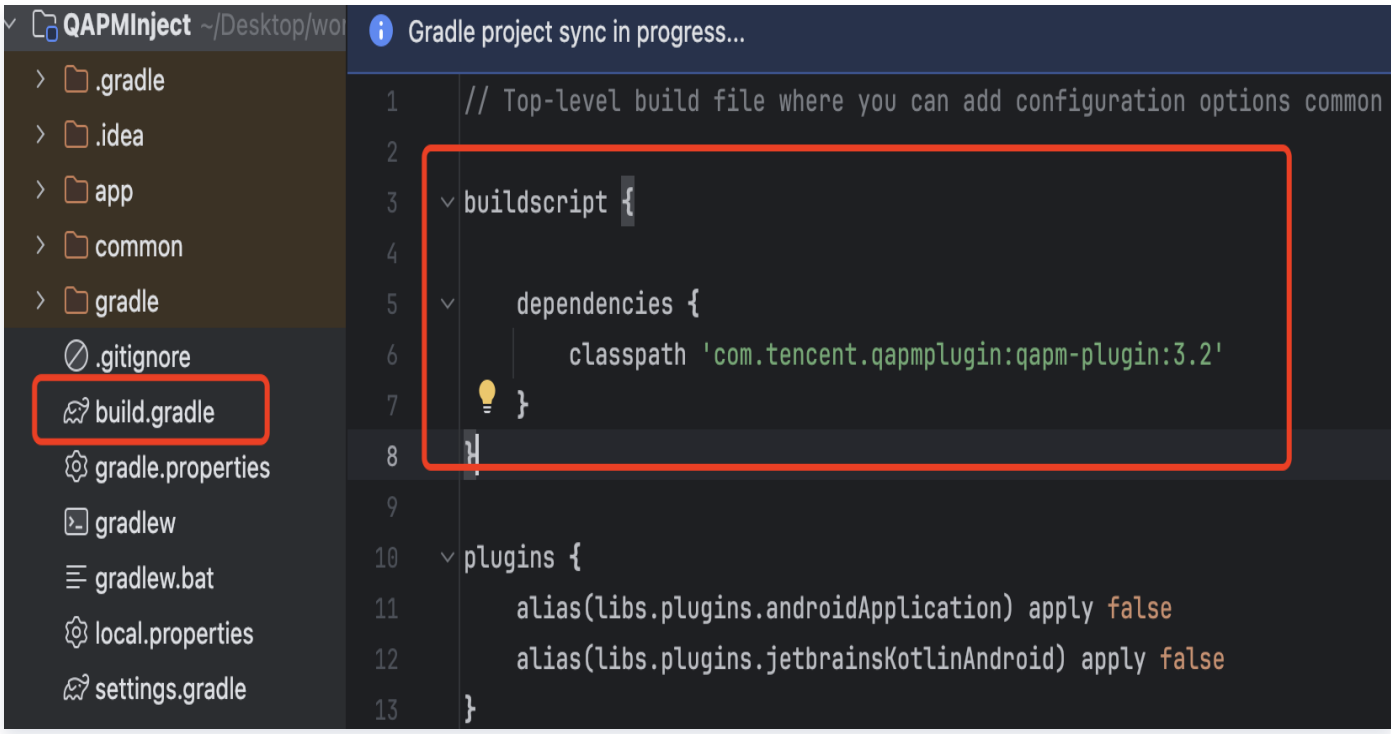
```
  pluginManagement {
      ...
      repositories {
        ...
        // 加入下面内容
        maven { url 'https://qapm-
maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
      }
  }

  dependencyResolutionManagement {
      ...
      repositories {
        ...
        // 加入下面内容
        maven { url 'https://qapm-
maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
      }
  }
```

注意：

如果您的 gradle 版本低于 7.0，请在 project 的 build.gradle 添加 maven 仓库源，如下所示：



2. 在 **project** 的 **build.gradle** 文件下增加插件的依赖。

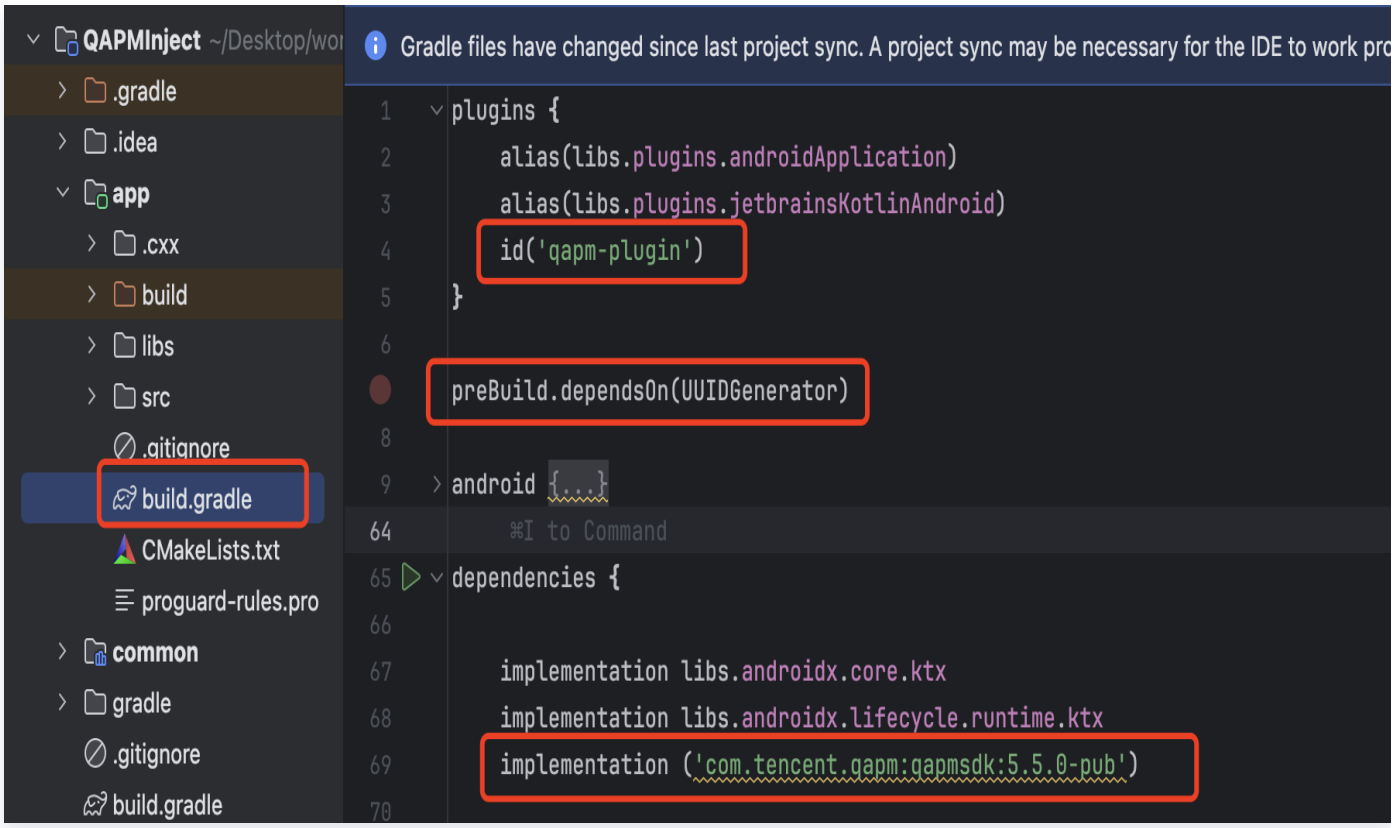


参考代码：

```
buildscript {
        ...
    dependencies {
        ...
        // 加入下面的内容
        classpath 'com.tencent.qapmplugin:qapm-plugin:3.2'
        ...
    }
}
```

> **注意：**
> 如果您的 **gradle 版本小于 7.4**，则将插件版本改为 2.39。

3. 在 **app 的 build.gradle** 文件下增加以下代码：



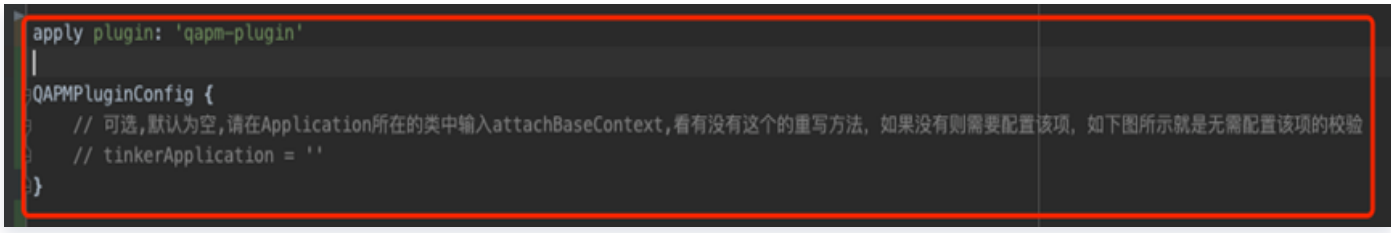
参考代码：

```
plugins {
    ...
    id('qapm-plugin') //添加插件
    ...
}

...
preBuild.dependsOn(UUIDGenerator) //生成唯一标识，用于后续堆栈翻译
...

dependencies {
    ...
    //添加qapmsdk依赖
    implementation 'com.tencent.qapm:qapmsdk:5.5.0-pub'
    ...
}
```

4. 请通过以下内容检查是否需要执行此步骤。

请在 Application 所在的类中输入 attachBaseContext，检查是否有这个的重写方法，如有重写方法则忽略该步骤，如没有请执行下一步。

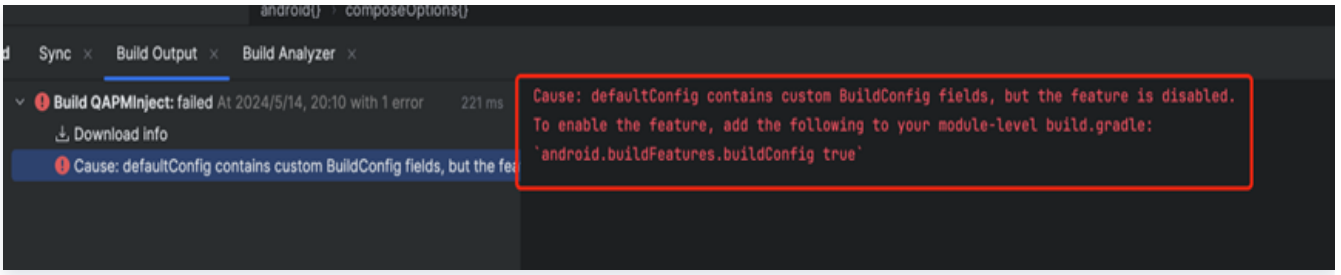
请将 Application 的包名路径添加进以下配置，如下所示：



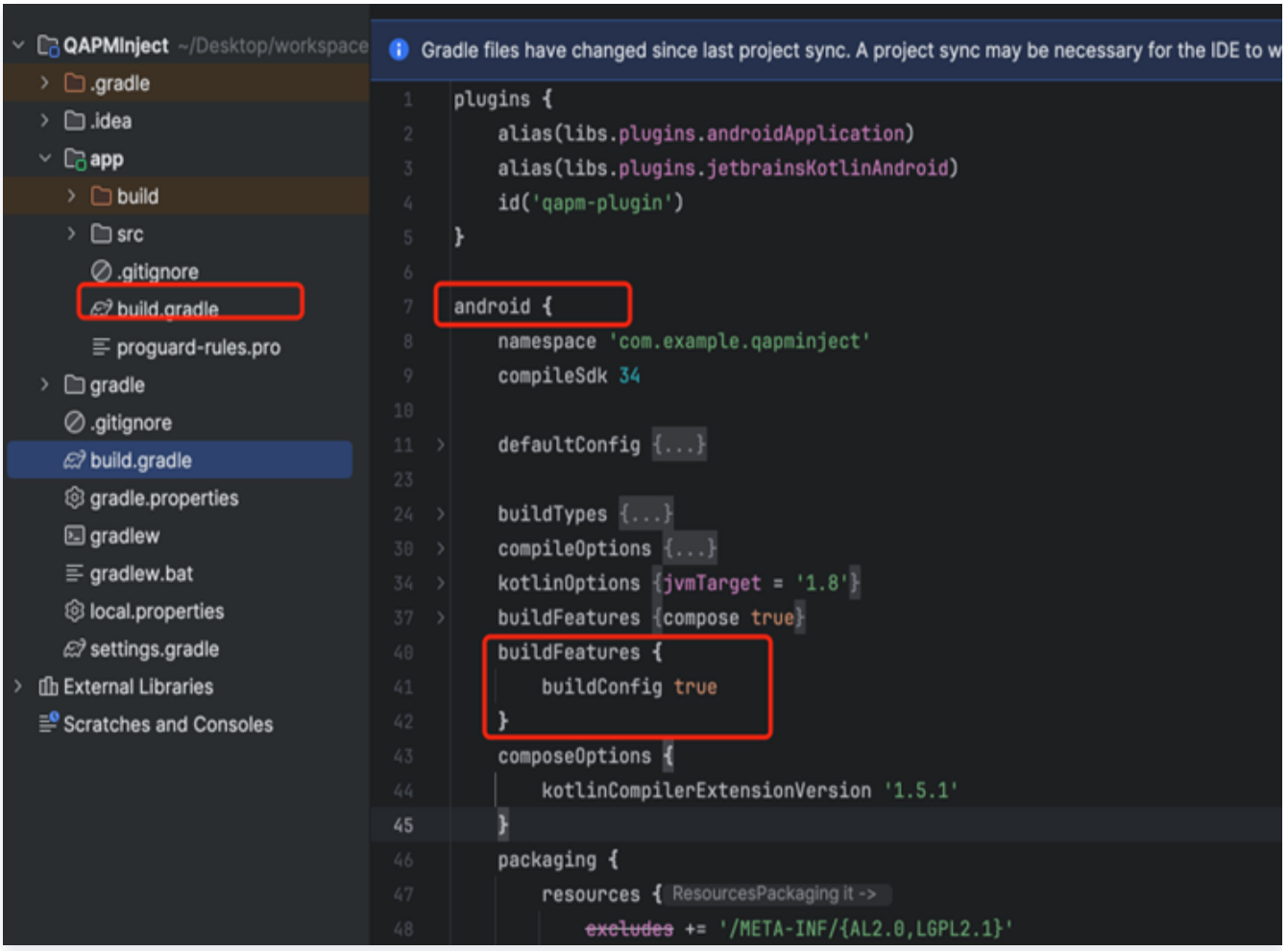
参考代码：

```
QAPMPluginConfig {
    // 可选，默认为空。请在Application所在的类中输入attachBaseContext，检查是否有这个重写方法，如果没有则需要配置该项，如下图所示就是无需配置该项的校验
    // tinkerApplication = 'com/tencent/qapm/demoApplication'
}
```

5. 此时您可以尝试进行编译。**编译相关的 FAQ** 可以查看此步骤。
    - **Q1：如果编译时出现 “feature is disabled” 的错误，如下：**



A1：这是因为插件需要动态在 BuildConfig 插入属性，请在 app 模块的 build.gradle 文件下增加以下代码。

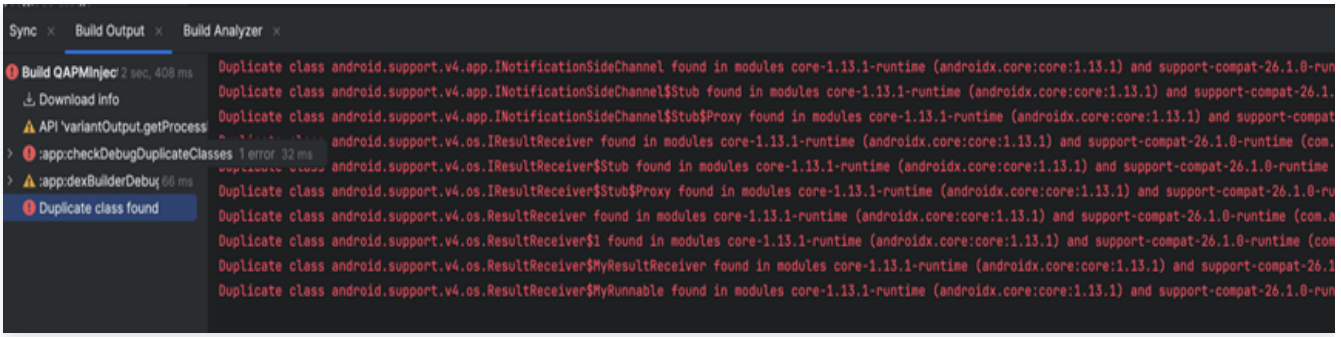


参考代码：

```
android {
    ...
    // 加入下面内容
    buildFeatures {
```

```
        ...
        buildConfig true
    }
    ...
}
```

- **Q2：如果编译时出现 “类冲突” 的错误，如下：**



A2：这是由于 qapm 与您的工程使用了 android 不同的 support 库，这里请在依赖的时候移除掉qapm 的 android.support 库，如下：

参考代码：

```
dependencies {
    ...
  implementation ('com.tencent.qapm:qapmsdk:5.5.0-pub') {
    // 加入下面内容
    exclude group: 'com.android.support'
  }
  ...
}
```

> ⚠ **注意：**

> 如果未使用 qapm-plugin 插件，则会影响启动、网络的监控。

步骤2：参数配置

1. 在 AndroidManifest.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. 为了避免混淆 SDK，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.**{*;}
```

步骤3：初始化 SDK

1. 登录 腾讯云可观测平台 控制台，在**终端性能监控**页面，选择 应用管理 > 应用设置 后，获取 Appkey（上报 ID）。

2. 拷贝下面代码，并修改其中部分字段。确保必需的接口被正确配置，其余接口配置请参考初始化的接口分析（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，该设置需要最先配置，一旦设置则全局生效。默认可以采集，设置为false则不采集，可能会影响到控制台的搜索、展示等。
// 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》
QAPM.setProperty(QAPM.PropertyKeyCollectOptionalFields, true);

// 该设置默认为false，设置为true时，QAPM的SDK网络监控数据中的个例数据和腾讯云可观测平台的应用性能监控的全链路数据将会打通，另外有以下几点业务方需要注意：
// 业务网络请求的requestHeader会增加sw8、traceparent相关协议下的value值，如果业务自行组建了全链路监控则可以设置为false
QAPM.setProperty(QAPM.PropertyNeedOpenTrace, false);

// 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 隐私合规 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与一次构建是相互对应关系，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，报错信息可忽略
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);
```

```
//目前上报的域名分为了三个不同的地址，分别是国内站、新加坡站、法兰克福站，根据实际需求选择一个环境上报即可
//国内站：https://app.rumt-zh.com
//新加坡站：https://app.rumt-sg.com
//法兰克福站：https://eu-frankfurt.rumapp.tencentcs.com

QAPM.setProperty(QAPM.PropertyKeyHost,"https://app.rumt-zh.com");
```

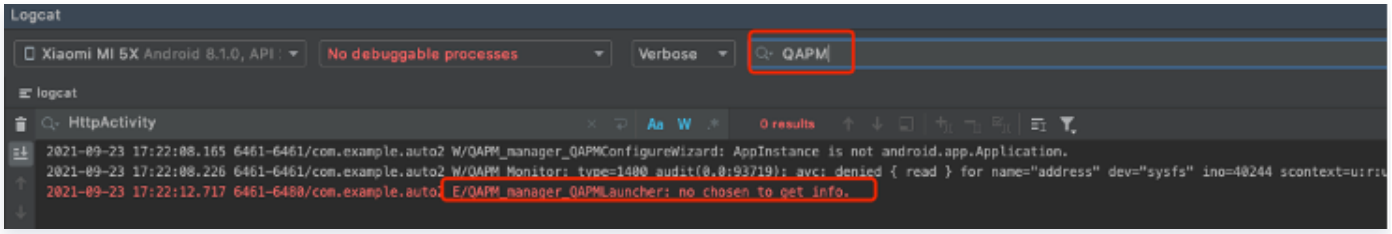
说明：

- AppKey 可参考 步骤三，在 终端性能监控 > 应用管理 > 应用设置 页面获取。
- 崩溃、ANR、启动、JSError 功能是全量上报，其他功能为了帮助您控制初始接入时的费用，默认采取采样上报，采样率为0.1%（千分之一）。验证过程中如果需要确保指定设备不受采样影响，可以将设置好的 userId 或者 deviceId 通过 应用管理 页面添加到白名单里；若需要整体调整指定功能在线上的采样率，请 联系我们 帮您调整，App 将会在下次启动时生效采样率变更。
- 多个进程需要各自初始化 QAPM。

步骤4：接入验证

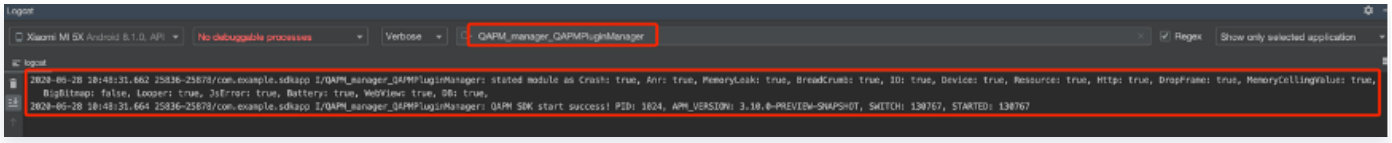
1. 若打印以下日志，代表该用户未被抽样命中，需参考前面步骤将该设备添加到白名单。

   参见 TAG：QAPM_manager_QAPMLauncher



2. 若打印以下日志，则代表初步接入成功，可以验证数据上报/尝试开启高级功能。

   参见 TAG：QAPM_manager_QAPMPluginManager



步骤5：监控功能启用

启动 QAPM，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：

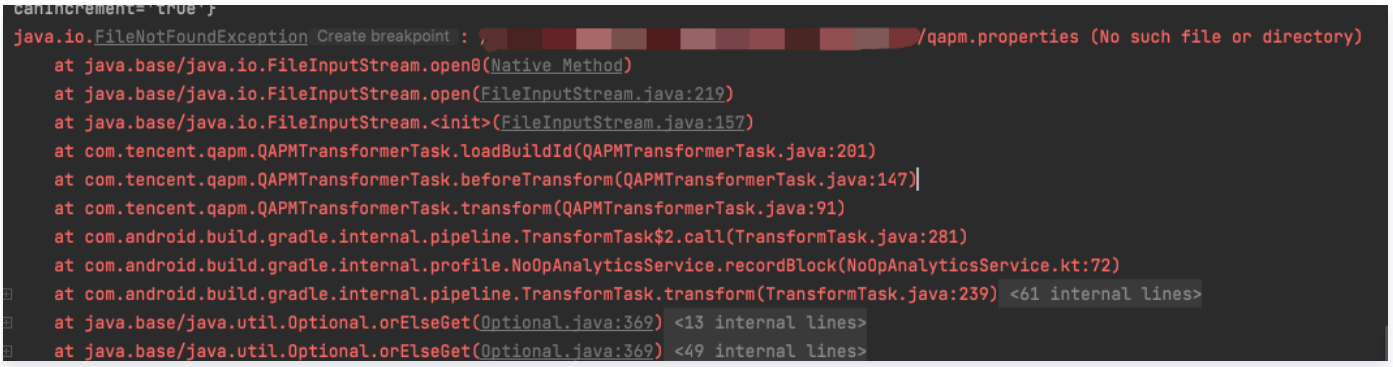
- 使用或运算的方式自定义开启性能模块，如开启 Crash 和 Anr：`beginScene(“Crash&ANR”, QAPM.ModeCrash| QAPM.ModeANR)` 。
- 掉帧监控需先伴随其他功能启动，如 `beginScene(“Crash&DropFrame”, QAPM.ModeCrash| QAPM.ModeDropFrame)` ，之后参考 掉帧率采集 进行埋点统计。
- **为响应工信部 “26号文” 要求，我们依据工信部对性能监控类 SDK 基础功能的定义，将网络监控与用户行为监控划分为扩展业务功能，这意味着为了避免采集网络日志信息、用户操作记录等个人信息，您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
```

其他问题

> **说明：**
> 通过 qapm 插件编译打包 App 时，App 需要一个 uuid 作为构建 id，如果项目目录下存在 qapm.properties 文件，并且文件里 qapm_uuid 属性的值存在，该值将被作为构建 id，否则插件会随机生成一个构建 id。

qapm-plugin 2.39及之前版本在编译 App 的过程中会报 IO 错误：java.io.FileNotFoundException，qapm.properties (No such file or directory)。



该报错仅在编译期间产生，不会影响 App 运行。

功能配置
原生层网络监控、大模型 SSE 监控以及默认拨测监控

最近更新时间：2025-04-16 14:27:12

前提条件

- 在 app 级别的 build.gradle 中配置了 qapm-plugin 插件，请参见 集成和初始化 。
- 目前只支持 okhttp3监控。okhttp3还依赖 okio 1.14.0以上的库版本。

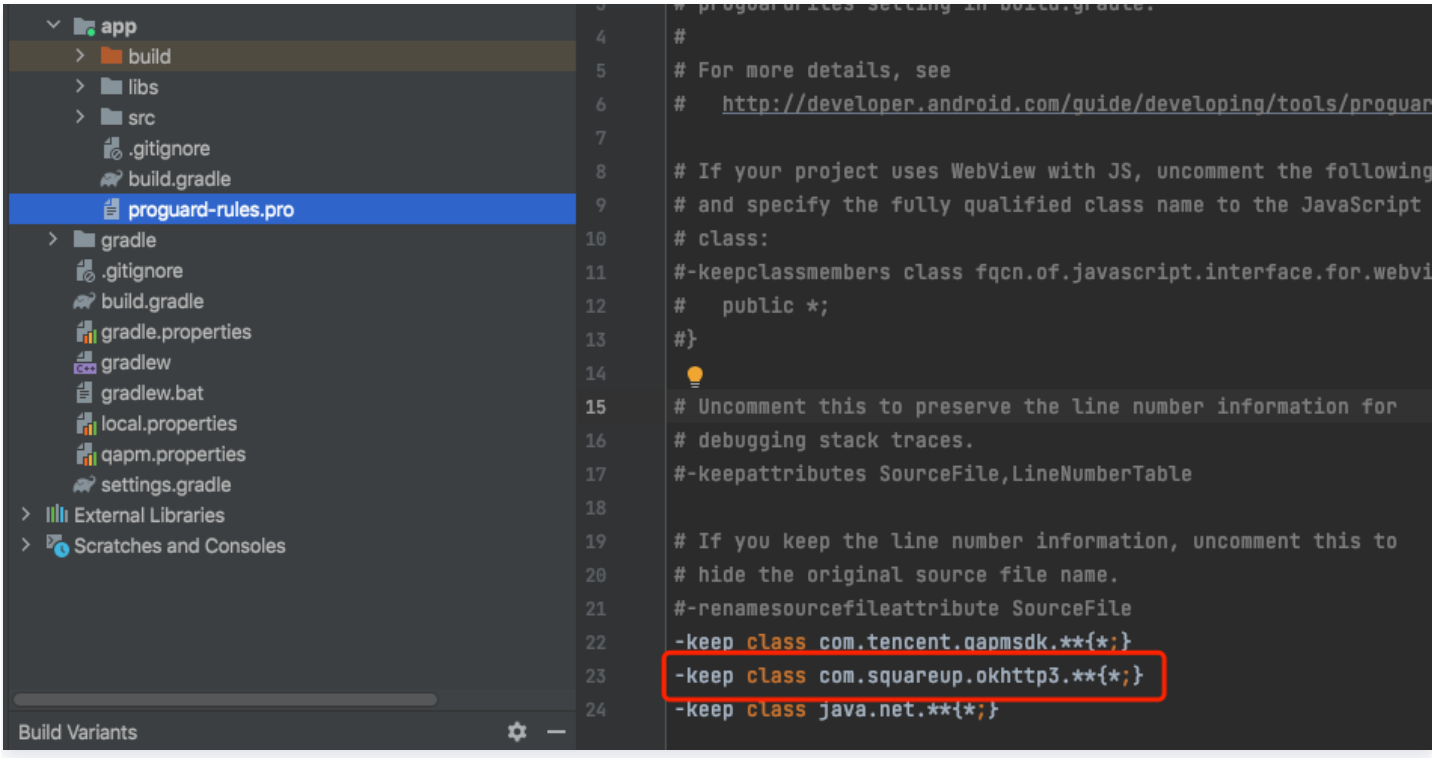
开启功能

原生层网络监控以及大模型 SSE 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。

配置步骤

在 app 目录的 proguard-rules.pro 文件里增加混淆规则，防止 okttp3的代码被混淆。

```
-keep class com.squareup.okhttp3.**{*;}
```



相关接口

```
/**
 * 设置网络请求的响应 header 黑名单，在黑名单中的 header key 将不会采集上报
 */
public static final int PropertyKeyHttpBlackHeader = 313;
/**
 * 设置域名白名单，如果设置了，则只有白名单中的域名请求才监控
 */
public static final int PropertyKeyHttpWhiteDomain = 314;
/**
 * 设置域名黑名单，如果设置了，在黑名单中的域名将不会监控
 */
public static final int PropertyKeyHttpBlackDomain = 315;
```

代码示例

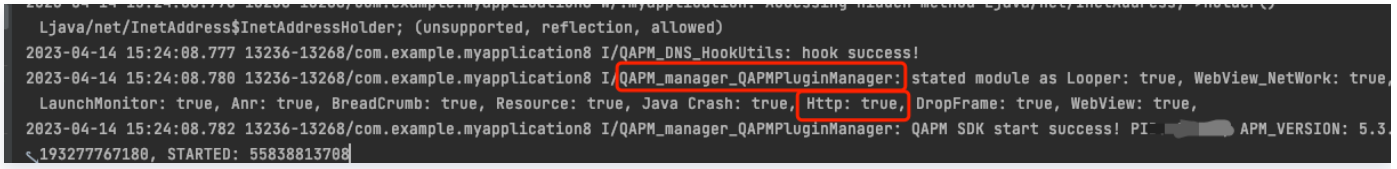
```
/**
 * 设置网络请求的响应 header 黑名单，在黑名单中的 header key 将不会采集上报
 */
List<String> blackHeader = new ArrayList<>();
blackHeader.add("");
blackHeader.add("x-nws-log-uuid");
QAPM.setProperty(QAPM.PropertyKeyHttpBlackHeader, blackHeader);

/**
 * 设置域名黑名单，如果设置了，在黑名单中的域名将不会监控
 */
List<String> blackDomain = new ArrayList<>();
blackDomain.add("21.54.255.67");
blackDomain.add("9.134.164.160");
QAPM.setProperty(QAPM.PropertyKeyHttpBlackDomain, blackDomain);

/**
 * 设置域名白名单，如果设置了，则只有白名单中的域名请求才监控
 */
List<String> WhiteDomain = new ArrayList<>();
WhiteDomain.add("21.54.255.67");
WhiteDomain.add("9.134.164.160");
```

```
QAPM.setProperty(QAPM.PropertyKeyHttpWhiteDomain, WhiteDomain);
```

> **注意：**
> 原生网络监控默认监控所有 url，可通过上面的黑白名单接口来决定是否进行过滤，在同时设置了域名白名单和域名黑名单接口的时候，只有域名白名单的设置才会生效，建议优先使用白名单设置。

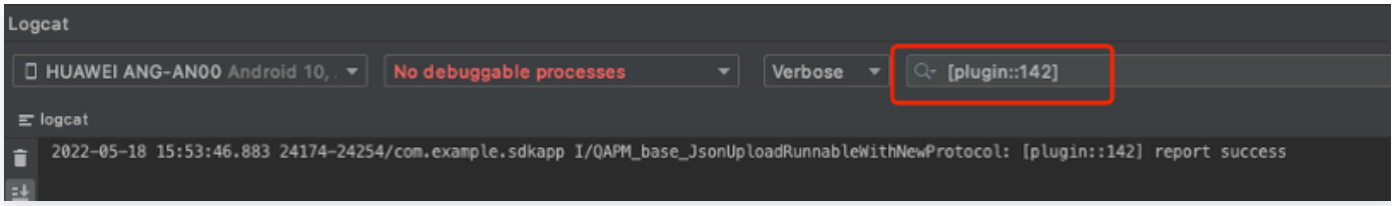
校验功能是否正常

- 检索 TAG:QAPM_manager_QAPMPluginManager



- 检索 TAG: [plugin::142]

  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：



> **注意：**
> - 需要使用 qapmplugin 插件进行插桩才可用，否则无效；大模型 SSE 监控需要发起对应的 SSE 监控请求，否则无效。
> - SDK 只负责抓取网络请求的相关信息，问题数据由后台分析，如慢请求（请求时间大于 xxs），网络错误（请求响应码 > 400）。
> - 数据在 终端性能监控 > 网络 > 慢请求和错误请求列表 中查看。

WebView 、JsError、Web 网络监控

最近更新时间：2024-06-12 15:09:01

开启功能

初始化需要开启 WebView、JsError、Web 网络监控，可通过 ModeStable 直接开启，代码如下：

```
QAPM.beginScene(QAPM.SCENE_ALL, 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.addJavascriptInterface(QAPMJavaScriptBridge.getInstance(),"QAPMA
ndroidJsBridge");
```

- 在 WebView 的 shouldInterceptRequest 代码里加入以下方法，用于拦截 web-sdk 并改用本地 SDK 资源，**请确保在该回调中的最早地方调用以下代码。**
  - **如果是 x5 请使用以下代码：**

```
@Overridepublic
public WebResourceResponse shouldInterceptRequest(WebView webView,
String s) {
    Object response
=QAPMJavaScriptBridge.getInstance().shouldInterceptRequestWithX5(s);

    if (response != null) {
        return (WebResourceResponse)response;
     }
     return super.shouldInterceptRequest(webView,s);
     }
```

- 如果是原生 WebView 请使用以下代码：

```
@Overridepublic
public WebResourceResponse shouldInterceptRequest(WebViewwebView,
String s) {
    WebResourceResponse response
=QAPMJavaScriptBridge.getInstance().shouldInterceptRequest(s);
    if (response != null) {
        return response;
    }
    return super.shouldInterceptRequest(webView, s);
  }
```

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

```
webView.setWebViewClient(new WebViewClient(){
    @Override
    public void onPageFinished (WebView view, String url) {
        super.onPageFinished(view, url);
        QAPMJavaScriptBridge.getInstance().initFileJS(view);
    }
});
```

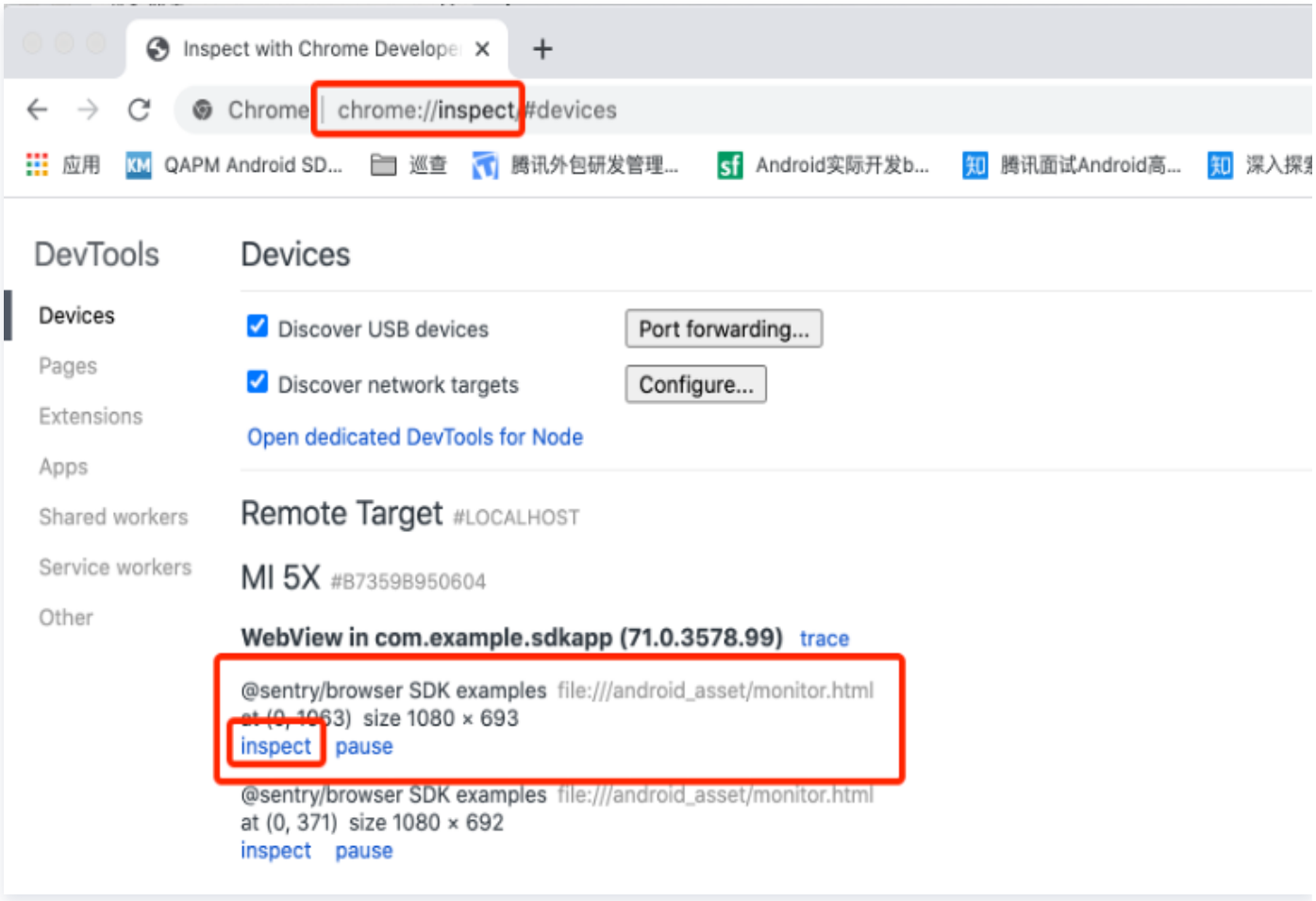
校验功能是否正常

原生 WebView、JsError 监控：

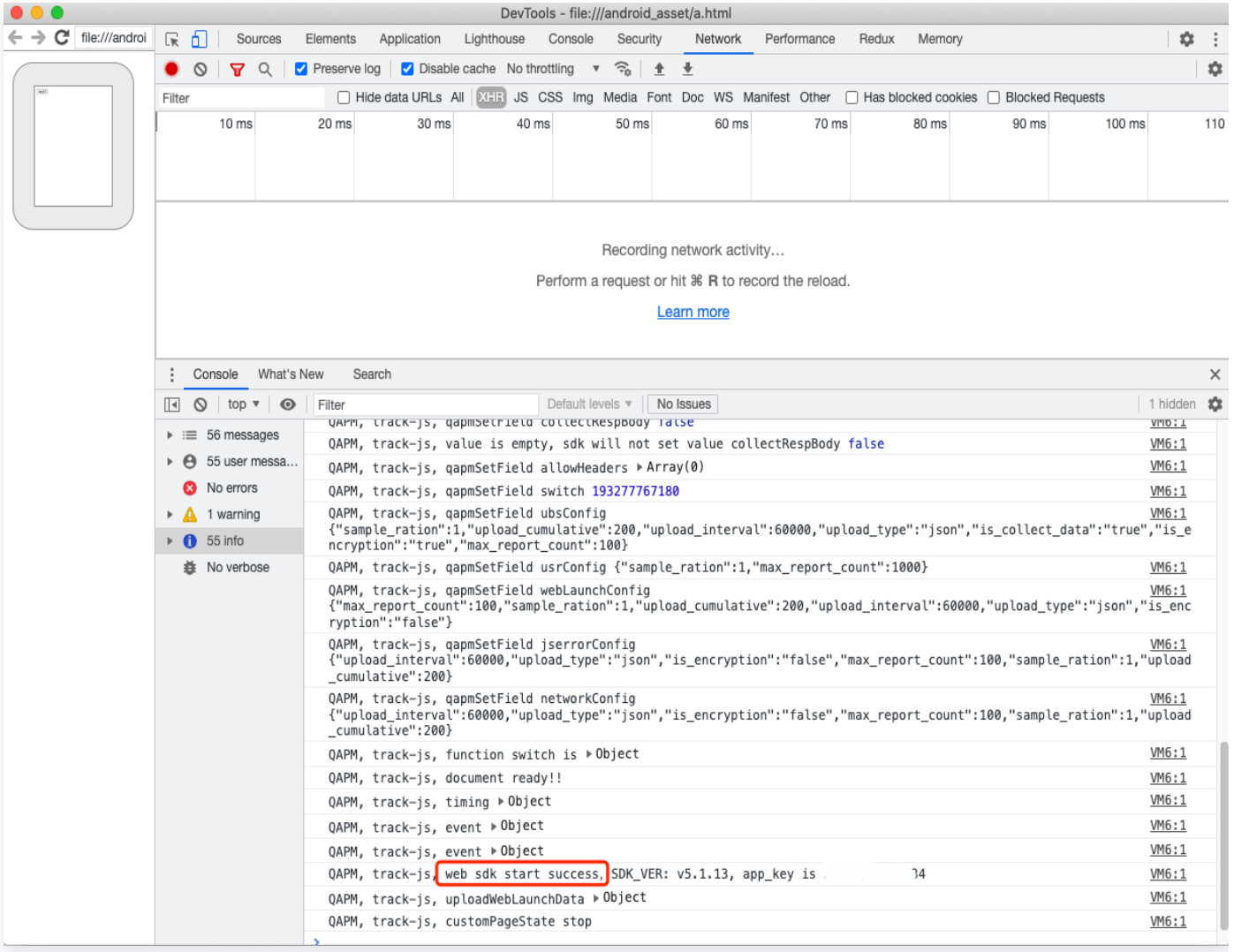
1. 代码中加入以下代码（用于远程调试）。

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

2. 打开谷歌浏览器，地址栏输入 chrome://inspect，在出现的设备中单击 inspect。

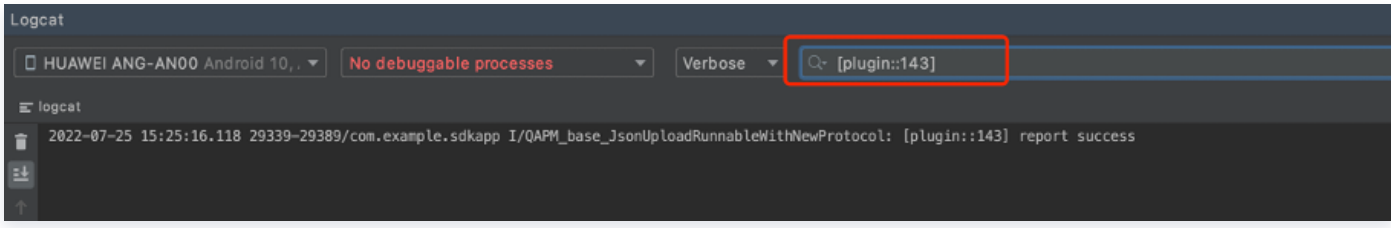


3. 进入后找到 Console 模块查询日志，如出现 `web start success ,vxxx`，则代表 WebSDK 注入成功。



4. 检测各个功能是否上报正常，以 JsError 上报为例，如下：

   检索 TAG: [plugin::143]。每次触发 JsError 错误，如打印以下日志，则代表 JsError 数据上报成功。



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：

1. 如需查看 WebView 监控是否正常需要通过 chrome 等浏览器调试查看。
2. 页面加载只有页面加载时长大于3.5s才可在问题个例详情里查看。
3. 网络请求只有在网络错误和网络慢时才会上报。
4. 数据在 终端性能监控 > Webview > 慢加载和 JS 错误问题列表 中查看。

Crash、ANR 监控

最近更新时间：2024-12-09 18:48:02

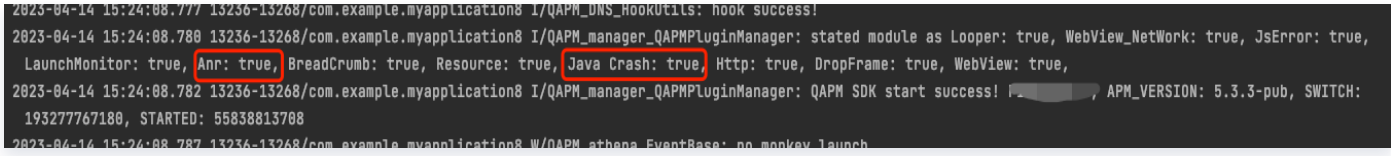
开启功能

初始化需要开启 Crash、ANR 监控，该监控会默认监控 Crash 和 ANR 信息。

```
// ModeStable模式默认包含了Crash、ANR监控
QAPM.beginScene(QAPM.SCENE_ALL, QAPM.ModeStable);
```

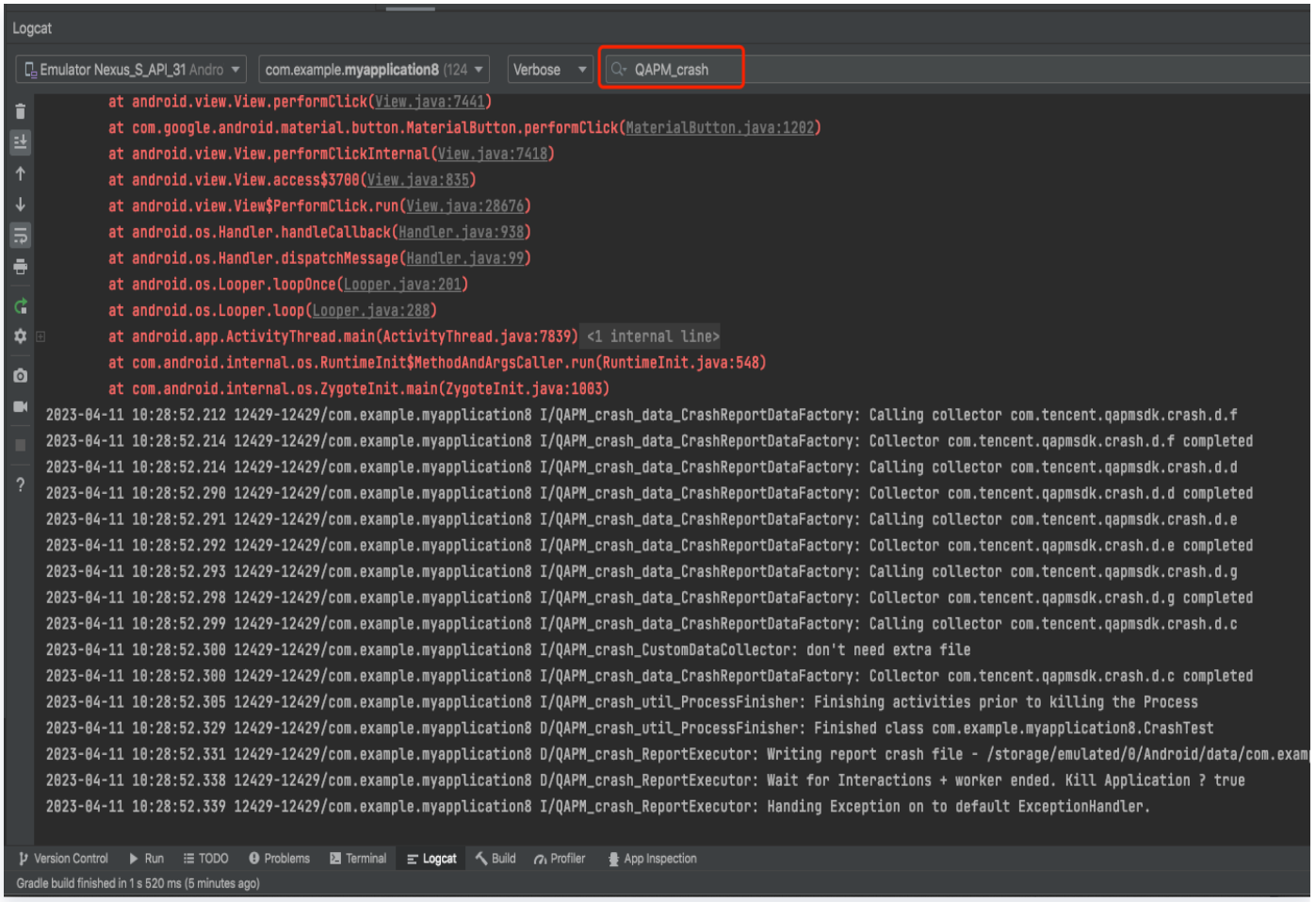
校验功能是否正常

- 检索 TAG: QAPM_manager_QAPMPluginManager



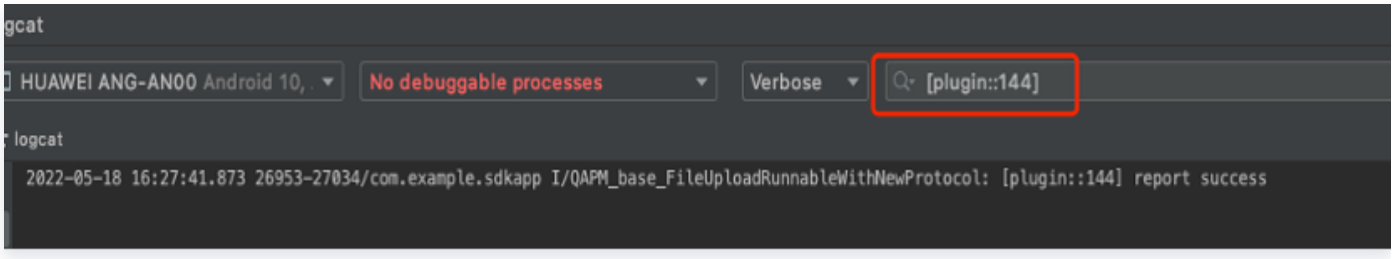
- 检索 TAG: QAPM_crash

  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: plugin::144

  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：



- 其他 crash 检索 TAG 分别如下：
  - ANR：[plugin::140]。
  - NativeCrash：[plugin::146]。

> **说明：**
> - 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。

- Crash 数据在 终端性能监控 > 崩溃分析 中查看。
- ANR 数据在 终端性能监控 > ANR 分析 中查看。

卡顿、帧率监控

最近更新时间：2024-06-12 15:09:01

前提条件

已完成 集成与初始化 。

功能配置

开启监控

初始化需要开启卡顿监控。卡顿无需埋点，而掉帧率需要额外埋点，建议打点在滑动列表上，如（ListView、GridView、RecyclerView 等）。

掉帧率埋点

- 在每次滑动前调用 QAPM.beginScene("xxx滑动", QAPM.ModeDropFrame);
- 在滑动结束后调用 QAPM.endScene("xxx滑动", QAPM.ModeDropFrame);

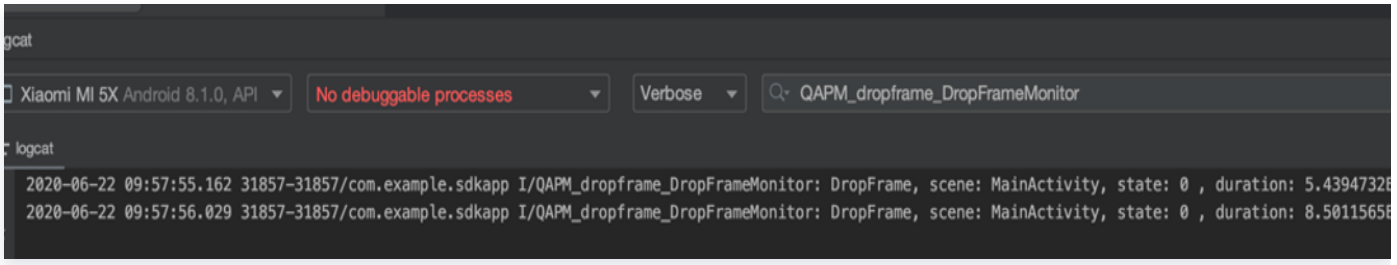
一般可以通过重写滑动组件 onScrollStateChanged 方法来实现，示例如下：

```
@Override
public void onScrollStateChanged(AbsListView view, int scrollState) {
    if (scrollState == AbsListView.OnScrollListener.SCROLL_STATE_IDLE) {
        QAPM.endScene("xxx滑动", QAPM.ModeDropFrame); //xxx滑动名称由您自定义，您可以填写任意名称
    } else {
        QAPM.beginScene("xxx滑动", QAPM.ModeDropFrame);//xxx滑动名称由您自定义，您可以填写任意名称
    }
```

校验功能是否正常

- 检索 TAG：QAPM_dropframe_DropFrameMonitor

  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：

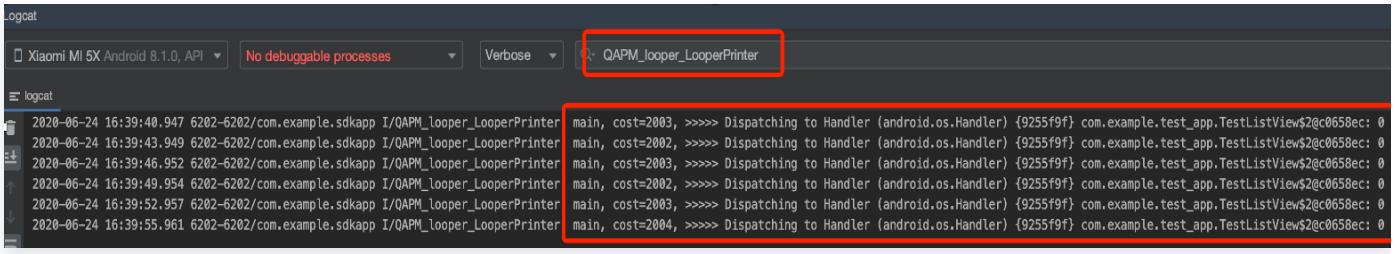


- 检索 TAG: [plugin::101]

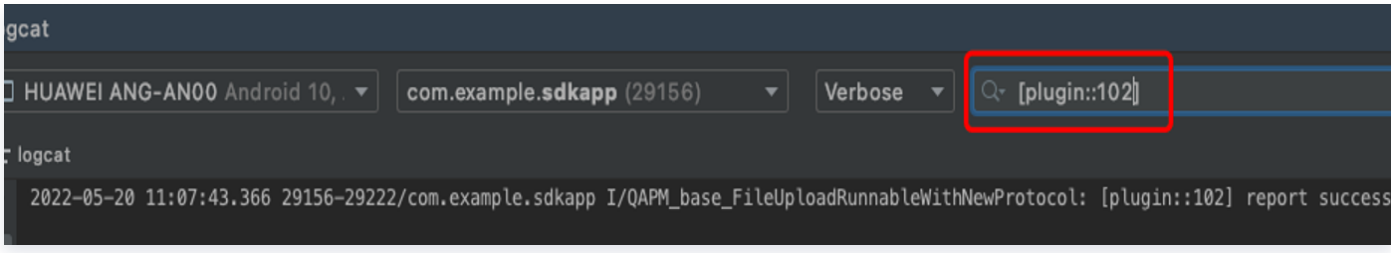
  存在 App 本地数据库的掉帧数据将会上报，打印以下数据，代表上报成功：

- 检索 TAG: QAPM_looper_LooperPrinter

  如打印以下日志，则代表卡顿监控正常：



  如打印以下日志，则代表卡顿上报正常：



说明：

- 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。
- 掉帧数据在 终端性能监控 > 卡慢分析 中查看。
- 卡慢数据在 终端性能监控 > 卡慢问题列表 中查看。

启动监控

最近更新时间：2024-06-12 15:09:01

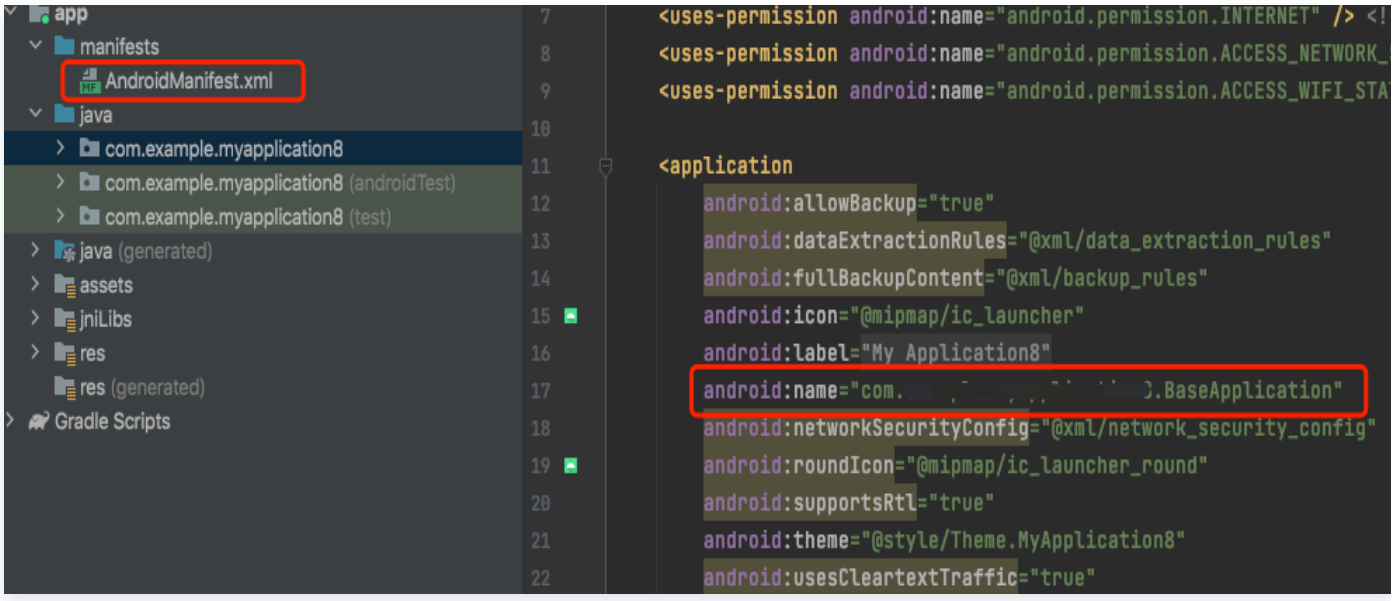
启动监控需要使用 qapmplugin 插件在编译期间进行插桩操作，默认插桩点为 Application 与 Activity 的各个生命周期。在 App SDK 统计默认的启动耗时为 Application 的 attachBaseContext 到第一个 Activity 的 onResume 结束。

前提条件

在 App 级别的 build.gradle 中配置了 qapm-plugin 插件。

配置步骤

1. 需要手动添加一个 Application 的子类，例如 BaseApplication（名称不作要求，子类可以不用实现任何方法，可以不用添加任何属性）。
2. 在 AndroidManifest.xml 文件的 application 的节点添加 android:name 属性，属性的 value为“包名+Application 子类的类名”。



额外打点

- 如果想统计启动区间内的某些方法的耗时，则需要额外的打点，示例如下：

```
public class BaseApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        b();
        try {
            Thread.sleep( millis: 2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        c();
    }

    private void b() {
        QAPM.beginScene(StageConstant.QAPM_APPLAUNCH,  extraInfo: "b方法名", QAPM.ModeResource);
        /**
         * 业务逻辑
         */
        QAPM.endScene(StageConstant.QAPM_APPLAUNCH,  extraInfo: "b方法名", QAPM.ModeResource);
    }

    private void c() {
        QAPM.beginScene(StageConstant.QAPM_APPLAUNCH,  extraInfo: "c方法名", QAPM.ModeResource);
        /**
         * 业务逻辑
         */
        QAPM.endScene(StageConstant.QAPM_APPLAUNCH,  extraInfo: "c方法名", QAPM.ModeResource);
    }
}
```

参见代码：

```
QAPM.beginScene(StageConstant.QAPM_APPLAUNCH, "xxx方法名",
QAPM.ModeResource);
/**业务逻辑*/
QAPM.endScene(StageConstant.QAPM_APPLAUNCH, "xxx方法名",
QAPM.ModeResource);
```

- 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：

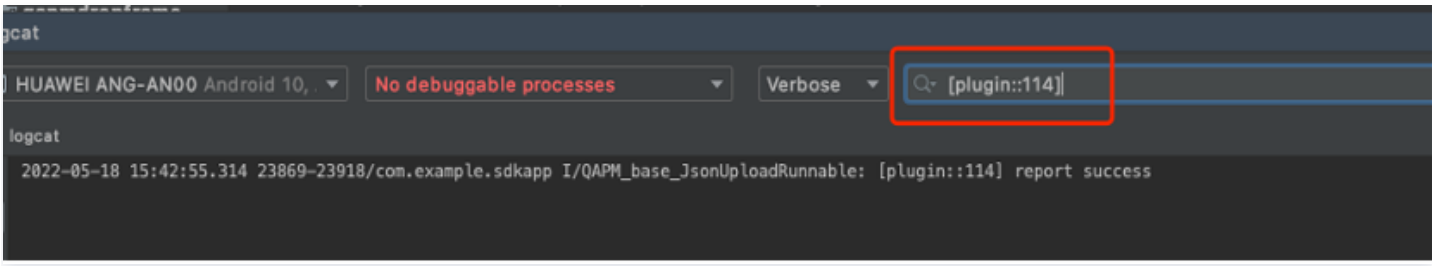
```
/**
 * 需要自定义结束点的用户需要在onResume之后的20s内，否则以APM的结束点为准
 */
QAPM.endScene(StageConstant.QAPM_APPLAUNCH, QAPM.ModeResource);
```

参见代码：

```
/**
 * 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]



注意：

- 需要使用 qapmplugin 插件进行插桩才可用，并且需要手动增加一个 Application 的子类。否则无效。
- 启动总耗时大于2.5s才会上报个例数据。
- 启动的问题数据可以在 终端性能监控 > 启动 > 问题列表 查看。

用户行为监控

最近更新时间：2024-06-12 15:23:01

开启功能

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。

```
// ModeStable模式默认包含了用户行为监控
QAPM.beginScene(QAPM.SCENE_ALL, QAPM.ModeStable);
```

自定义用户行为事件

QAPM 提供了相关接口，可用于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，对应的key的值只能是d1~d30/info1~info10范围的值
 * @param values     事件关联的一系列数值类属性，该参数可为空，对应的key的值只能是
v1~v30范围的值
 * @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)
```

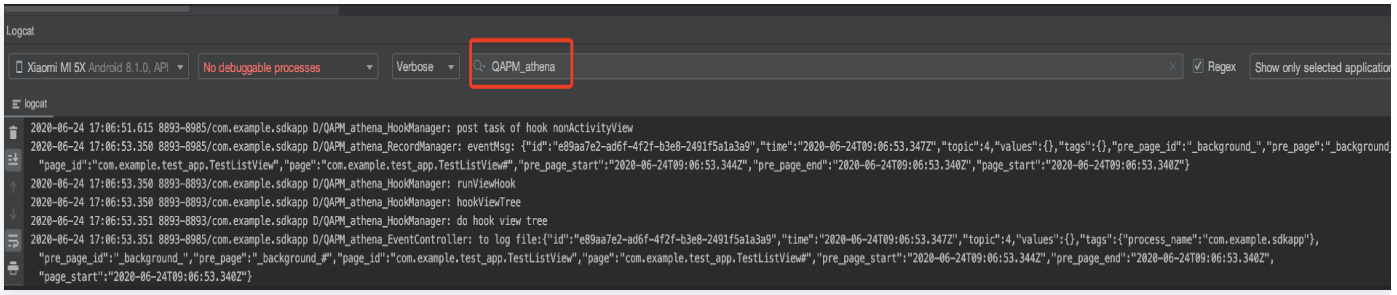
具体示例：

```
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");
```

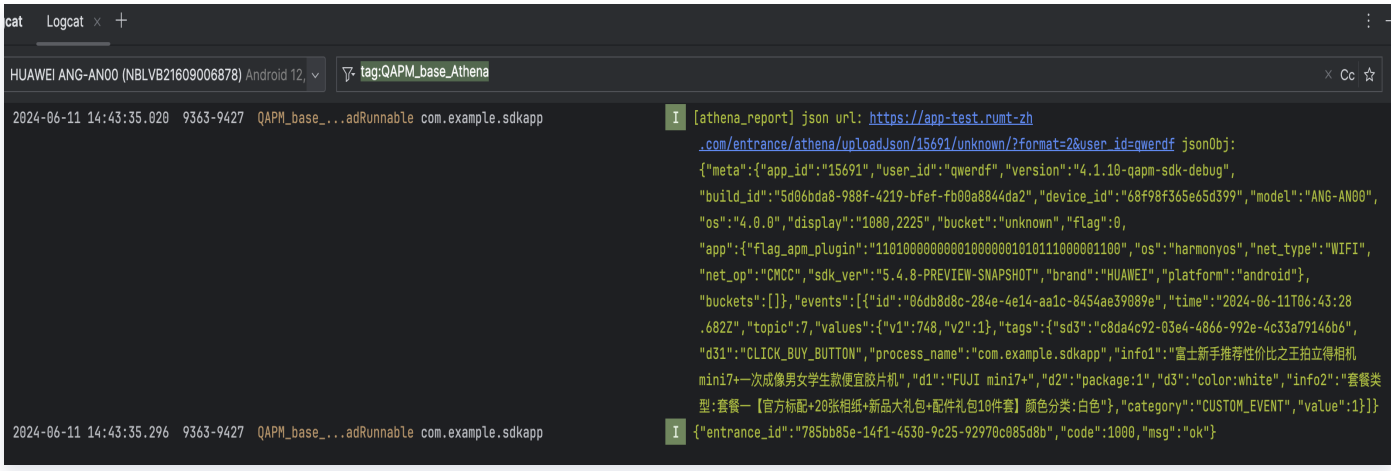
```
Map<String, Long> values = new HashMap<>();
values.put("v1", 748L);
values.put("v2", 1L);

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

校验功能是否正常

- 检索 TAG: QAPM_athena。如打印以下日志，则代表用户行为功能开启正常。



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



Webview 用户行为监控

最近更新时间：2024-06-28 17:59:51

开启功能

初始化需要开启 Webview 用户行为监控，该监控默认会收集 Web 页面中用户的点击、页面切换等事件，也可通过 ModeStable 实现快捷开启：

```
// ModeStable模式默认包含了用户行为监控
QAPM.beginScene(QAPM.SCENE_ALL, QAPM.ModeStable);
```

自定义用户行为事件

在 Web 页面代码中需要使用自定义事件的位置（例如点击“购买基金”按钮的处理函数中），编写如下代码：

```
window.QAPM.customEvent(category, tags, values)
参数说明：
（1）category：事件名称，字符串，值的长度不超过100
（2）tags：事件辅助信息，json格式，key只能填写d1~d30和info1~info10，它们的值都是字符串，其中d1~d30的值的长度不超过100，info1~info10的值的长度不超过1024，超长的部分会被截断
（3）values：事件辅助信息，json格式，key只能填写v1~v30，它们的值都是int
```

接口调用实例：

```
var event = {
    "category": "CLICK_BUY_BUTTON",
    "values": {
        "v1": 748,
        "v2": 1
    },
    "tags": {
        "d1": "FUJI mini7+",
        "d2": "package:1",
        "d3": "color:white",
        "info1": "Fuji novice recommends the king of cost-effective instant camera mini7+ one-time imaging cheap film machine for male and female students",
        "info2": "Package type: Package 1 [Official standard + 20 pieces of photo paper + new product gift package + 10-piece accessory gift
```

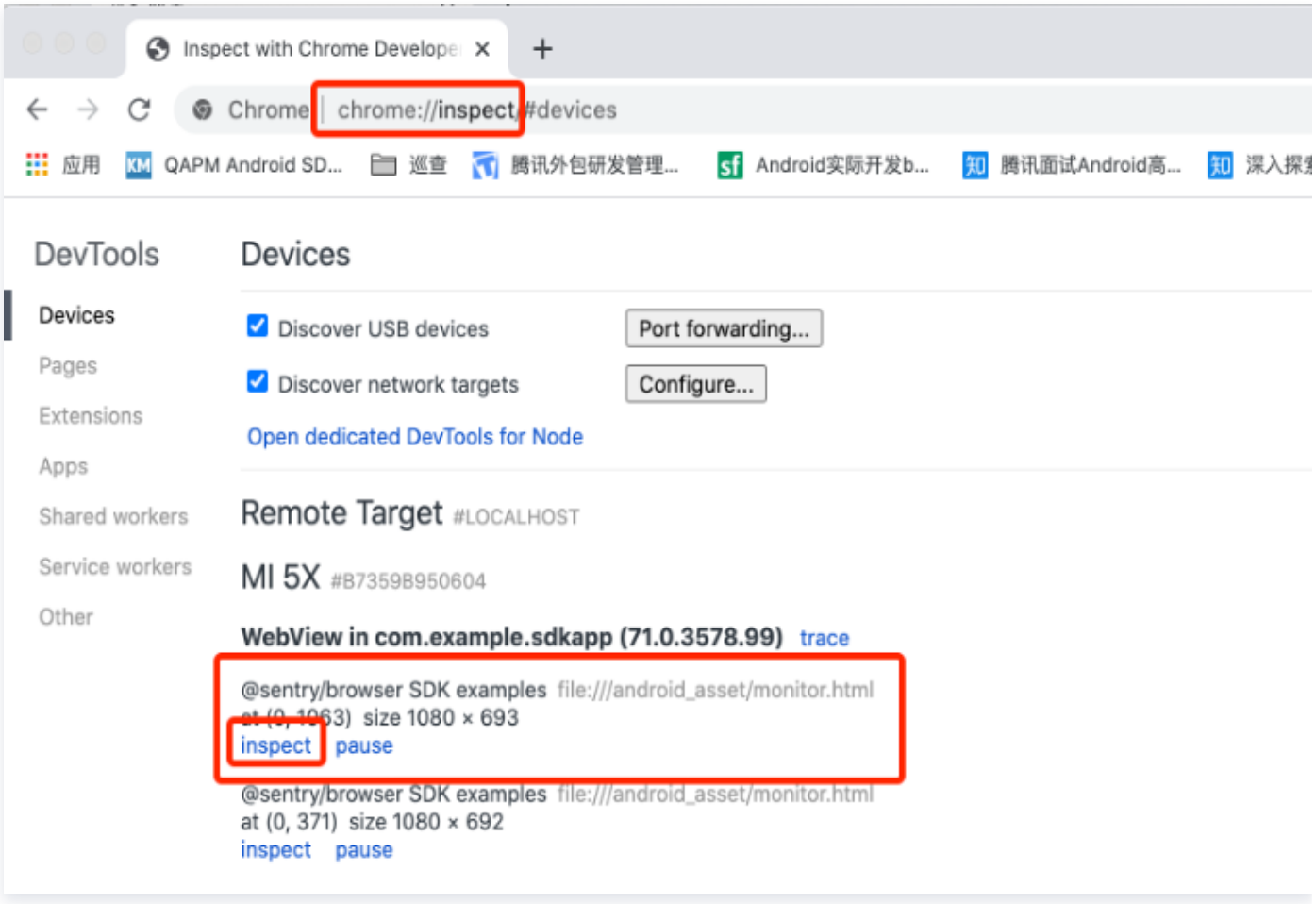
```
package] Color classification: White"
    }
};
window.QAPM && typeof window.QAPM.customEvent==="function" &&
window.QAPM.customEvent(event.category, event.tags, event.values);
```

校验功能是否正常

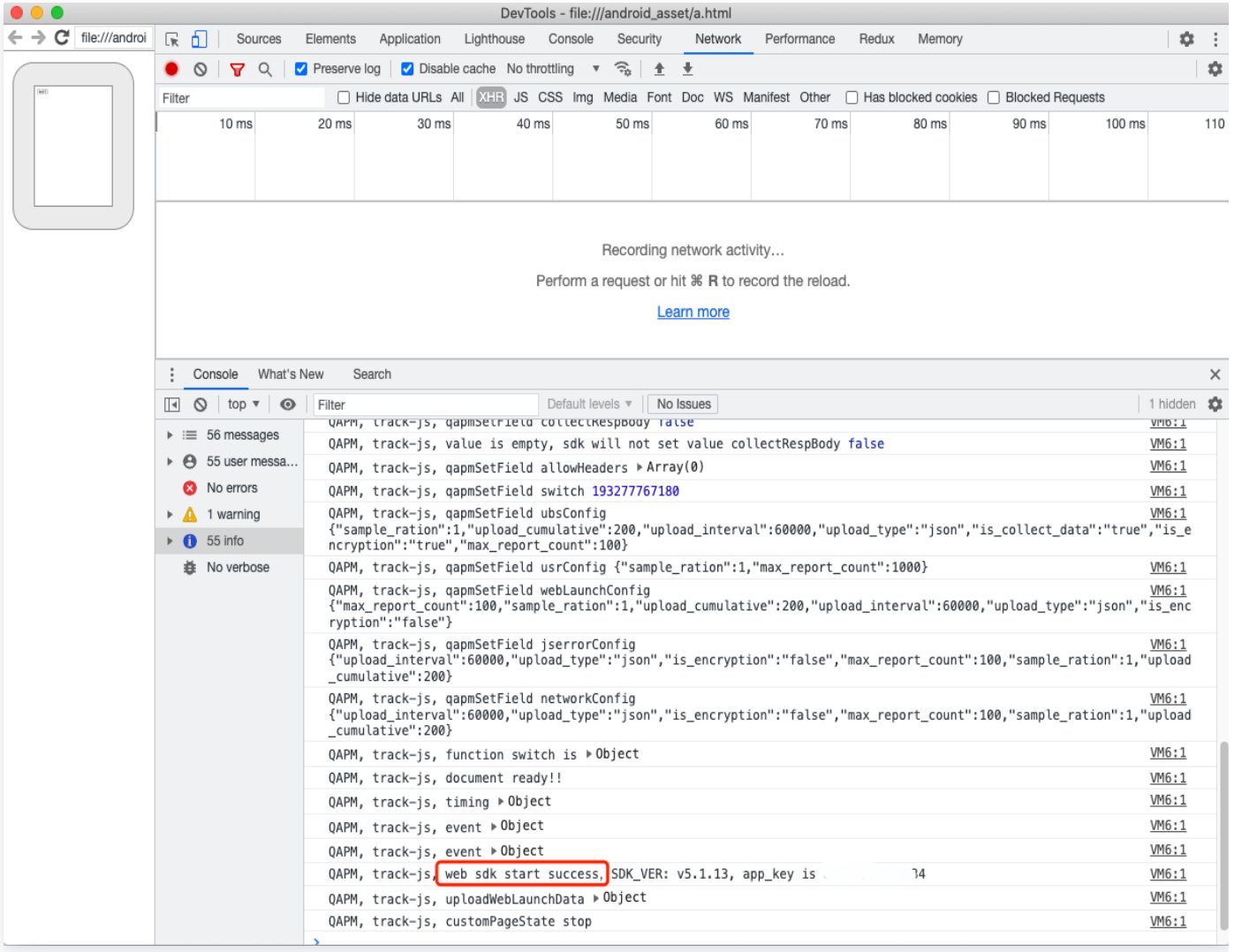
1. 代码中加入以下代码（用于远程调试）：

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

2. 打开谷歌浏览器，地址栏输入 `chrome://inspect`，在出现的设备中单击 **inspect**。

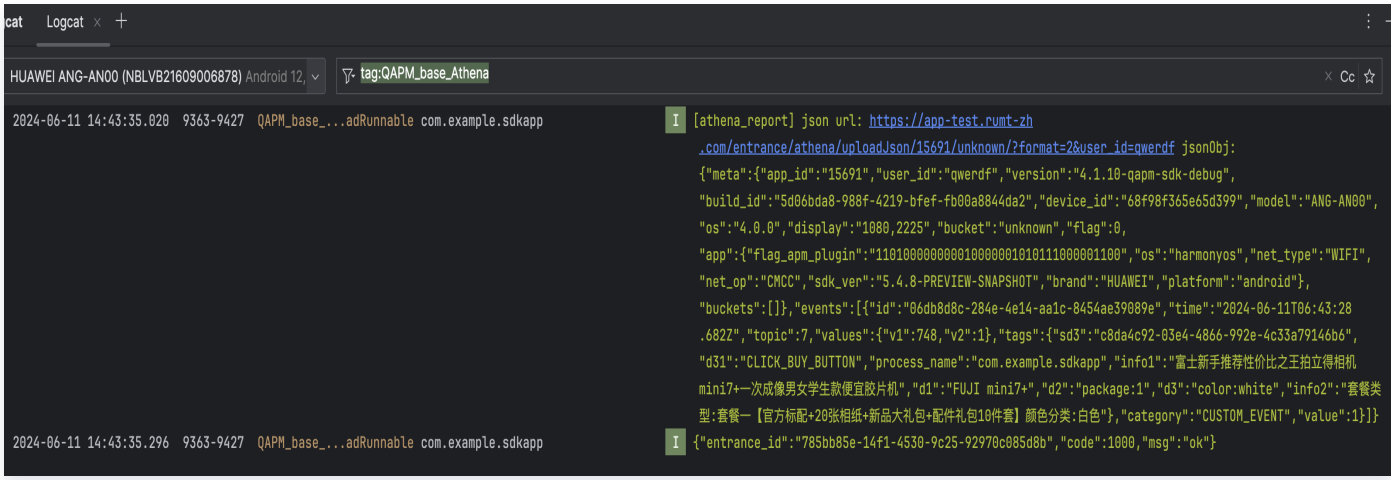


3. 进入后找到 Console 模块查询日志，如出现 `web start success ,vxxx`，则代表 WebSDK 注入成功。



4. 检测 Webview 用户行为/自定义事件采集功能是否上报正常，如下：

   检索 TAG: QAPM_base_Athena。如打印以下日志，请注意需要检索确认 platform 是 Web，则代表 Webview 用户行为上报正常。



隐私合规（重要）

最近更新时间：2024-06-12 16:19:51

标识符自助生成

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 集成和初始化>步骤三
}
```

标识符生成

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，若您本身不会获取 “IMEI”，“IMEI” 字段可缺省。

> **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。

首次启动数据自助上报

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 ) ;
```

标识符变更获取

当前监管要求 SDK 不允许直接或间接采集 IMEI 等信息，我们只能通过用户自行传入标识符的方法去区分不同的设备。

> ⚠ **注意：**
> 仅使用5.0.7以前版本 SDK 的用户需要关注。

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。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。

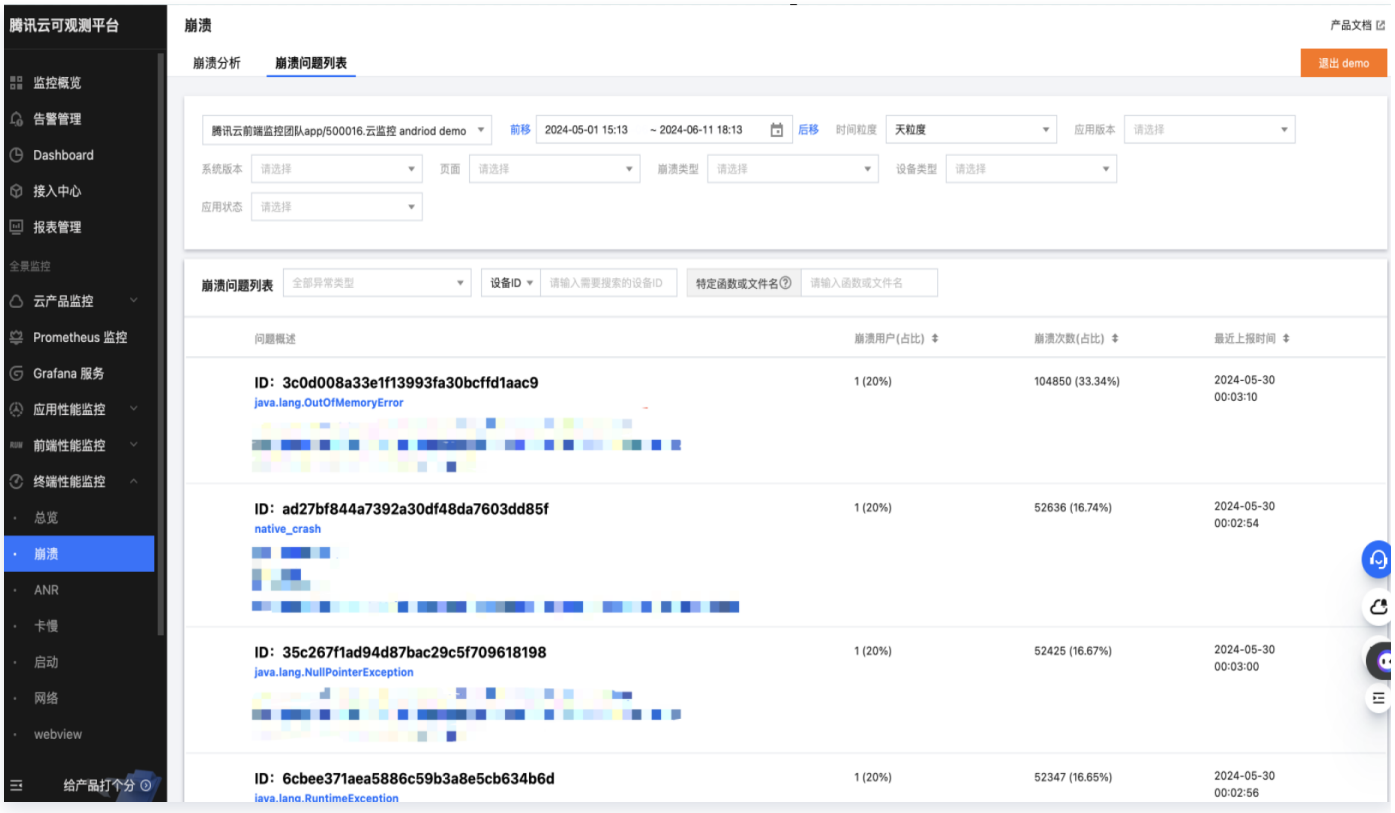
符号表配置

最近更新时间：2024-06-12 16:19:51

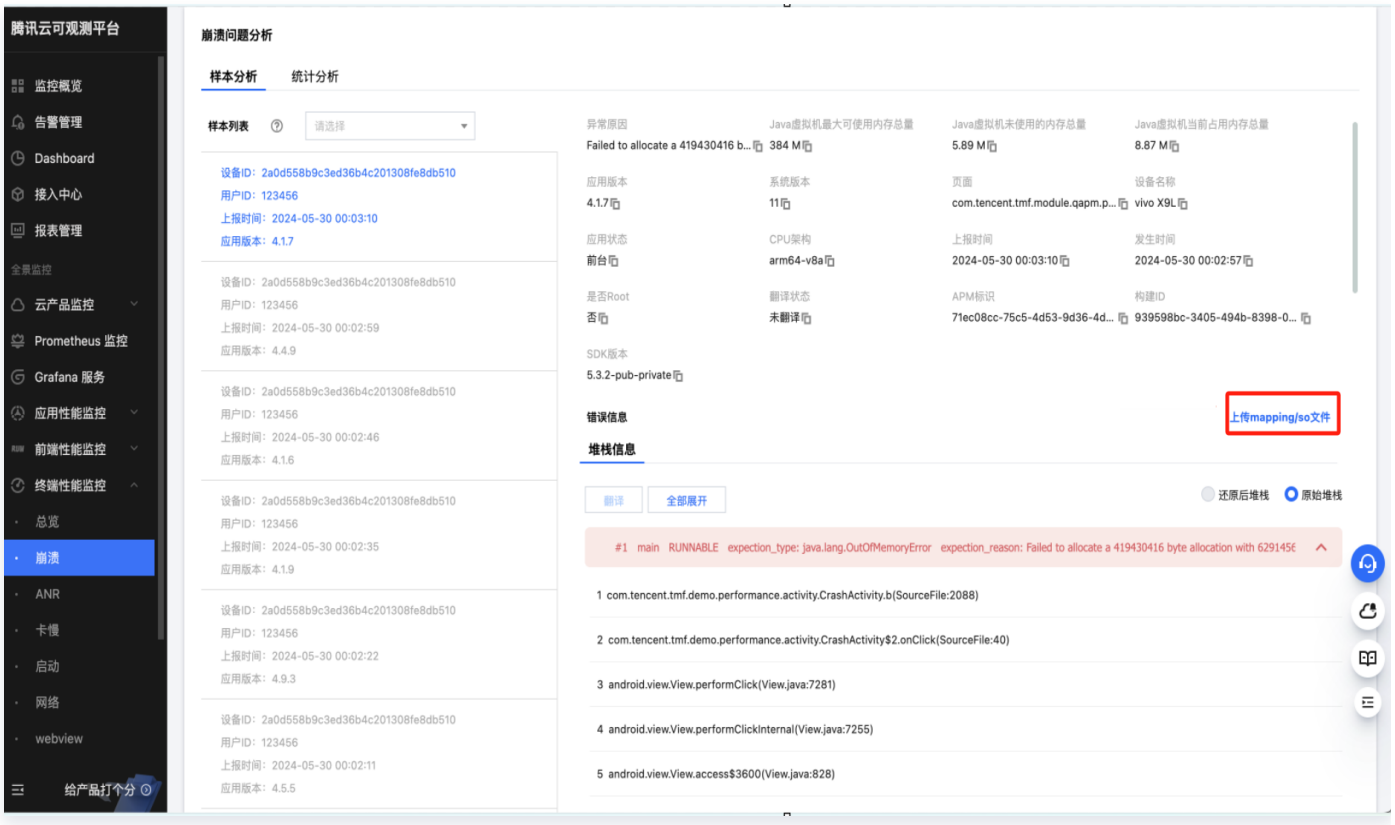
利用已上报的数据个例，根据个例页面的构建 ID 找到对应的符号表后完成上传，需要注意对已上报的问题样本不会再自动做重新翻译处理，针对这些样本可以点击手动翻译以实现反混淆。

手动上传

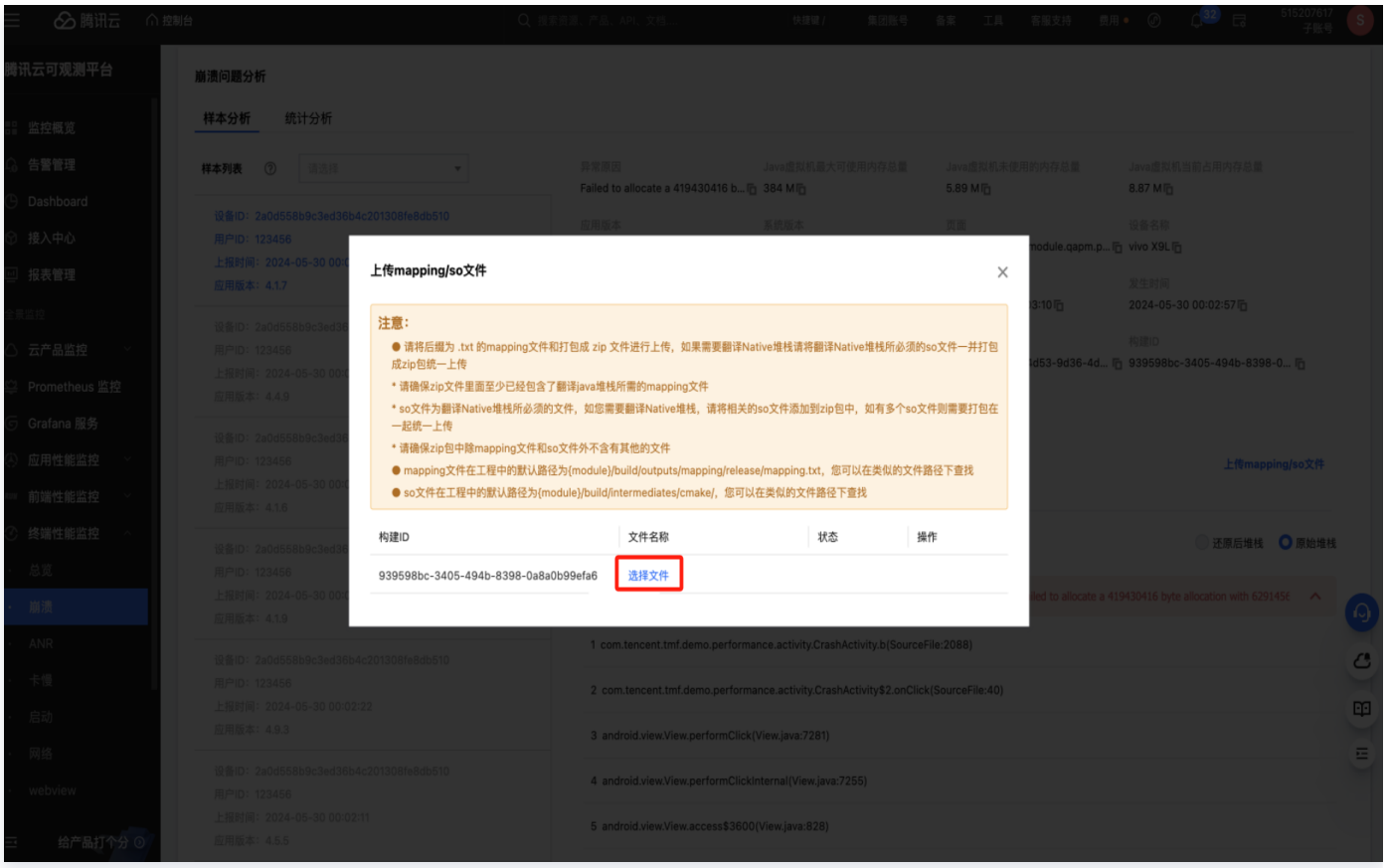
- 点击 问题列表 中任意 issue 进入问题详情。



- 在问题详情页面中点击上传符号表。



- 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。


腾讯云可观测平台
崩溃问题分析
样本分析
统计分析
上传mapping/so文件
注意：
请将后缀为 .txt 的mapping文件和打包成 zip 文件进行上传，如果需要翻译Native堆栈请将翻译Native堆栈所必须的so文件一并打包成zip包统一上传
* 请确保zip文件里面至少已经包含了翻译java堆栈所需的mapping文件
* so文件为翻译Native堆栈所必须的文件，如您需要翻译Native堆栈，请将相关的so文件添加到zip包中，如有多个so文件则需要打包在一起统一上传
* 请确保zip包中除mapping文件和so文件外不含有其他的文件
mapping文件在工程中的默认路径为{module}/build/outputs/mapping/release/mapping.txt，您可以在类似的文件路径下查找
so文件在工程中的默认路径为{module}/build/intermediates/cmake/，您可以在类似的文件路径下查找
构建ID
文件名称
状态
操作
939598bc-3405-494b-8398-0a8a0b99efa6
选择文件
1 com.tencent.tmf.demo.performance.activity.CrashActivity.b(SourceFile:2088)
2 com.tencent.tmf.demo.performance.activity.CrashActivity$2.onClick(SourceFile:40)
3 android.view.View.performClick(View.java:7281)
4 android.view.View.performClickInternal(View.java:7255)
5 android.view.View.access$3600(View.java:828)

iOS 应用场景
集成和初始化

最近更新时间：2025-01-08 14:13:52

本文指导您使用 iOS SDK 的集成与初始化。

> **说明：**
> 隐私合规相关配置请查看 合规使用指南 。

操作步骤

步骤1：SDK 集成

支持通过 CocoaPods 集成和手动集成两种方式集成 SDK。

CocoaPods 集成

在 podfile 文件中增加如下操作，然后执行 `pod install` 指令。

```
pod 'QAPM',:source => 'https://github.com/TencentCloud/QAPM-iOS-CocoaPods.git'
```

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)
   - libz.dylib (libz.tbd)
   - libresolv.tbd
4. 在工程的 Other Linker Flags 中添加 `-ObjC` 参数。
5. 导入配置文件：
   - 5.3.5之前的版本将 framework 里面的 `js_sdk.js` 文件导入到工程根目录。
   - 5.3.5及以后的版本将 framework 里面的 `QAPMResourceFile.bundle` 文件导入到工程根目录。

> **说明：**
> - iOS SDK 最低兼容系统版本 iOS 8.0。
> - 在正确获取到 appkey 后可先体验 Demo，快速触发终端性能监控的各功能数据上报。

- 常见接入 FAQ 请参见 常见问题 。

步骤2：Web 端环境配置

登录 腾讯云可观测控制台 ，在**终端性能监控**页面，选择**应用管理 > 应用设置**，进入**应用设置**后，获取 Appkey（上报 ID）。

步骤3：SDK 初始化

1. 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。
2. 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，该设置需要最先配置，一旦设置则全局生效。默认设置为YES，代表可以采集；设置为NO则不采集，可能会影响到前端的搜索、展示等。
//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》:https://cloud.tencent.com/document/product/683/70827
[QAPMConfig getInstance].collectOptionalFields = YES;

// 该设置默认为NO，若需要开启全链路监控功能请设置为YES，QAPM的SDK网络监控数据中的个例数据和腾讯云可观测平台的应用性能监控的全链路数据将会打通，另外业务方需要注意：
// 业务网络请求的requestHeader会增加sw8、traceparent相关协议下的value值，如果业务自行组建了全链路监控则可以设置NO
[QAPMConfig getInstance].isOpenTrace = NO;

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

//目前上报的域名分为了三个不同的地址，分别是国内站、新加坡站、法兰克福站，根据实际需求选择一个环境上报即可
//国内站：https://app.rumt-zh.com
//新加坡站：https://app.rumt-sg.com
//法兰克福站：https://eu-frankfurt.rumapp.tencentcs.com
[QAPMConfig getInstance].host = @"https://app.rumt-zh.com";

[QAPMConfig getInstance].customerAppVersion = @"设置app自定义版本号";
```

```
//根据实际情况设置userid 和deviceID，为了快速验证数据上报，请在研发流程内通过设置
用户或者设备白名单进行验证，添加其中一项白名单即可，白名单设置规则请参照下面的注意项
进行添加，

#ifdef DEBUG
[QAPMConfig getInstance].userId = @"请填写用户白名单的值";
[QAPMConfig getInstance].deviceID = @"请填写设备白名单的值";
#else
//设备唯一标识deviceID，例如IDFV配合Keychain使用
[QAPMConfig getInstance].userId = @"自定义userId";
[QAPMConfig getInstance].deviceID = @"自定义deviceId";
#endif

/// 启动QAPM
[QAPM startWithAppKey:@"产品唯一的appKey"];
return YES;

}
```

> **注意：**
> 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，可登录 终端性能监控控制台，进入**应用管理 > 白名单管理 > 白名单配置**页面，单击**添加**完成操作。

步骤4：监控功能启用

初始化后需开启监控功能，支持使用推荐配置一键开启监控功能和自定义开启某个功能。

推荐使用

- 使用 `QAPMModelStableConfig.h` 文件中的接口：
  `- (void)setupModelAll; //此接口可开启全部监控功能。`
- 使用 `QAPMModelStableConfig.h` 文件中的接口：
  `- (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，可选值为如下：

```
///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。

功能配置
配置 SDK 功能

最近更新时间：2025-04-16 14:27:12

本文将为您介绍如何配置 SDK 功能。

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：**
>
> - 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%的采样抽样。

原生层网络监控以及大模型 SSE 监控

功能说明

QAPMMonitorTypeHTTPMonitor http 日志是聚合上报，1min上报一次。

相关接口

```
@interface QAPMConfig : NSObject
/**
 取得 QAPM 配置的共享实例，修改实例的属性必须在调用 QAPM 启动函数之前执行
 @return QAPMConfig 的共享实例
 */
+ (instancetype)getInstance;

#pragma mark *** 全链路监控相关接口的设置***
/**
 该字段用于设置全链路监控，默认值为 NO,如果设置为 YES 有以下几点需要注意
 1、业务网络请求的 requestHeader 会增加 sw8、traceparent 相关协议下的 value 值，如果业务自行组建了全链路监控则可以设置为 NO
 2、客户服务端可能有防跨越、防注入保护功能、header 信息增加后可能会导致服务端无法返回正确信息的风险，需要服务端放开 Nginx 配置开启、可参照 https://cloud.tencent.com/document/product/248/87108链接下的步骤3
 */
@property (nonatomic, assign) BOOL isOpenTrace;

/**
 全链路监控，只有在上述 isOpenTrace 开关为 YES 时以下参数设置才会生效
 以下四种方式在使用的时候建议只使用注入或者忽略其中的一种设置
 设置不需要注入(忽略)请求头的请求url，
 */
@property (nonatomic, copy)NSArray<NSString *> *injectTraceIgnoreUrls;
/**
 设置不需要注入(忽略)请求头的请求 RegExp
 */
```

```
@property (nonatomic, copy)NSArray<NSString *>
*injectTraceIgnoreUrlsByRegex;
/**
 设置需要注入请求头的请求 url
 */
@property (nonatomic, copy)NSArray<NSString *> *injectTraceUrls;
/**
 设置需要注入请求头的请求 RegExp
 */
@property (nonatomic, copy)NSArray<NSString *> *injectTraceUrlsByRegex;


#pragma mark ***原生网络监控相关接口的设置***
/**
 * 设置网络请求的响应 header 黑名单，在黑名单中的 header key 将不会采集上报
 */
- (void)addHttpBlackHeader:(NSArray<NSString *> *)blackHeader;
/**
 * 设置域名白名单，如果设置了，则只有白名单中的域名请求才监控
 */
-(void)addHttpWhiteDomain:(NSArray<NSString *> *)whiteDomain;
/**
 * 设置域名黑名单，如果设置了，在黑名单中的域名将不会监控
 */
-(void)addHttpBlackDomain:(NSArray<NSString *> *)blackDomain;


#pragma mark ***SSE 大模型监控相关接口的设置***
/**
 * 设置当前大模型名称
 */
-(void)setCurrentLLM:(NSString*)llmName;
@end
```

代码示例

以原生网络监控设置接口为例，在进行 QAPM 的 SDK 初始化 时加入如下代码。

```
//设置域名黑名单，以下域名将不会被监控
[[QAPMConfig getInstance] addHttpBlackDomain:@[@"www.qq.com"]];

//设置域名白名单，只有白名单中的域名才会被监控
[[QAPMConfig getInstance] addHttpWhiteDomain:@[@"www.baidu.com"]];
```

```
//设置响应 header 黑名单，在黑名单中的 header key 将不会采集上报
[[QAPMConfig getInstance] addHttpBlackDomain:@[@"authorization"]];
```

⚠ **注意：**

- 从5.4.6版本开始，原生层网络监控开始加入了大模型 SSE 监控，该功能随原网络监控功能同时开启。
- 全链路监控实际效果以及页面配置请参见 全链路监控 。
- 原生网络监控默认监控所有 url，可通过上面的黑白名单接口来决定是否进行过滤，在同时设置了域名白名单和域名黑名单接口的时候，只有域名白名单的设置才会生效，建议优先使用白名单设置。

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（默认4000ms），则会上报个例详情。个例详情包括启动耗时、自定义打点区间和启动过程堆栈。

相关接口

```
@interface QAPMLaunchProfile : NSObject
/**
 开启启动耗时监控的调用
*/
+ (void)didEnterMain;
@end
```

代码示例

在工程的 main 函数导入 `#import <QAPM/QAPMLaunchProfile.h>` 头文件，并进行如下调用。

```
@import UIKit;
#import "QAPMAppDelegate.h"
#import <QAPM/QAPMLaunchProfile.h>
int main(int argc, char * argv[])
{
    @autoreleasepool {
        [QAPMLaunchProfile didEnterMain];
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([QAPMAppDelegate class]));
```

```
    }
}
```

Web 监控功能

功能说明

该功能能够监控 Web 网络资源加载耗时和 jserror 监控。

相关接口

```
@interface QAPMWebViewProfile : NSObject
/**
@param breadCrumbBuckets  自定义上报 webview 移动分析部分的分桶
@return 返回注入的基本信息，包含 QAPM 的初始化信息
*/
+ (NSString *)qapmBaseInfo:(NSString *) breadCrumbBuckets;

/**
@return 注入启动 js 监控的信息，请在 resetConfig 方法调用完之后调用
*/
+ (NSString *)qapmJsStart;
@end
```

代码示例

在工程对应的类里面导入头文 `#import <QAPM/QAPMWebViewProfile.h>` 头文件，在 WKWebView 的代理方法 didFinishNavigation 中添加如下代码。

```
- (void)webView:(WKWebView *)webView didFinishNavigation:
(WKNavigation *)navigation{
[webView evaluateJavaScript:[QAPMWebViewProfile qapmBaseInfo:@""]
completionHandler:nil];
[webView evaluateJavaScript:[QAPMWebViewProfile qapmJsStart]
completionHandler:nil];
}
```

> **注意：**
> 如果使用手动集成的方式，需要将 framework 里面的 QAPMResourceFile.bundle 下的 js_sdk 以 Add Files to 方式引入到工程中；如果使用 cocoaPods 集成方式则不需要。

用户行为功能

功能说明

QAPMMoniterType：QAPMMonitorTypeIUPMonitor 用户行为功能，将会在1min或者超过200条数据聚合上报。

相关接口

- 自定义用户行为功能：在需要打点的页面加入 `#import <QAPM/QAPMUBSMonitor.h>` 头文件后，调用如下接口。

```
@interface QAPMUBSMonitor : NSObject
+ (instancetype)manager;
/**
 用户自定义用户行为操作调用,外部用户接口，调用该接口时请完成 QAPM 的一系列初始化操作，设置完 QAPM 的 appKey 后调用。
 @param category  事件名，建议全部用大写字母表示
 @param tags      字符串的 map 标记，对应的 key 的值只能是d1~d30/info1~info10范围的值
 @param values    数值的 map 标记，对应的 key 的值只能是 v1~v30范围的值
 @return 用户行为事件 uuid，如果返回为 nil，则表示生成自定义用户行为事件失败
*/
- (NSString *)customEvent:(NSString *)category
                     tags:(NSDictionary<NSString *, NSString *> *)tags
                   values:(NSDictionary<NSString *, NSNumber *>
*)values;
```

- 接口调用实例：

```
NSDictionary *tags = @{@"d1":@"iPhone 15 Pro",
                       @"d2":@"OLED显示屏",
                       @"d3":@"多摄像头系统",
                       @"info1":@"包括对专业应用的支持，如增强现实（AR）和专业摄影",
                       @"info2":@"Pro系列的iPhone通常使用更高级的材料，如不锈钢边框和陶瓷保护罩"
                       };
NSDictionary *values = @{@"v1":@512,
                       @"v2":@2048
                       };
```

```
[[QAPMUBSMonitor manager] generateUserEvent:@"CUSTOM_PERF_EVENT"
label:@"" action:@"" value:@1 tags:tags values:values];
```

Webview 用户行为功能

- QAPMMoniterType：QAPMMonitorTypeWebViewIUPMonitor Webview 用户行为功能，将会在1min或者超过200条数据聚合上报。
- 自定义用户行为功能：在 Web 页面代码中需要使用自定义事件的位置（例如单击**购买基金**的处理函数中），编写如下代码：

```
window.QAPM.customEvent(category, tags, values)
参数说明：
（1）category：事件名称，字符串，值的长度不超过100
（2）tags：事件辅助信息，json 格式，key 只能填写 d1~d30和 info1~info10，它们的值都是字符串，其中 d1~d30的值的长度不超过100，info1~info10的值的长度不超过1024，超长的部分会被截断
（3）values：事件辅助信息，json 格式，key 只能填写 v1~v30，它们的值都是 int
```

- 接口调用实例：

```
var event = {
    "category": "CLICK_BUY_BUTTON",
    "values": {
        "v1": 748,
        "v2": 1
    },
    "tags": {
        "d1": "FUJI mini7+",
        "d2": "package:1",
        "d3": "color:white",
        "info1": "Fuji novice recommends the king of cost-effective
instant camera mini7+ one-time imaging cheap film machine for male and
female students",
        "info2": "Package type: Package 1 [Official standard + 20
pieces of photo paper + new product gift package + 10-piece accessory
gift package] Color classification: White"
    }
};
window.QAPM && typeof window.QAPM.customEvent==="function" &&
window.QAPM.customEvent(event.category, event.tags, event.values);
```

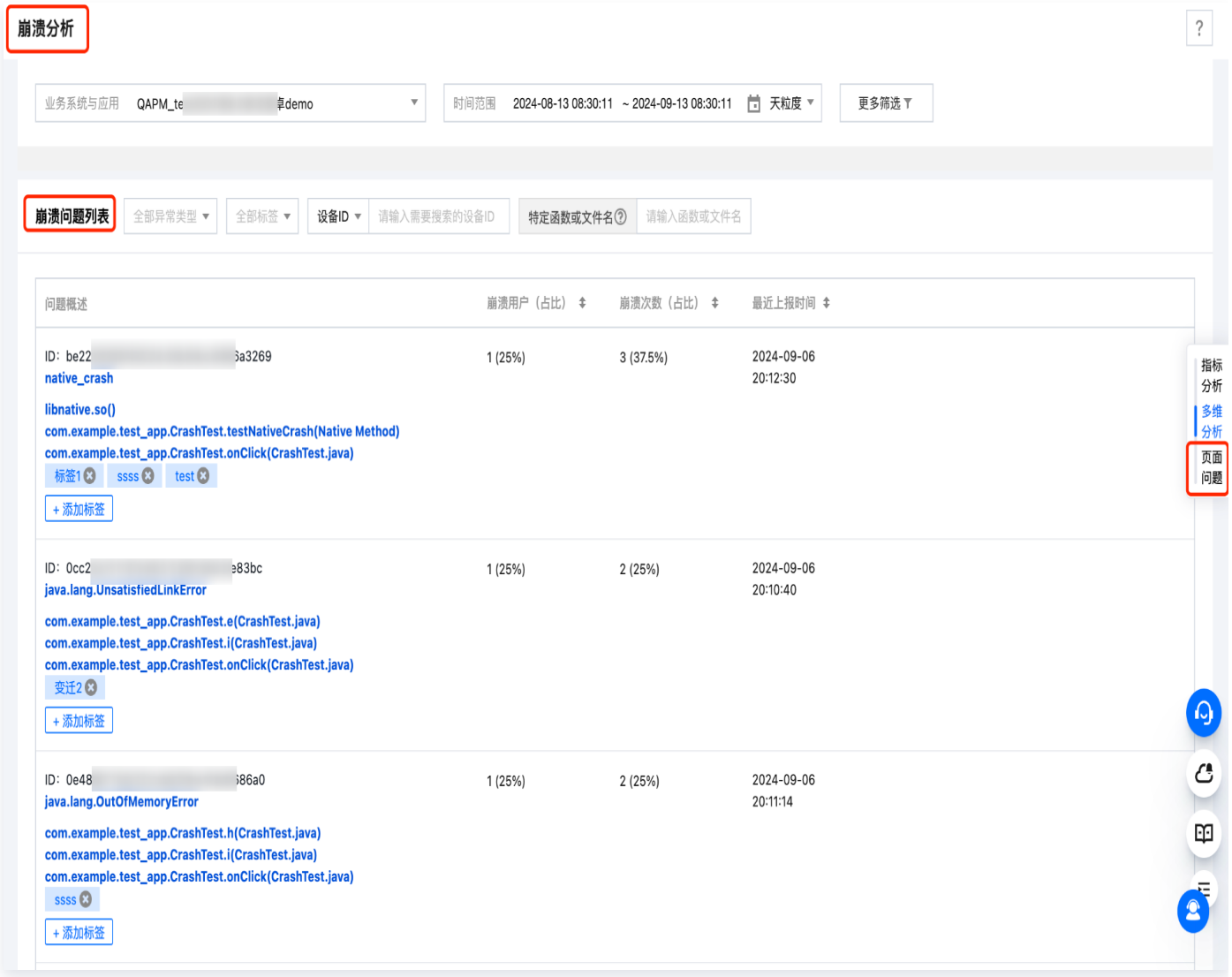
配置符号表

最近更新时间：2024-12-27 11:58:52

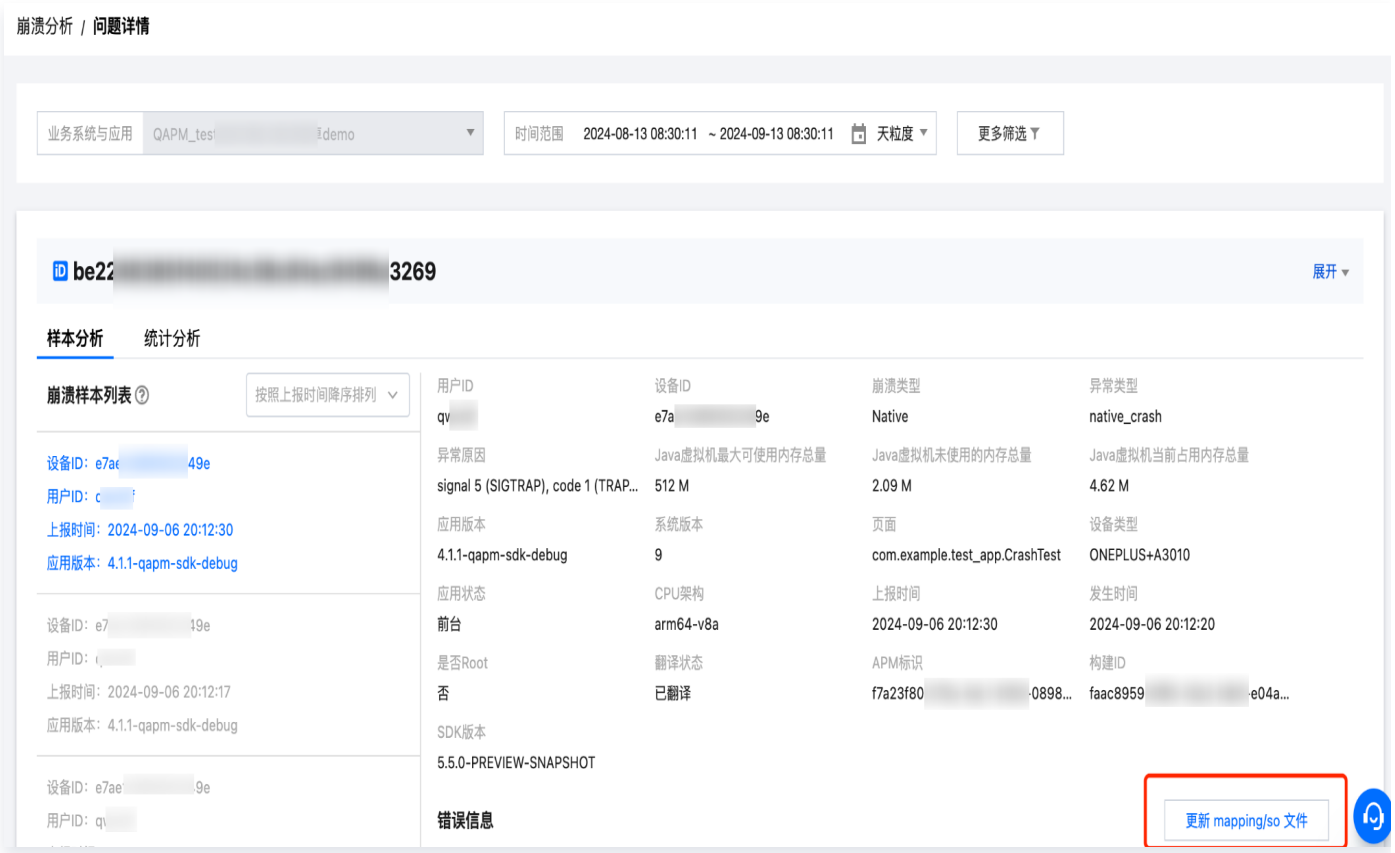
利用已上报的数据个例上传符号表。根据个例页面的构建 ID 找到对应的符号表，可以使用官方 atos 命令翻译个例页面某行堆栈，确认符号表和翻译正常。

操作步骤

1. 在个例页面中有上传的入口，例如进入 终端性能监控 > 崩溃 页面，在**崩溃分析 > 崩溃问题列表**下，进入详情页面。



2. 在详情页面中，单击**更新 mapping/so 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

最近更新时间：2024-12-27 11:58:52

本文将为您介绍如何查看并分析 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：

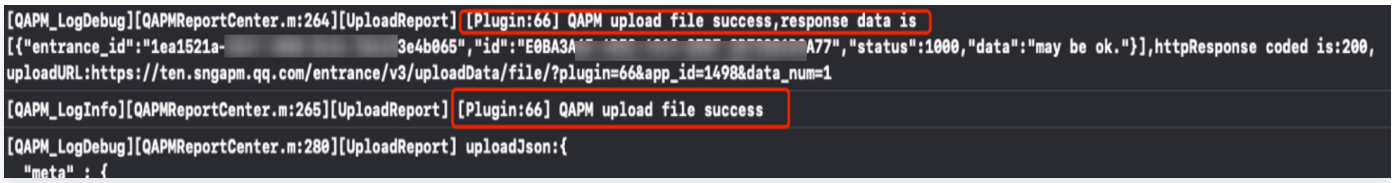
```
[QAPM_LogEvent][QAPM.m:327][Config] [Custom lag monitoring function] The background is not allowed to be enabled
[QAPM_LogEvent][QAPM.m:331][Config] [Frame drop rate monitoring function] is not allowed to be enabled in the background
[QAPM_LogEvent][QAPM.m:335][Config] [Blue (lag monitoring) function] 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：

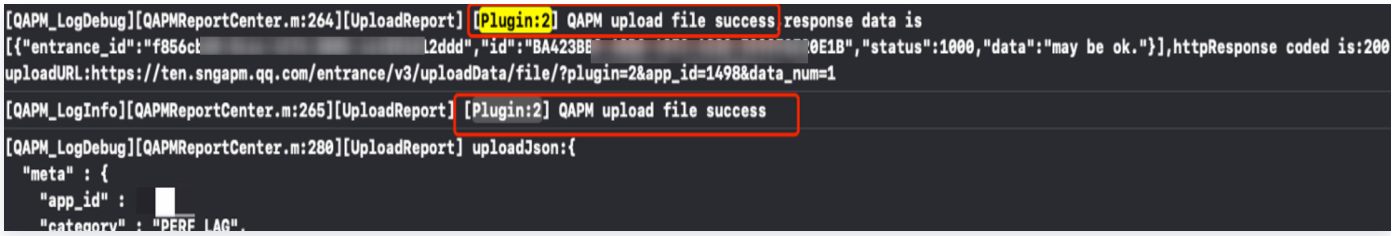
```
  },
  "update_date" : "2024-06-26"
}
2024-06-26 17:39:02.333587+0800 QAPM_Example[24608:742590] [QAPM_LogDebug][QAPMNetworkHandler.m:226][Config] 2024-06-26 Record this time 2024-06-26 Functions that have been turned on on the day: 53
2024-06-26 17:39:02.333751+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPMPathUtil.m:56][Config] getPubKeyUrl:https://ten.sngapm.qq.com/entrance/1018/requestForPubKey/?app_key=64871f09&p_id=1018&deviceid=weiton-test111&plugin_ver=1&sdk_ver=5.3.8
2024-06-26 17:39:05.670937+0800 QAPM_Example[24608:742590] [QAPM_LogDebug][QAPMCrashMonitorManager.mm:55][CRASH] crash install state: OK
2024-06-26 17:39:05.671247+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:181][Config] [Normal crash monitoring function] is enabled
2024-06-26 17:39:05.673486+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:186][Config] [Memory Graph monitoring function] is enabled
2024-06-26 17:39:05.673757+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:191][Config] [Deadlock monitoring function] is enabled
2024-06-26 17:39:05.690233+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPMOOMDetectorManager.m:228][Foom] The current APP is connected to Xcode status or non-App Store packages or whitelists, and the FOOM monitoring function is fully enabled
2024-06-26 17:39:05.690500+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:197][Config] [FOOM monitoring function] is enabled
2024-06-26 17:39:05.690736+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:206][Config] [Crash indicator] is enabled
2024-06-26 17:39:05.691776+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:233][Config] [User behavior monitoring AthenaSDK function] is enabled
2024-06-26 17:39:05.696618+0800 QAPM_Example[24608:742590] [QAPM_LogDebug][QAPMEventConn.m:104][Athena] appid:1018,userid:123456789,buildid:unknown,version:1.25
2024-06-26 17:39:05.697034+0800 QAPM_Example[24608:742590] [QAPM_LogDebug][QAPMEventConn.m:127][Athena] The current performance event reporting logic is real-time JSON reporting, and the reporting interval is 1 minute
2024-06-26 17:39:05.697931+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:236][Config] [ubs monitoring function] is enabled
2024-06-26 17:39:05.698597+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:242][Config] [Custom lag monitoring function] is enabled
2024-06-26 17:39:05.702093+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:252][Config] [Yellow (VC leak detection function)] is enabled
2024-06-26 17:39:05.702603+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:257][Config] [Memory maximum usage value monitoring (peak rate function)] is enabled
2024-06-26 17:39:05.704638+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMDataUploadCenter.m:655][UploadReport] The local certificate is valid
2024-06-26 17:39:05.704913+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMDataUploadCenter.m:664][UploadReport] The server certificate is the same as the local certificate
2024-06-26 17:39:05.705444+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:263][Config] [The Chunk Malloc Monitoring Function] is enabled
2024-06-26 17:39:05.705712+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:272][Config] [QQLeak Memory Leak Detection Function] is enabled
2024-06-26 17:39:05.706203+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:283][Config] [Blue (lag monitoring) monitoring function] is enabled
2024-06-26 17:39:05.709615+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:291][Config] [HTTP monitoring function] is enabled
2024-06-26 17:39:05.710182+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:299][Config] [WebMonitor] background configuration is allowed to be enabled, please click on the corresponding page
2024-06-26 17:39:05.710352+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:302][Config] [WebView ubs monitoring function] is enabled
2024-06-26 17:39:05.710616+0800 QAPM_Example[24608:742590] [QAPM_LogEvent][QAPM.m:307][Config] [Frame drop rate monitoring function] background configuration is allowed to be enabled, please click on the corresponding page
2024-06-26 17:39:05.741104+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMDataUploadCenter.m:696][Config] pub_key success, data is {"pub_key":"[redacted]"}
2024-06-26 17:39:05.741340+0800 QAPM_Example[24608:742578] [QAPM_LogInfo][QAPMNetworkHandler.m:308][Config] pub_key success
2024-06-26 17:39:07.377407+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMEventPageChange.m:38][Athena] pre_page is empty
2024-06-26 17:39:07.377843+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMiupObj.m:91][Athena] push_activity
2024-06-26 17:39:09.111994+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMDataUploadCenter.m:655][UploadReport] The local certificate is valid
2024-06-26 17:39:09.112292+0800 QAPM_Example[24608:742578] [QAPM_LogDebug][QAPMDataUploadCenter.m:664][UploadReport] The server certificate is the same as the local certificate
2024-06-26 17:39:09.163794+0800 QAPM_Example[24608:742580] [QAPM_LogInfo][QAPMDataUploadCenter.m:141][UploadReport] [Plugin:35] QAPM upload json success,httpResponse coded is:200, uploadURL:https://te[redacted]rance/1018/uploadJson/
2024-06-26 17:39:09.164089+0800 QAPM_Example[24608:742580] [QAPM_LogDebug][QAPMDataUploadCenter.m:142][UploadReport] [Plugin:35] QAPM upload json success,response data is {"apm_identify":"744717c2[redacted]b9164d9b","status":1000,"data":"may be ok."},httpResponse coded is:200,
  uploadURL:https://ten.sngapm.qq.com/entrance/1018/uploadJson/
2024-06-26 17:39:09.164879+0800 QAPM_Example[24608:742580] [QAPM_LogDebug][QAPMDataUploadCenter.m:155][UploadReport] uploadJson:{
  "event_time" : 1719393798525,
```

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。

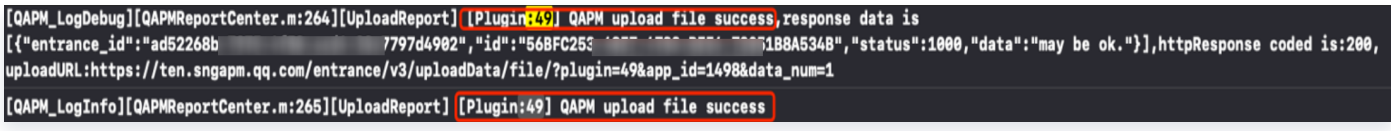
- 启动耗时的上报：



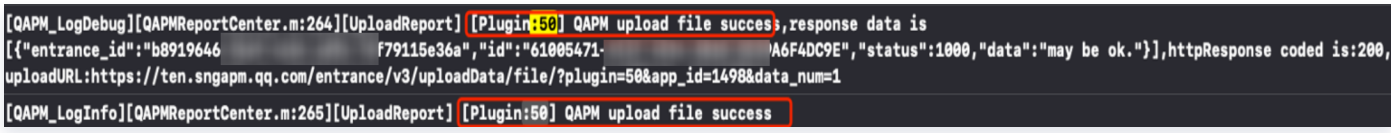
- 卡顿个例的上报：



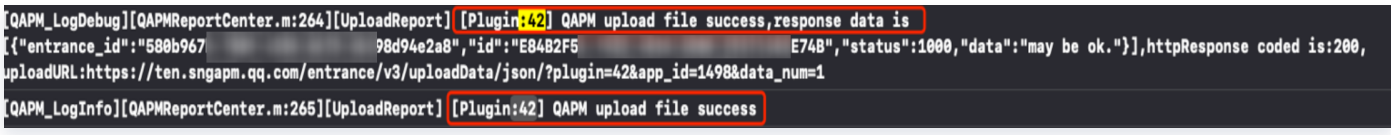
- FOOM 个例上报：



- Deadlock 个例上报：



- HTTP 监控上报：



- 普通崩溃（normal crash）的上报：

  在触发 normal crash 上报时，请不要将数据线连接 Xcode，触发完 normal crash 后，下次重启 App 时即可看到上报信息。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：

```
[QAPM_LogDebug][QAPMReportCenter.m:264][UploadReport] [Plugin:46] QAPM upload file success,response data is
[{"entrance_id":"55643c4                    70b9","id":"4F0CB2EC-                    337E82","status":1000,"data":"may be ok."}],httpResponse coded is:200,
uploadURL:https://ten.sngapm.qq.com/entrance/v3/uploadData/file/?plugin=46&app_id=1498&data_num=1
[QAPM_LogInfo][QAPMReportCenter.m:265][UploadReport] [Plugin:46] QAPM upload file success
```

- 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。

```
[QAPM_LogDebug][QAPMReportCenter.m:264][UploadReport] [Plugin:41] QAPM upload file success,response data is
[{"entrance_id":"5426eba                    88a9aa","id":"fdb5fd4                    fd7031","status":1000,"data":"may be ok."}],httpResponse coded is:200,
uploadURL:https://ten.sngapm.qq.com/entrance/v3/uploadData/json/?plugin=41&app_id=1498&data_num=1
[QAPM_LogInfo][QAPMReportCenter.m:265][UploadReport] [Plugin:41] QAPM upload file success
```

- 用户行为的上报：

  用户行为数据和自定义事件的数据可搜索 athena upload json is 关键字。

```
2024-06-06 21:17:26.045735+0800 QAPM_Example[9259:2422691] [QAPM_LogDebug][QAPMDataUploadCenter.m:450][UploadReport] athena upload json is {
  "meta" : {
```

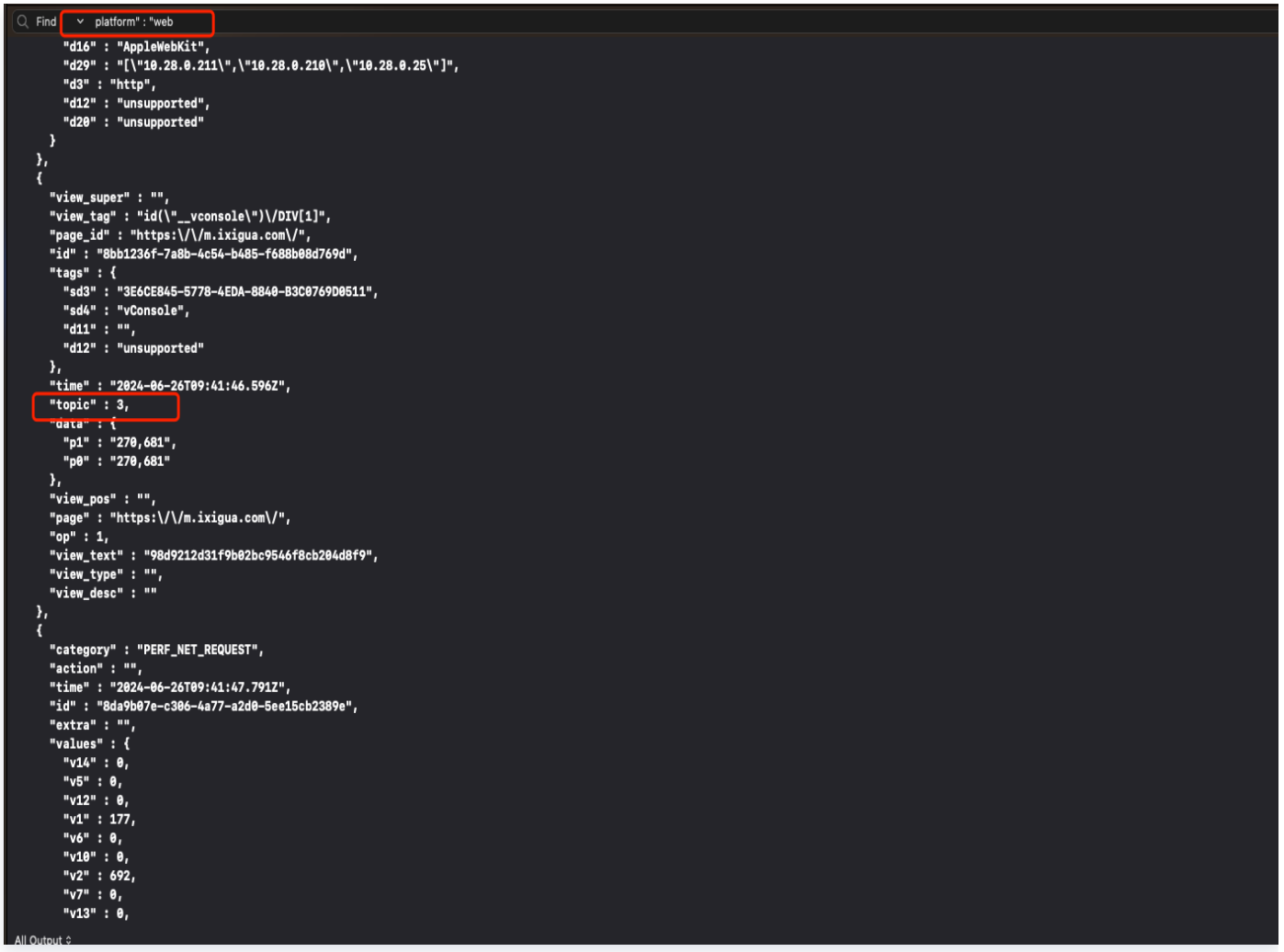
```
64      NSDictionary *tags = @{@"d1":@"iPhone 15 Pro",
65                             @"d2":@"OLED显示屏",
66                             @"d3":@"多摄像头系统",
67                             @"info1":@"包括对专业应用的支持，如增强现实（AR）和专业摄影",
68                             @"info2":@"Pro系列的iPhone通常使用更高级的材料，如不锈钢边框和陶瓷保护罩"
69                            };
70
71      NSDictionary *values = @{@"v1":@512,
72                               @"v2":@2048
73                              };
74
75      [[QAPMUBSMonitor manager] customEvent:@"CUSTOM_BUTTON" tags:tags values:values];
76  }
```

QAPM_Example

```
    },
    {
      "category" : "CUSTOM_EVENT",
      "action" : "",
      "time" : "2024-06-11T01:45:59.808Z",
      "id" : "e950009d-bd01-4277-b122-49f732641d19",
      "values" : {
        "v2" : 2048,
        "v1" : 512
      },
      "topic" : 7,
      "label" : "",
      "value" : 1,
      "tags" : {
        "d31" : "CUSTOM_BUTTON",
        "sd3" : "53BBBCAB-6D42-4E90-A82C-2C0868E9A065",
        "d3" : "多摄像头系统",
        "info1" : "包括对专业应用的支持，如增强现实（AR）和专业摄影",
        "d1" : "iPhone 15 Pro",
        "info2" : "Pro系列的iPhone通常使用更高级的材料，如不锈钢边框和陶瓷保护罩",
        "d2" : "OLED显示屏"
      }
    }
  ]
}
```

- Webview 用户行为的上报：

  Webview 用户行为数据和自定义事件的数据可搜索 athena upload json is 关键字，同时关注 platform 为 web，topic 为3。



鸿蒙应用场景

最近更新时间：2024-12-09 17:35:12

操作场景

本文指导您使用鸿蒙 SDK 的集成与初始化。

> ⚠ **注意：**
> 最低支持鸿蒙 SDK 5.0.0（API12），最低支持系统版本 NEXT.0.0.36。

操作步骤

步骤一：SDK 下载

1. 配置鸿蒙三方仓库，执行以下命令。

```
ohpm config set registry https://ohpm.openharmony.cn/ohpm/
```

2. 下载安装。

```
ohpm install qapmsdk@latest
```

步骤二：加入 QAPMPlugin 插件

1. 在 entry 根目录加入 QAPMPlugin.ts 文件，代码爆红可忽略，可能是 IDE 解析问题，不影响使用。
2. 在 entry 模块的 hvigorfile.ts 中加入以下代码。

```
// 引入QAPMPlugin依赖
import { QAPMPlugin } from './QAPMPlugin'

...
export default {
    ...
    plugins:[
        // 加入插件
        QAPMPlugin()
    ]          /* Custom plugin to extend the functionality of Hvigor.
*/
    ...
}
```

```
...
```

步骤三：混淆规则配置

鸿蒙端二次混淆 Har 包代码，可能会导致 SDK 运行异常。为了避免 SDK 包跟随业务代码混淆，而导致功能失效，建议业务侧在 hap 工程下的混淆配置文件（obfuscation-rules.txt）中加入以下规则。

```
-keep
./oh_modules/qapmsdk
```

步骤四：初始化 SDK

1. 登录 腾讯云可观测平台 控制台，在**终端性能监控**页面，选择 应用管理 > 应用设置 后，获取 Appkey（上报 ID）。

2. 拷贝下面代码，并修改其中部分字段。

```
import { QAPM } from 'qapmsdk' // 导入QAPM模块
import BuildProfile from 'BuildProfile'; // 引入BuildProfile，该文件编译时生成，可忽略报错

initQAPM(context: common.UIAbilityContext): void {
  // 初始化SDK
```

```
  QAPM.getInstance()
  .setContext(context)                                    // 需要传入
UIAbilityContext，必须
  .setAppKey("your appkey")                               // 填写你的AppKey，
必填，用于区分上报的产品，该值由终端性能监控的产品配置页面获取，可参考前一步骤
  .setUserId("userId")                                    // 填写你的用户ID，选
填，不填默认值为USER_NOT_SET
  .setDeviceId("deviceId")                                // 填写设备ID，必填，
应保证该值的唯一性
  .setModel("model")                                      // 填写设备类型，选
填，不填默认值为USER_NOT_SET，参考 deviceInfo.productModel
  .setLogLevel(QAPM.LogDebugLevel)                        // 设置日志等级，选
填，开发阶段开启Debug，线上请关闭
  .setBuildId(BuildProfile.QAPM_UUID)                     // 设置BuildId，选
填，用于拉取被混淆堆栈的mapping （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与一次构建是相互对应关系，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，报错信息可忽略，该设置
如不填会影响后续堆栈翻译
  .setHost("https://app.rumt-zh.com")                     // 设置QAPM的外网上
报域名，必填
  .setCollectOptionalFields(true)                         // 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，该设置需要最先配置，
一旦设置则全局生效。默认可以采集，设置为false则不采集，可能会影响到控制台的搜索、展
示等。可选个人信息包括但不限于以下信息：设备制造商、系统、设备型号等，详见《QAPM
SDK合规使用指南》
}

// 启动QAPM
startQAPM(): void {
  // 启动SDK
  QAPM.getInstance()
  .start(QAPM.ModeAll)                                    // 开启所有功能，当前
功能包含JsCrash、CppCrash、AppFreeze
}
```

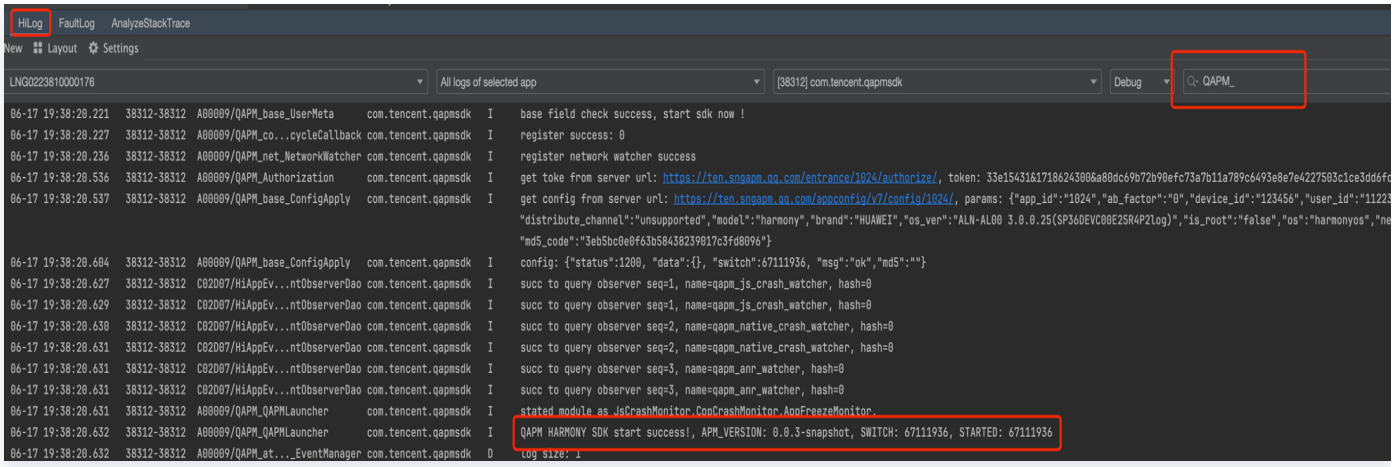
说明：

- AppKey 可参考步骤四，在 终端性能监控 > 应用管理 > 应用设置 页面获取。
- 崩溃与 AppFreeze 分析均采用全量采集上报策略，不做采样处理。

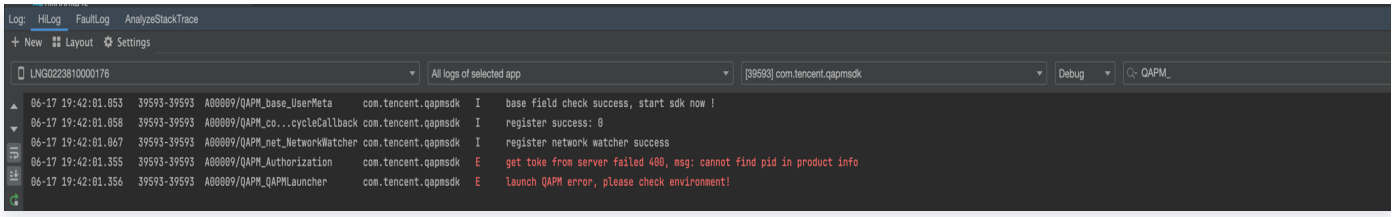
步骤五：接入验证

1. 检测 SDK 接入是否正常

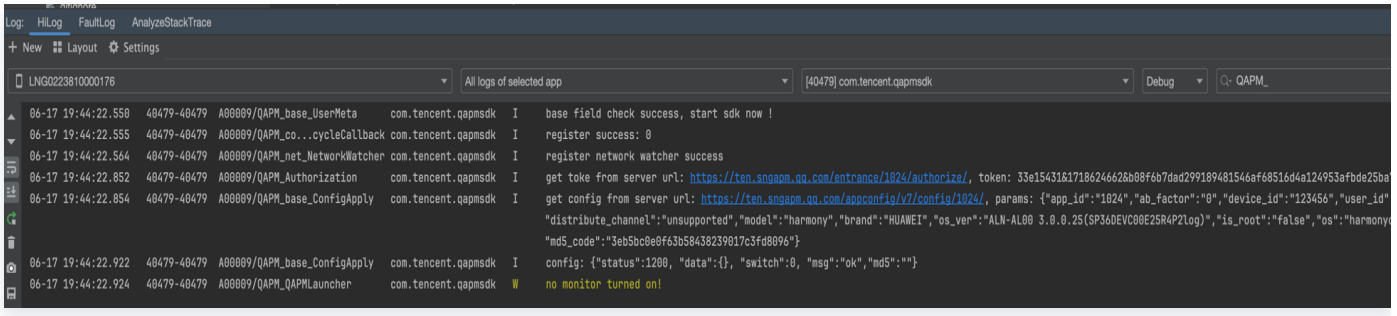
- 需要开启 QAPM 的日志等级为 Debug，启动 App，打开 Hilog，过滤 TAG 为 QAPM_，如出现以下日志代表接入成功。



- 如出现 `launch QAPM error, please check environment!` 的错误，请检查 appkey 或者域名是否设置错误。



- 如出现 `no monitor turned on!` 则可能是抽样未命中，请前往 终端性能监控 > 应用管理 > **白名单管理**页面加入用户或者设备白名单，日志中可搜索 user_id 或者 device_id 查看具体值。



2. 我们在 SDK 中内嵌了一些崩溃测试 API，如下：

- 调用 QAPM.getInstance().testJsCrash()，触发 Js 崩溃，下次启动时观察日志是否有[plugin::744]的上报。

- 调用 QAPM.getInstance().testCppCrash()，触发 Cpp 崩溃，下次启动时观察日志是否有[plugin::746]的上报。
- 调用 QAPM.getInstance().testAppFreeze()，触发 AppFreeze，下次启动时观察日志是否有[plugin::740]的上报。

ReactNative 应用场景

最近更新时间：2025-01-24 09:56:12

说明：

- 目前 RN 强绑定终端运行，如果已经有了 Android 或 iOS 的应用，请直接使用对应的 AppKey，如没有请创建时选择 Android 或 iOS 应用。
- 仅支持 RN 0.60.0以上版本。

自动集成

执行如下命令安装依赖，即可自动集成 SDK。

```
yarn add @qapm/react-native-qapm --save
```

配置原生支持

Android 平台

配置 Gradle

1. 在工程级的 `build.gradle` 文件下，增加 `maven` 依赖和 `qapm-plugin` 插件。

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

    dependencies {
        // 添加qapm-plugin插件
        classpath 'com.tencent.qapmplugin:qapm-plugin:3.4'
    }
}


allprojects {
    repositories {
```

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

> ⚠ **注意：**
> 如果您的 **gradle 版本小于7.4**，则将 qapm-plugin 插件版本改为2.39。

2. 在 `app的build.gradle` 文件下，增加 `qapm-plugin` 的引用。

```
...
// 引用插件
apply plugin: "qapm-plugin"
...

// 添加自动生成UUID的Task
preBuild.dependsOn(UUIDGenerator)
```

配置权限

在 `AndroidMainfest.xml` 文件中增加以下的权限。

```
<!--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" />
```

配置混淆

在 `proguard 混淆配置文件` 中增加以下内容，避免打包报错或 SDK 不可用。

```
-dontwarn com.tencent.cos.xml.**
-dontwarn com.tencent.qcloud.**
-keep class com.tencent.qapmsdk.** { *; }
-dontwarn com.tencent.qapmsdk.**
```

iOS 平台

在 iOS 的工程路径下，执行 `pod install` 更新依赖。

```
pod install
```

初始化 QAPM SDK

在 RN 入口文件 `index.js` 中添加如下代码。

```
import { QAPM, QAPMModel, QAPMConfiguration, QAPMFuncBuilder,
SdkVerbosity } from '@qapm/react-native-qapm';
import { version as reactNativeVersion } from 'react-
native/package.json';
import { Platform } from 'react-native';

const modelBuilder = new QAPMFuncBuilder()
    .enable(QAPMModel.MODEL_CRASH)
    .build();
// 如果您工程的原生层也使用到了QAPM，请和原生开发联系，确保以下的参数设置与原生侧一
致，否则可能会出现数据错乱的问题
const config = new QAPMConfiguration.Builder()
    .setAppKey(Platform.OS === 'ios' ? '<your-ios-appKey>' : '<your-
android-appKey>') // 上报凭证
    .setAppVersion('<your-app-version>')          // app应用版本
    .setCrossLibraryVersion(reactNativeVersion) // 引擎版本
    .setCrossVersion('<your-cross-version>')     // 跨端应用版本
    .setUserId('<user-id>')                       // 用户ID
    .setDeviceId('<device-id>')                   // 设备ID
    .setServerHost('<report-url>')                // 上报地址
    .setPhoneModel('<user-phone-model>')          // 设备类型
    .setLogLevel(SdkVerbosity.DEBUG)              // 设置日志等级
    .setAppFuncSwitch(modelBuilder)
    .build();
QAPM.start(config);
```

接入路由追踪

- 如果您使用了 `@react-navigation/native`，则在 `App.tsx`（根据接入应用而定）中添加如下代码。

```
import { QAPMReactNavigationTracking } from '@qapm/react-native-qapm';
```

```
const Stack = createNativeStackNavigator();
function App(): React.JSX.Element {
    const navigationRef: React.Ref<
        NavigationContainerRef<ReactNavigation.RootParamList>
    > = React.useRef(null);
    return (
        <NavigationContainer
            ref={navigationRef}
            onReady={() => {
                // 将路由信息传给QAPM

QAPMReactNavigationTracking.start(navigationRef.current);
            }}
        >
            <!--
                路由定义
            -->
        </NavigationContainer>
    );
}
```

- 如果项目没有使用 `@react-navigation/native`，可以使用手动上报。

```
import { QAPMReactNavigationTracking } from '@qapm/react-native-qapm';
// 主动上报页面
QAPMReactNavigationTracking.updatePage('HomeScreen');
```

验证接入是否成功

- 关键字搜索 `QAPM_ReactNative`，如出现以下日志（以 android 为例），则代表接入成功。

```
 (NOBRIDGE) INFO  QAPM_ReactNative:  QAPM react navigation track
start!
 (NOBRIDGE) INFO  QAPM_ReactNative:  state change , current route key
Home-jXJD2FvowJDjvRHrSaPAr, name Home
 (NOBRIDGE) DEBUG  QAPM_ReactNative:  onConfigFetch success: true,
            message: success,
            mode: 1000000000000000000000000000000000000000,
            config:
 (NOBRIDGE) INFO  QAPM_ReactNative:  checking for QAPMErrorTracking
```

```
(NOBRIDGE) INFO  QAPM_ReactNative:  start QAPMErrorTracking
(NOBRIDGE) INFO  QAPM_ReactNative:  start error monitor success
```

- 如出现以下日志，说明接入失败，请检查 appKey 等参数是否填写正确。

```
QAPM_ReactNative:  start rn qapm sdk failed, because xxxx!
```

注意：

- 如果您开启了混淆，请保管好 .map 的符号表文件，如果您有多份 .map，请一起压缩成 zip 并上传。
- 如果原生也使用了 QAPM，不需要再引入 QAPM 依赖，SDK 会直接使用 API 的方式让上层模块可以直接调用。

Flutter 应用场景

最近更新时间：2025-01-24 14:07:52

> **说明：**
> 目前 Flutter 强绑定终端运行，如果已经有了 Android 或 iOS 的应用，请直接使用对应的 AppKey，如没有请创建时选择 Android 或 iOS 应用。

自动集成

1. 安装。
   - 在项目的 `pubspec.yaml` 文件中添加 `qapm_flutter` 的依赖。

```
# 在pubspec.yaml中加入
dependencies:
    qapm_flutter: ^0.1.0
```

   - 运行 `flutter pub get` 更新依赖。

```
flutter pub get
```

2. 导入。

```
import 'package:qapm_flutter/qapm_flutter.dart';
```

配置原生支持

Android 平台

配置 Gradle

1. 在工程级的 `build.gradle` 文件下，增加 `maven 依赖` 和 `qapm-plugin 插件` 。

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

```
    }

    dependencies {
        // 添加qapm-plugin插件
        classpath 'com.tencent.qapmplugin:qapm-plugin:3.4'
    }
}


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

> **说明：**
> 如果您的 **gradle 版本小于7.4**，则将 qapm-plugin 插件版本改为2.39。

2. 在 app 的 `build.gradle` 文件下，增加 `qapm-plugin` 的引用。

```
...
// 引用插件
apply plugin: "qapm-plugin"
...

// 添加自动生成UUID的Task
preBuild.dependsOn(UUIDGenerator)
```

配置权限

在 `AndroidMainfest.xml` 文件中增加以下权限。

```
<!--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" />
```

配置混淆

在 `proguard` 混淆配置文件中增加以下内容，避免打包报错或 SDK 不可用。

```
-dontwarn com.tencent.cos.xml.** { *; }
-dontwarn com.tencent.qcloud.** { *; }

-dontwarn com.squareup.okhttp.** { *; }
-dontwarn okhttp3.** { *; }

-keep class com.tencent.qapmsdk.** { *; }
-dontwarn com.tencent.qapmsdk.** { *; }
```

iOS 平台

在 iOS 的工程路径下，执行 `pod install` 更新依赖。

```
pod install
```

初始化 QAPM SDK

1. 为保证能够捕获全局的异常，建议您将应用的 `void main() => runApp(MyApp());` 替换成以下代码。
   - 如果您的工程是纯 flutter 开发，或者原生（Android 和 iOS）并没有接入 QAPM，则参考以下方式初始化 QAPM。

```
  void main() {
  // 初始化QAPM
  QAPMBuilder builder = QAPMBuilder();
  // 目前AppKey还是沿用原生层的，所以这里需要在控制台创建对应平台的应用，并且
分别设置
  if (Platform.isAndroid) {
    builder.appKey = "2b6f3f82-11907";
  } else if (Platform.isIOS) {
    builder.appKey = "26388a8d-3649";
  }
  builder.model = "HUAWEI";                    // 设置手机型号
  builder.appVersion = "1.0";                  // 设置原生应用版本，这里纯
flutter开发可以和crossVersion填一样值，如是混合开发建议和原生侧协商
  builder.crossVersion = "1.0";                // 跨端的应用版本
  builder.deviceId = "uuid";                   // 设置设备ID
```

```
  //目前上报的域名分为了三个不同的地址，分别是国内站、新加坡站、法兰克福站，根据实际需求选择一个环境上报即可
  //国内站：https://app.rumt-zh.com
  //新加坡站：https://app.rumt-sg.com
  //法兰克福站：https://eu-frankfurt.rumapp.tencentcs.com
  builder.host = "https://app.rumt-zh.com"; // 设置上报域名
  builder.needLog = true;                   // 是否需要开启日志，release下建议关闭
  builder.userId = "11223344";              // 设置用户ID
  builder.funcSwitch = QAPM.MODE_CRASH; // 需要开启跨端的功能，这里目前仅支持crash
  // 确保去掉原有的WidgetsFlutterBinding.ensureInitialized() ，以免出现重复初始化绑定的异常造成无法正常初始化，SDK内部已通过initFlutterBinding入参带入继承的WidgetsFlutterBinding实现初始化操作
  builder.initFlutterBinding = MyApmWidgetsFlutterBinding.ensureInitialized;
  // 启动SDK
  QAPM.init(builder, () {
    runApp(MyApp());
  });
}

class MyApmWidgetsFlutterBinding extends QapmWidgetsFlutterBinding {
  @override
  void handleAppLifecycleStateChanged(AppLifecycleState state) {
    // 添加自己的实现逻辑
    print('AppLifecycleState changed to $state');
    super.handleAppLifecycleStateChanged(state);
  }

  static WidgetsBinding? ensureInitialized() {
    MyApmWidgetsFlutterBinding();
    return WidgetsBinding.instance;
  }
}
```

- 如果您的工程是**原生 + Flutter 都需要使用 QAPM 的混合开发场景**，则参考以下方式初始化 QAPM。

```
// 设置与原生通信的桥
const channel = MethodChannel("com.example.flutter/qapm");

void main() async {
```

```
  QAPMBuilder builder = QAPMBuilder();
  builder.needLog = true;
// 是否需要开启日志，release下建议关闭
  builder.crossVersion = "1.0";
// 跨端的应用版本
  builder.funcSwitch = QAPM.MODE_CRASH;
// 需要开启跨端的功能，这里目前仅支持crash
  // 确保去掉原有的WidgetsFlutterBinding.ensureInitialized() ，以免出现重复初始化绑定的异常造成无法正常初始化，SDK内部已通过initFlutterBinding入参带入继承的WidgetsFlutterBinding实现初始化操作
  builder.initFlutterBinding = MyApmWidgetsFlutterBinding.ensureInitialized;

  QAPM.init(builder, () {
    runApp(MyApp());
  }, beforeInitRunner: (builder) async {
    // 原生层通过channel，实现对应逻辑，并将对应值传递给flutter，需要确保原生和flutter初始化的参数一致！！！
    builder.appKey = await channel.invokeMethod("getAppKey") as String;          // 目前AppKey还是沿用原生层的，所以这里需要在控制台创建对应平台的应用，并且分别设置
    builder.model = await channel.invokeMethod("getModel") as String;            // 设置手机型号，需要跨端开发和原生开发协商，设置一致的值
    builder.appVersion = await channel.invokeMethod("getAppVersion") as String; // 设置原生应用版本，需要跨端开发和原生开发协商，设置一致的值
    builder.deviceId = await channel.invokeMethod("getDeviceId") as String;      // 设置设备ID，需要跨端开发和原生开发协商，设置一致的值
    builder.host = await channel.invokeMethod("getHost") as String;
// 设置上报域名，需要跨端开发和原生开发协商，设置一致的值
    builder.userId = await channel.invokeMethod("getUserId") as String;          // 设置用户ID，需要跨端开发和原生开发协商，设置一致的值
  });
}

class MyApmWidgetsFlutterBinding extends QapmWidgetsFlutterBinding {
  @override
  void handleAppLifecycleStateChanged(AppLifecycleState state) {
    // 添加自己的实现逻辑
    print('AppLifecycleState changed to $state');
    super.handleAppLifecycleStateChanged(state);
  }
```

```
  static WidgetsBinding? ensureInitialized() {
    MyApmWidgetsFlutterBinding();
    return WidgetsBinding.instance;
  }
}
```

2. 在 App 中设置路由监听。

```
class _MyAppState extends State<MyApp> with WidgetsBindingObserver {

  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('APM Example'),
        ),
        body: Center(
          child: MyHome(),
        ),
      ),
      // 添加 QapmNavigatorObserver
      navigatorObservers: [
        QapmNavigatorObserver()
      ],
      routes: {
        'MainPage': (context) => MainPage(),
        'FirstPage': (context) => FirstPage(),
        'SecondPage': (context) => SecondPage(),
      },
    );
  }
}
```

如果是接入了 `TRouter` 等混合栈的路由框架，由于其 接管了原生的 `Navigator` ，需要将 `QapmNavigatorObserver` 设置到混合栈框架中，代码如下：

```
runApp(MaterialApp(
    home: TRouteContainer(navigatorObservers:
QapmNavigatorObserver(),),
));
```

> ⚠ **注意：**
> TRouter 需要使用2.5.7+ 版本，否则部分生命周期会丢失。

验证接入是否成功

关键词搜索 `QAPM_flutter`，如出现以下日志（`Android` 为例），说明初始化成功了。

```
2025-01-06 21:29:35.330 20883-21019 flutter
com....qapmsdk.qapm_flutter_example  I  [QAPM_flutter_CrashMonitor]
[DEBUG] init FlutterExceptionHandler success
```

> ⚠ **注意：**
> - 确保去掉原有的 `WidgetsFlutterBinding.ensureInitialized()` ，以免出现重复初始化绑定的异常造成无法正常初始化，SDK 内部已通过 initFlutterBinding 入参带入继承的 WidgetsFlutterBinding 实现初始化操作。
> - 请 正确设置路由监听 ，否则可能采集不到页面，影响指标计算。
> - 请尽可能确保页面路由使用可读性强的名称，如 'MainPage': (context) => MainPage()。
> - 如果您**启用了 Dart 混淆**，请保管好对应的符号表（Android 为 symbols，iOS 为 App.Framework.dSYM）文件，并在详情页面上根据指引将符号表上传。开启混淆后部分类、属性也会相应混淆，请将 `--extra-gen-snapshot-options=--save-obfuscation-map` 指令生产的符号文件一起上传。
>   - Android 上传符号表请参见 符号表配置 。
>   - iOS 上传符号表请参见 符号表配置 。
> - 如果原生也使用了QAPM，不需要再引入 QAPM 依赖，SDK 会直接使用 API 的方式让上层模块可以直接调用。

控制台操作指南

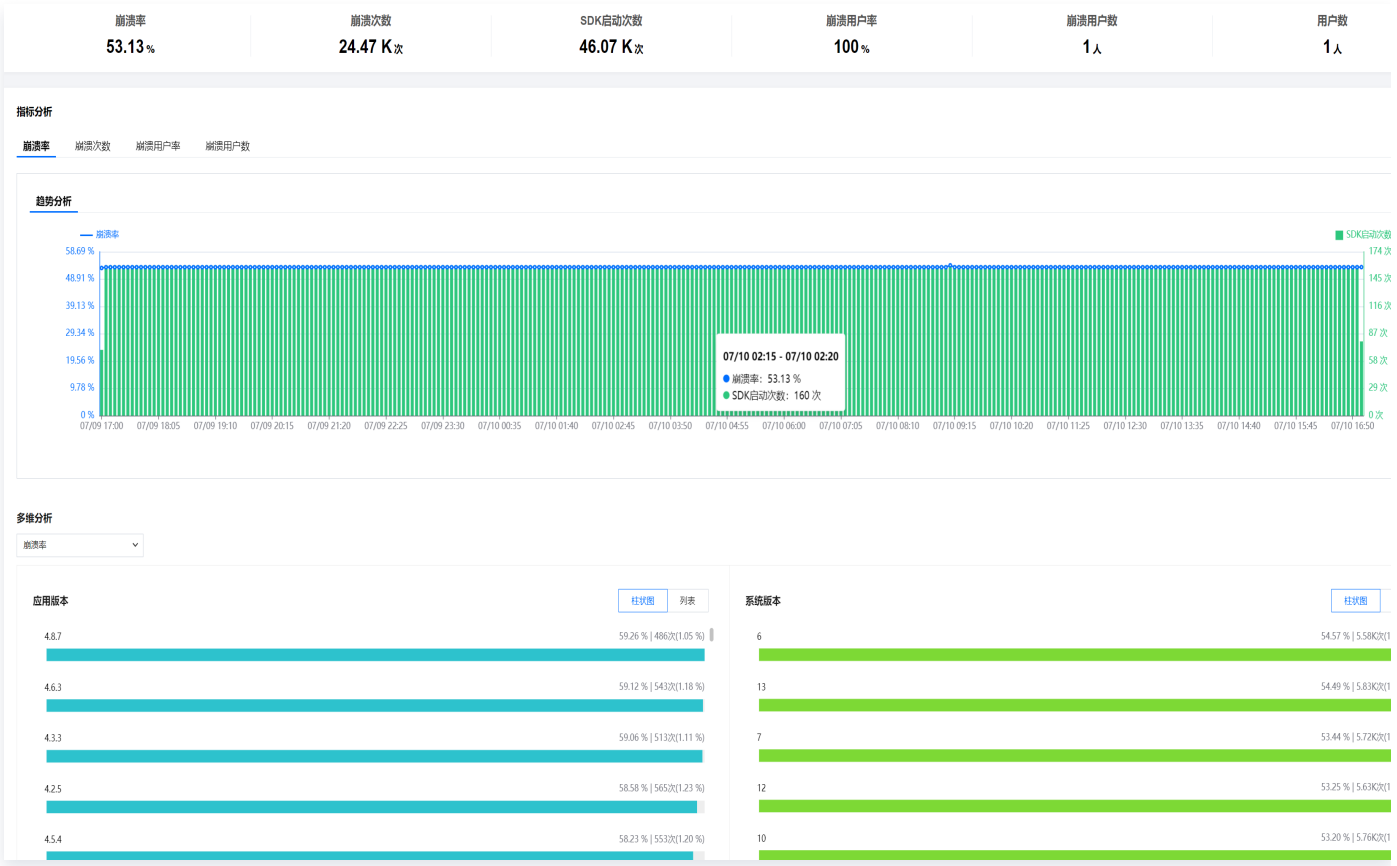
崩溃

最近更新时间：2024-07-11 10:20:41

终端性能监控通过对崩溃问题个例提取关键特征进行聚合，便于您针对 App 崩溃的根因分析。

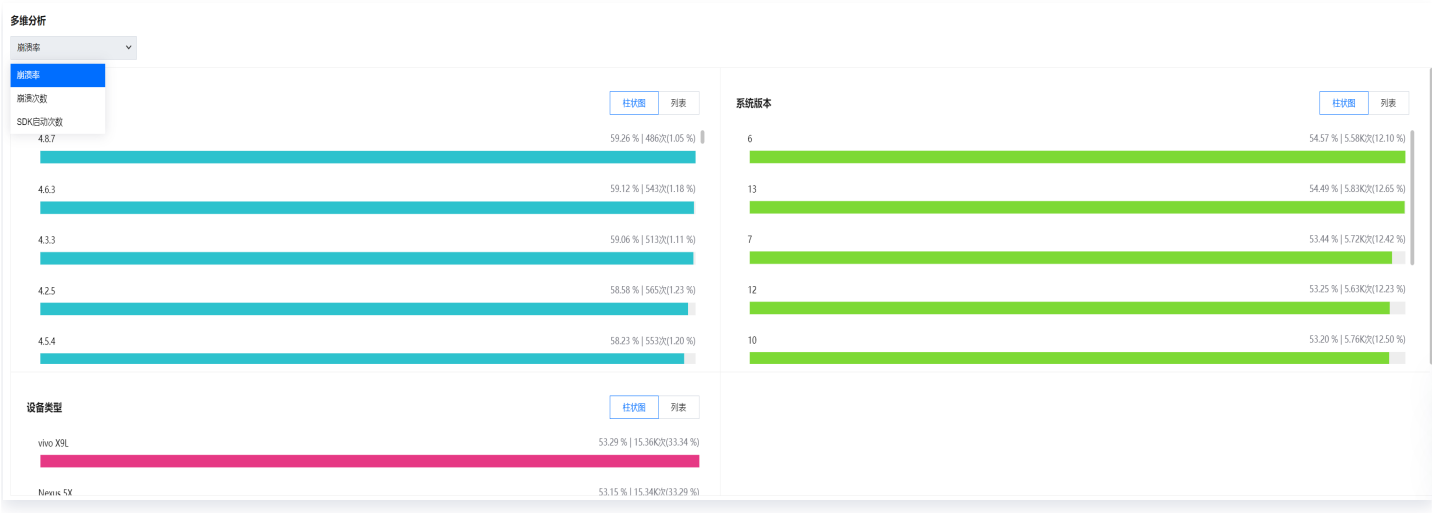
功能入口

1. 登录 终端性能监控控制台 。
2. 在左侧导航栏中选择**崩溃**，选择需要查看的业务系统、应用、时间范围等分析崩溃问题。



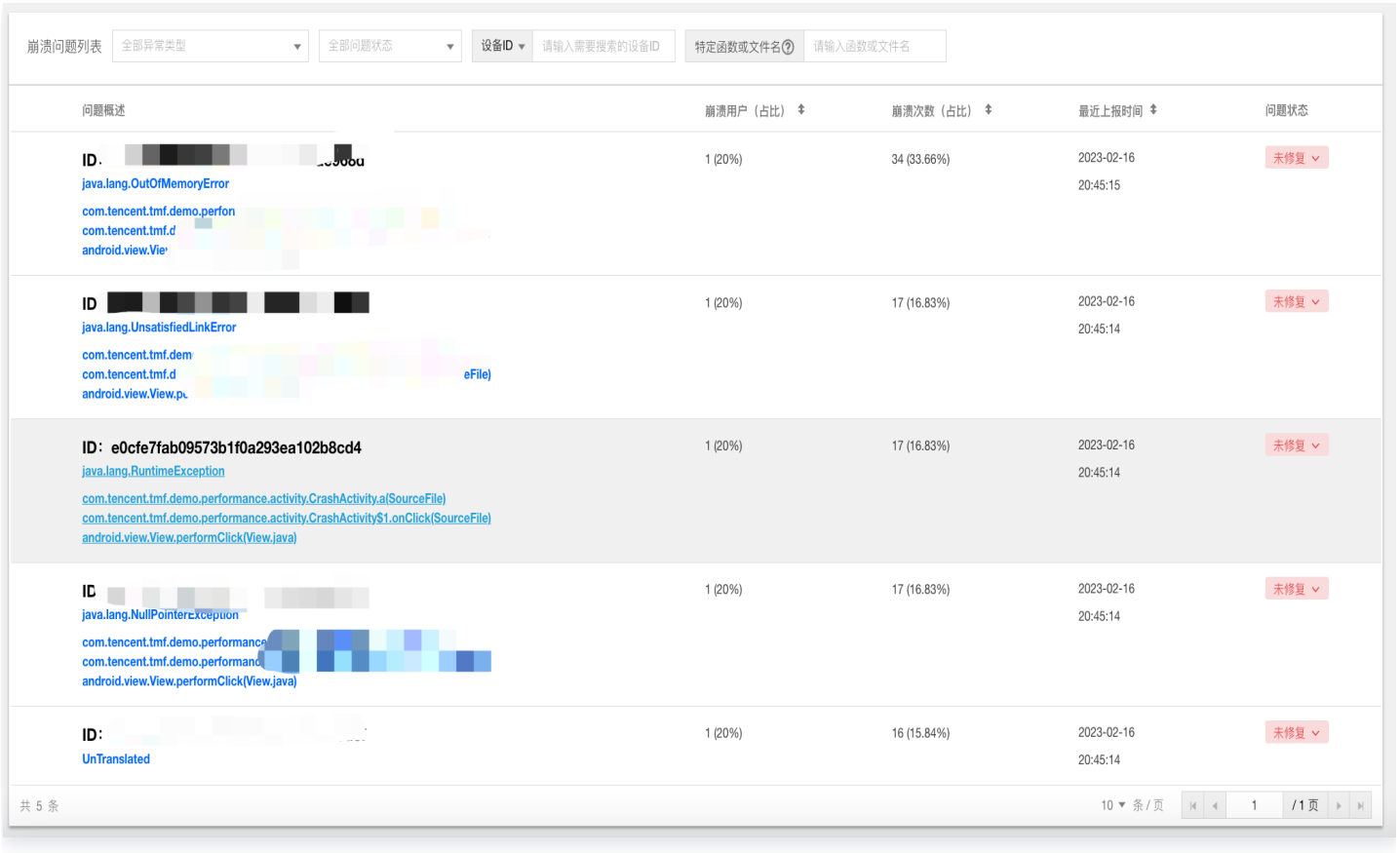
多维分析

多维分析基于应用版本、崩溃类型、系统版本、设备类型、应用状态等多个维度分析关键指标，便于您聚焦的现象进行针对性的崩溃根因分析。



崩溃问题列表

崩溃问题列表展示了所有设备的崩溃问题。您可以根据问题异常类型、问题设备 ID、特定函数或文件名快速筛选相关崩溃问题。您还可以单击**问题概述**查看崩溃问题详情，定位分析崩溃根因。



指标说明

相关指标说明如下表所示：

指标名称	指标说明
崩溃率	指定时间范围内崩溃发生次数/App 启动次数
崩溃用户率	指定时间范围内受到崩溃影响的用户数/启动 App 的用户数
崩溃次数	指定时间范围内崩溃发生次数
崩溃用户数	指定时间范围内受到崩溃影响的用户数
崩溃类型	按照崩溃问题发生位置将崩溃类型分类为 Java 崩溃与 Native 崩溃
SDK 启动次数	应用 SDK 启动次数

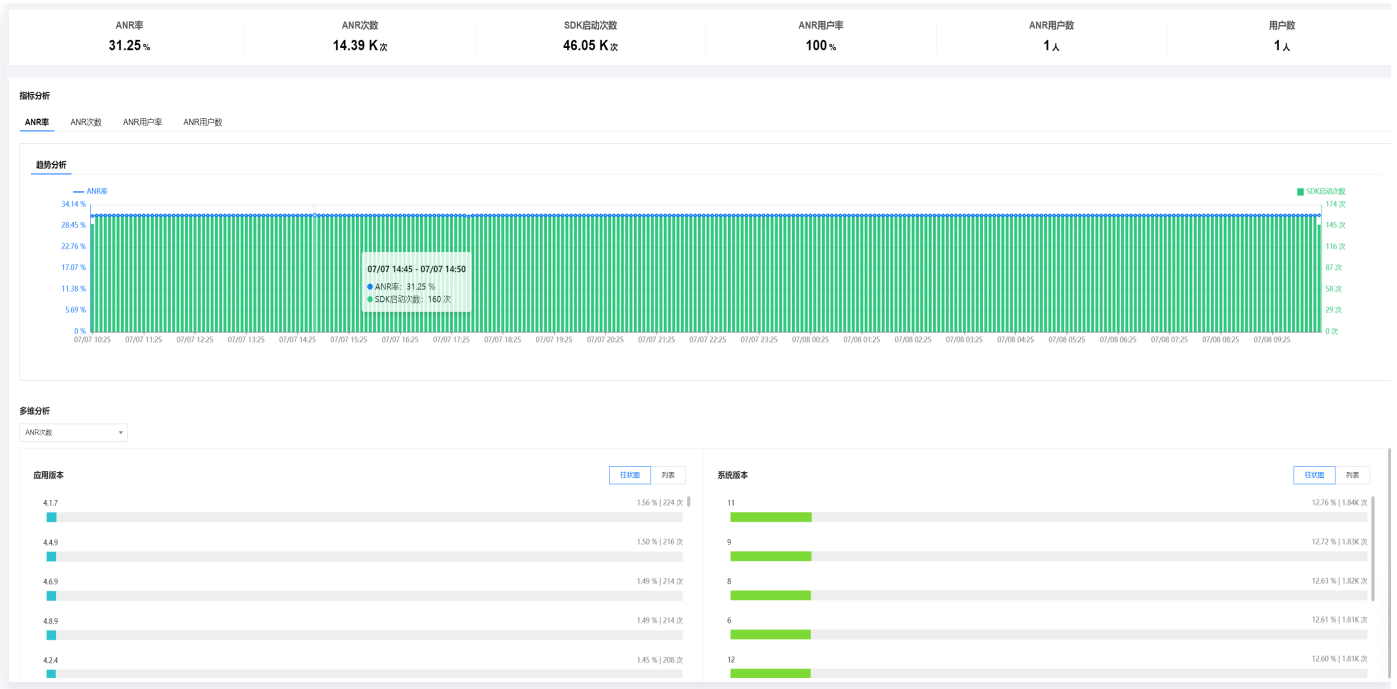
ANR

最近更新时间：2024-07-09 10:29:11

终端性能监控通过对 ANR 问题个例提取关键特征进行聚合，便于您针对 App 的 ANR 问题进行根因分析。

功能入口

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**终端性能监控** > **ANR**。
3. 选择需要查看的业务系统、应用、时间范围等分析 ANR 问题。



多维分析

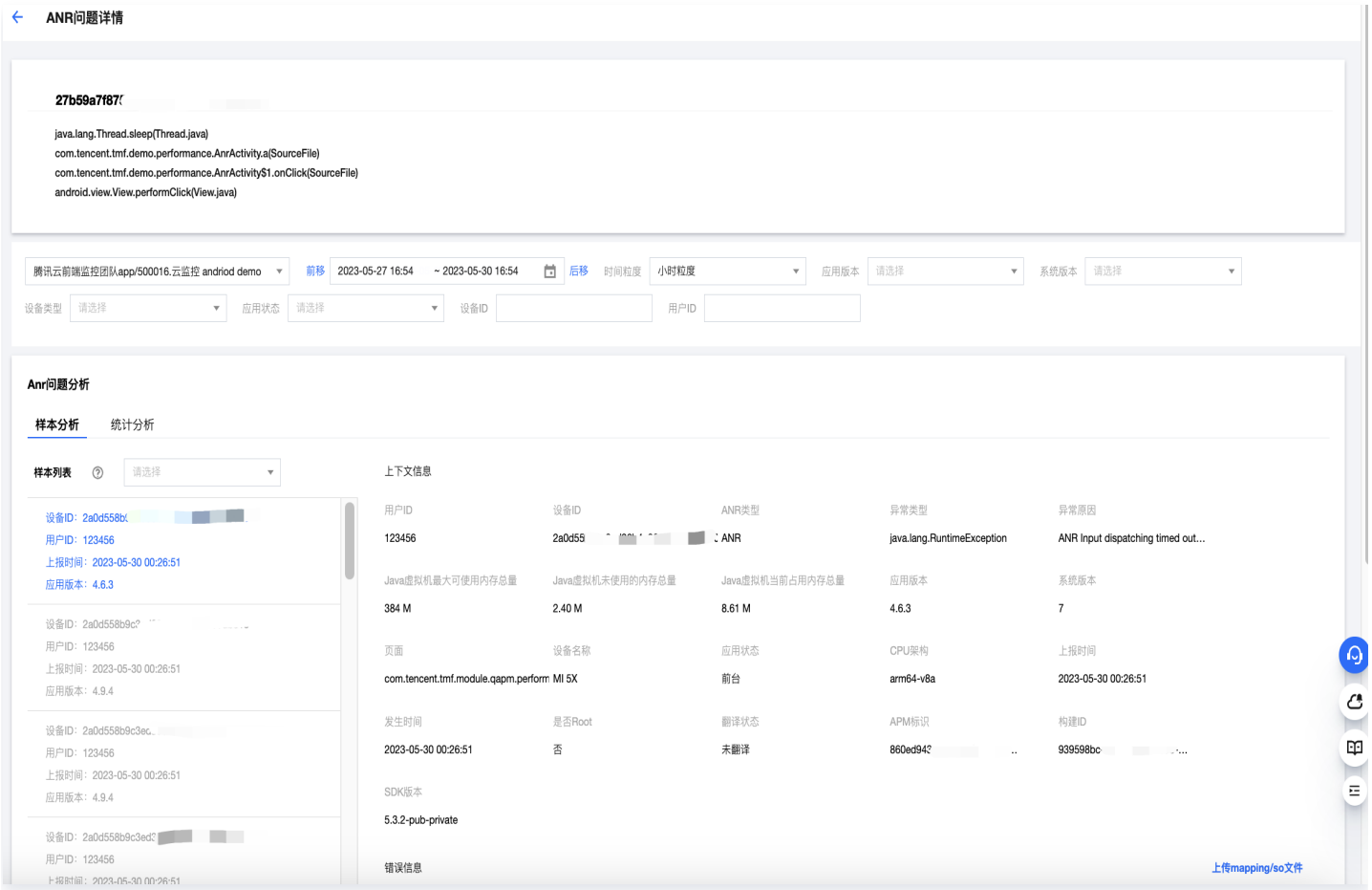
多维分析基于 ANR 次数、SDK 启动次数、 ANR 率多个维度分析关键指标，便于您聚焦现象进行针对性的 ANR 情况根因分析。

ANR 问题列表

ANR 问题列表页提供针对聚类问题的展示、搜索、排序、管理功能，并可通过单击查看详情按钮下钻到问题详情页。

ANR 问题详情

问题详情页提供针对某一类问题的多维统计分析及针对各问题个例的分析功能，您可以单击对应的问题描述进入问题详情。



指标说明

ANR 相关指标说明如下表所示：

指标名称	指标说明
ANR 率	指定时间范围内应用发生 ANR 设备数 / 总设备数
ANR 次数	指定时间范围内应用发生 ANR 的次数
SDK 启动次数	应用 SDK 启动次数
ANR 用户率	指定时间范围内受到 ANR 影响的用户数 / 启动应用总用户数
ANR 用户数	指定时间范围内受到 ANR 影响的用户数
用户数	启动应用总用户数

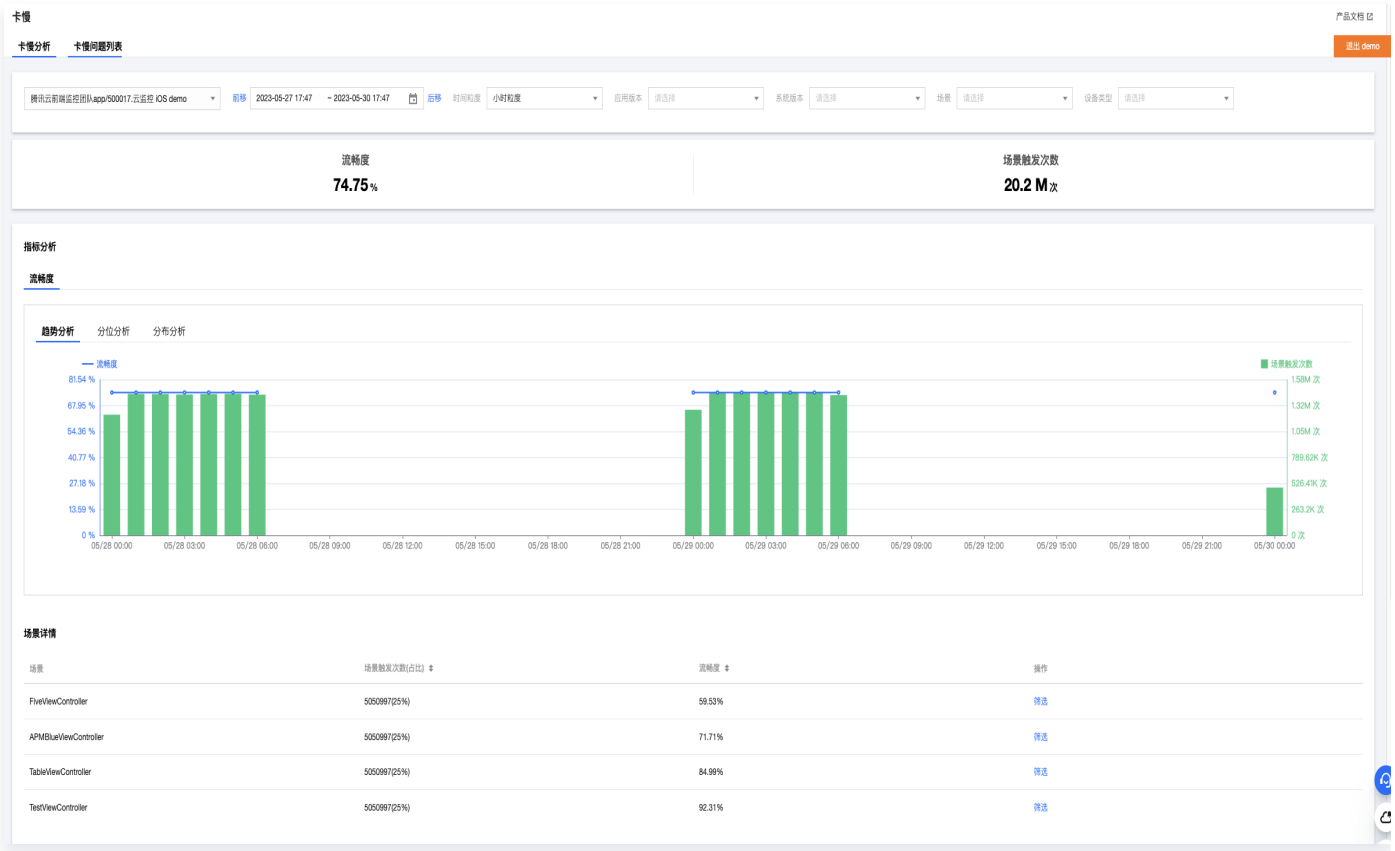
卡慢

最近更新时间：2024-08-16 11:52:01

终端性能监控通过对卡慢问题个例提取关键特征进行聚合，便于您针对 App 的卡慢问题进行根因分析。

功能入口

1. 登录 腾讯云可观测平台 。
2. 在左侧菜单栏中单击**终端性能监控** > **卡慢**。
3. 选择需要查看的业务系统、应用、时间范围等分析卡慢问题。



指标说明

卡慢相关指标说明如下表所示：

指标名称	指标说明
流畅度	流畅页面两帧之间的间隔时间 16.67ms，1000ms（1秒）/16.67ms=60FPS，60FPS 就是理想的流畅的页面。 仅显示占比大于0.1%的数据

场景触发次数	卡慢场景触发次数

启动

最近更新时间：2024-06-07 15:31:51

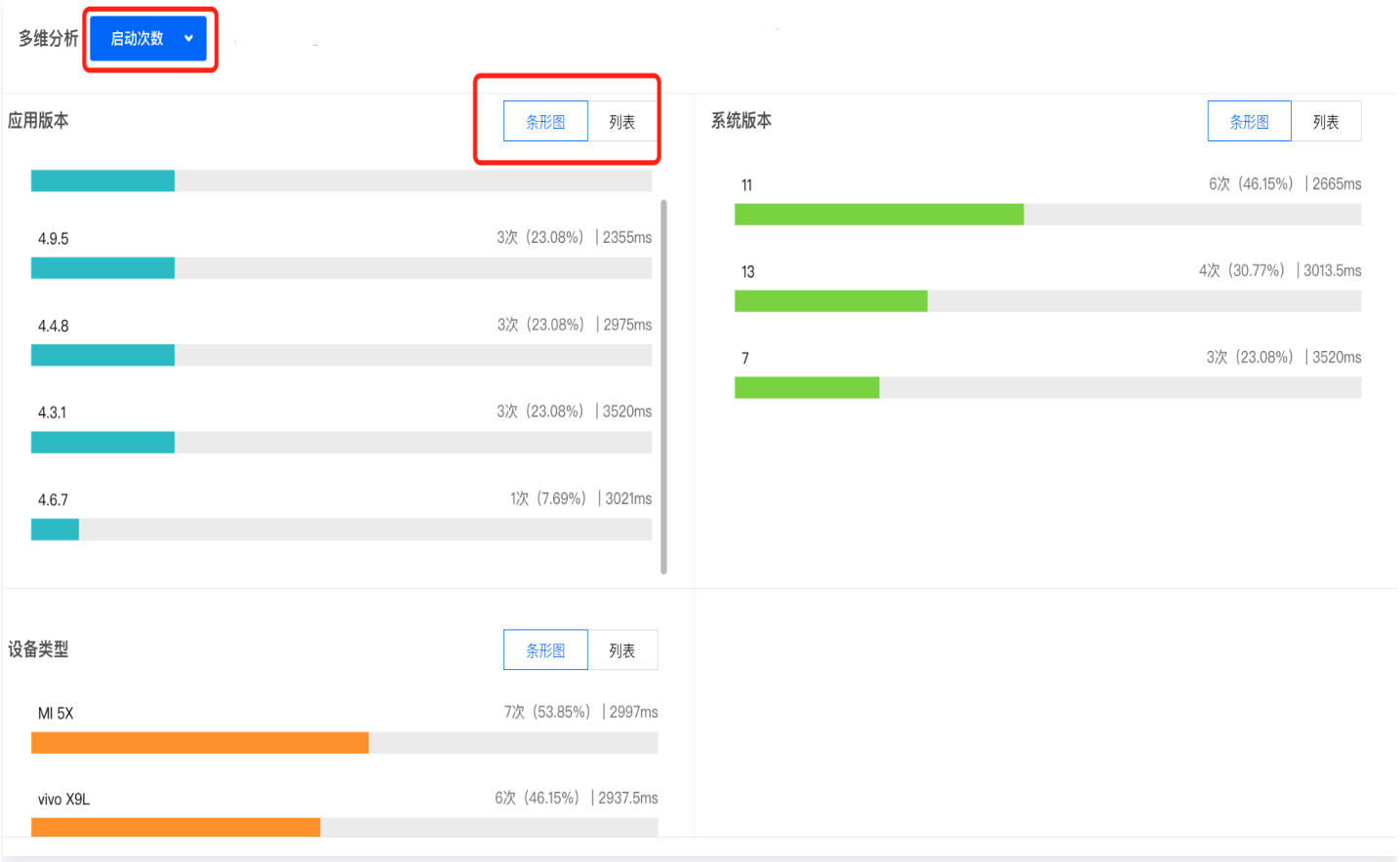
持启动耗时、慢启动占比等进行启动指标分析，您可通过慢启动问题列表定位分析 App 慢启动根因。

功能入口

1. 登录 终端性能监控控制台 。
2. 在左侧导航栏中选择**启动**，选择需要查看的业务系统、应用、时间范围等分析启动问题。

多维分析

多维分析基于应用版本、系统版本、设备类型、地区、运营商等多个维度分析关键指标，便于您聚焦的现象进行针对性的慢启动根因分析。



慢启动问题列表

慢启动问题列表展示了所有设备的慢启动问题。您可以根据问题状态、问题设备 ID、特定函数或文件名快速筛选相关慢启动设备。您还可以单击**问题概述**查看慢启动问题详情，定位分析 App 慢启动根因。

对于每一个应用冷启动/首次启动样本，Android 系统若启动耗时大于2500ms 为慢启动，iOS 系统若启动耗时大于4000ms 为慢启动；对于每一个应用热启动样本，Android 系统若启动耗时大于 1000ms 为慢启动，iOS 系统若启动耗时大于2000ms 为慢启动，仅慢启动样本会显示在问题列表中。

指标说明

相关指标说明如下表所示：

指标名称	指标说明
慢启动次数	应用慢启动次数。
启动耗时	启动应用耗时。
平均耗时	线上样本启动耗时之和/应用启动次数。
慢启动占比	发生慢启动的次数/总启动次数。 • 在 Android 端，启动耗时超过2.5s 被默认定义为慢启动。 • 在 iOS 端启动耗时超过4s 被默认定义为慢启动，阈值可自定义。
首次启动	App 安装后的第一次启动，属于特殊的冷启动。
冷启动	App 结束进程，或退出到后台，进程被系统回收后，再次启动的过程。
热启动	App 切换后台3min 后，从后台被唤起，或从其他 App 界面切换回来的过程。

网络

最近更新时间：2024-06-07 15:31:51

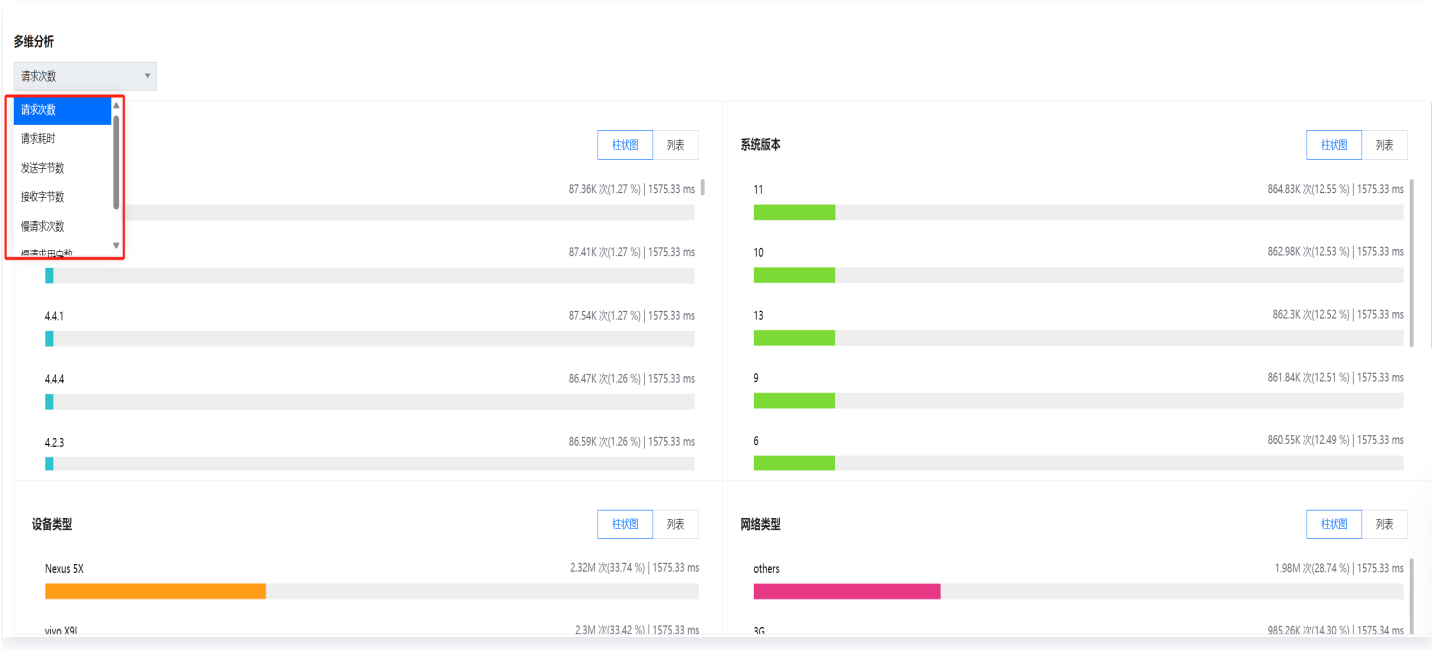
通过吞吐量、请求次数、网络响应时间、慢请求占比、HTTP 错误率、网络错误率、TCP 建连时间等指标进行网络问题分析。

功能入口

1. 登录 终端性能监控控制台 。
2. 在左侧导航栏中选择**网络**，从业务系统、应用、时间范围等多个维度分析网络问题。

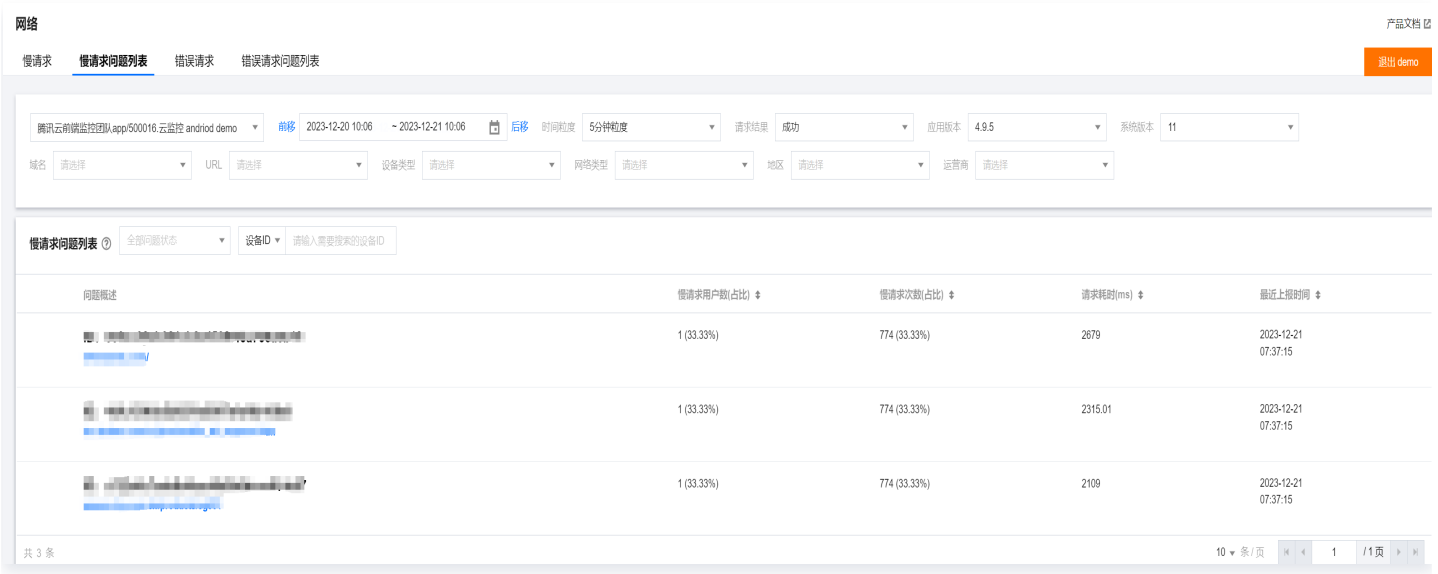
慢请求、错误请求多维分析

多维分析基于应用版本、系统版本、域名、URL、设备类型、网络类型、运营商和地区等多个维度分析关键指标，便于您聚焦的现象进行针对性的慢请求/错误请求根因分析。

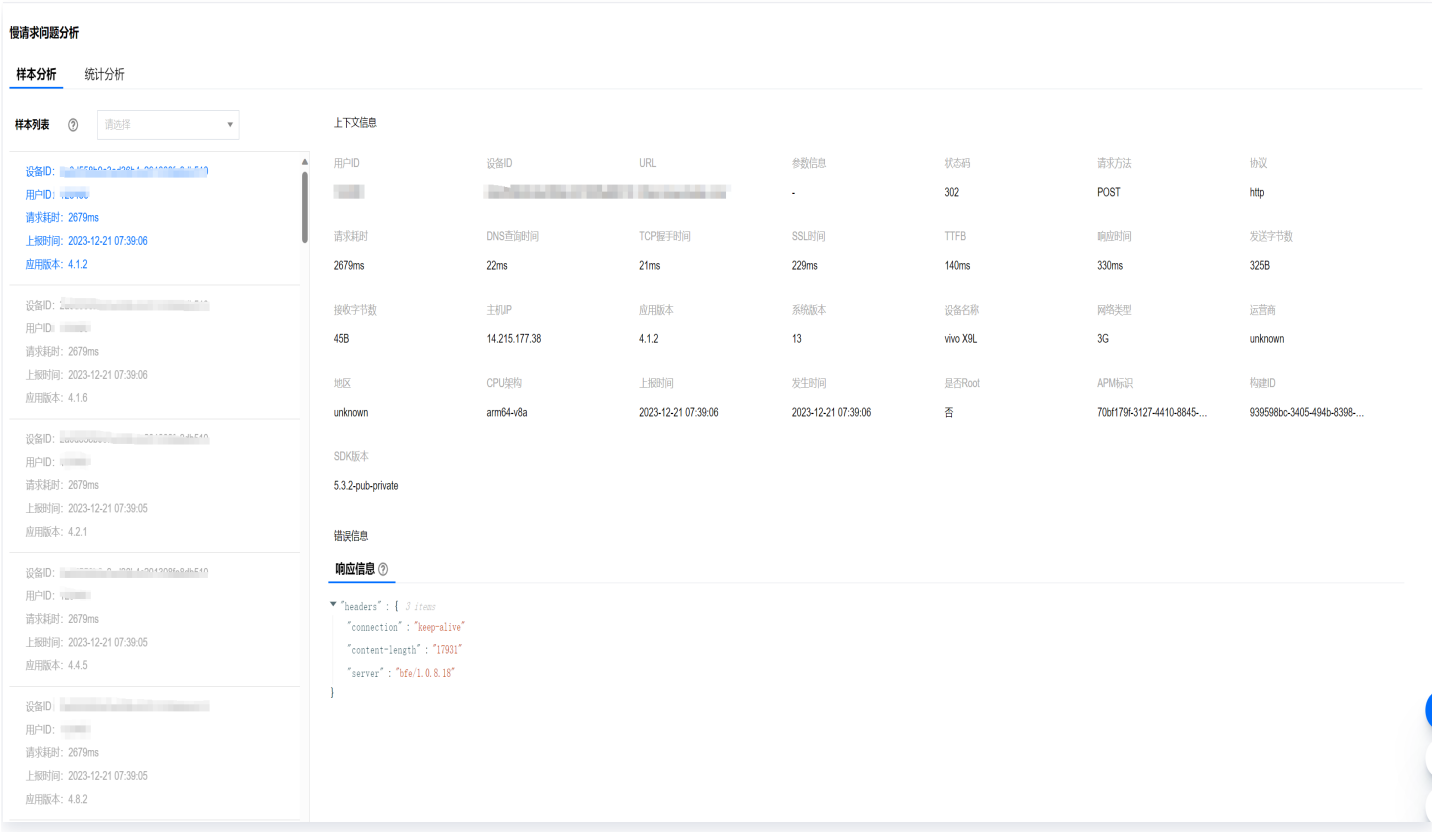


慢请求问题列表

慢请求问题列表展示了所有设备的慢请求问题。您可以根据问题状态和问题设备 ID 快速筛选相关慢加载设备。您还可以单击**问题概述**下的相关链接查看慢请求问题详情，定位分析 App 慢请求根因。

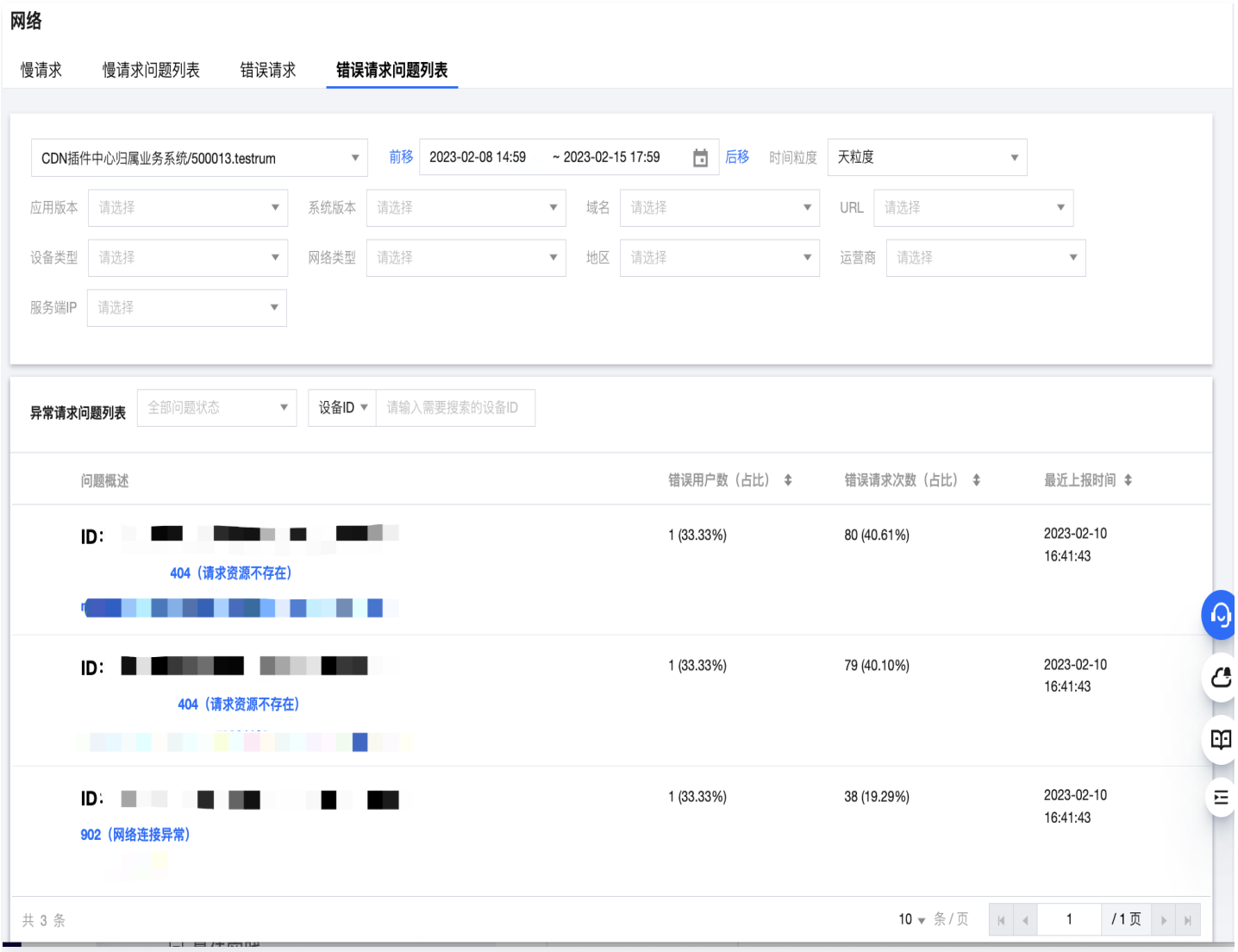


对于每一个 HTTP 请求样本，传输数据大于 50KB 时 ，传输速度小于 10KB/s 为慢请求；若传输数据小于等于 50KB，响应时间大于 2s 为慢请求，这些慢请求样本会显示在问题列表中。



错误请求问题列表

发生 HTTP 请求错误、 DNS 解析错误、无法建连、连接超时等网络方面的错误。将会展示在错误请求列表中，您可以单击**问题概述**下的相关链接，查看错误请求问题详情，定位分析错误请求根因。



指标说明

相关指标说明如下表所示：

指标名称	指标说明
请求耗时	应用请求耗时
慢请求比例	慢请求占比为选定时间段内，慢请求数量与访问量的比值 • 传输数据大于50KB时，传输速度小于10KB/s为慢请求 • 若传输数据小于等于50KB，响应时间大于2s为慢请求
慢请求次数	慢请求次数为选定时间段内的慢请求数量

	• 传输数据大于50KB时，传输速度小于10KB/s为慢请求 • 若传输数据小于等于50KB，响应时间大于2s为慢请求
请求次数	应用请求总次数
慢请求用户比例	指定时间范围内受到慢请求影响的用户数与总用户的比值
慢请求用户数	指定时间范围内受到慢请求影响的用户数
请求错误率	请求错误次数/总请求次数
错误请求次数	在选定时间段内，出现网络错误的数量 错误请求指发生 HTTP 请求错误、DNS 解析错误、无法建连、连接超时等网络方面的错误
错误用户比例	指定时间范围内受到错误请求影响的用户数与总用户的比值
错误用户数	指定时间范围内受到错误请求影响的用户数

Webview

最近更新时间：2024-06-07 15:31:51

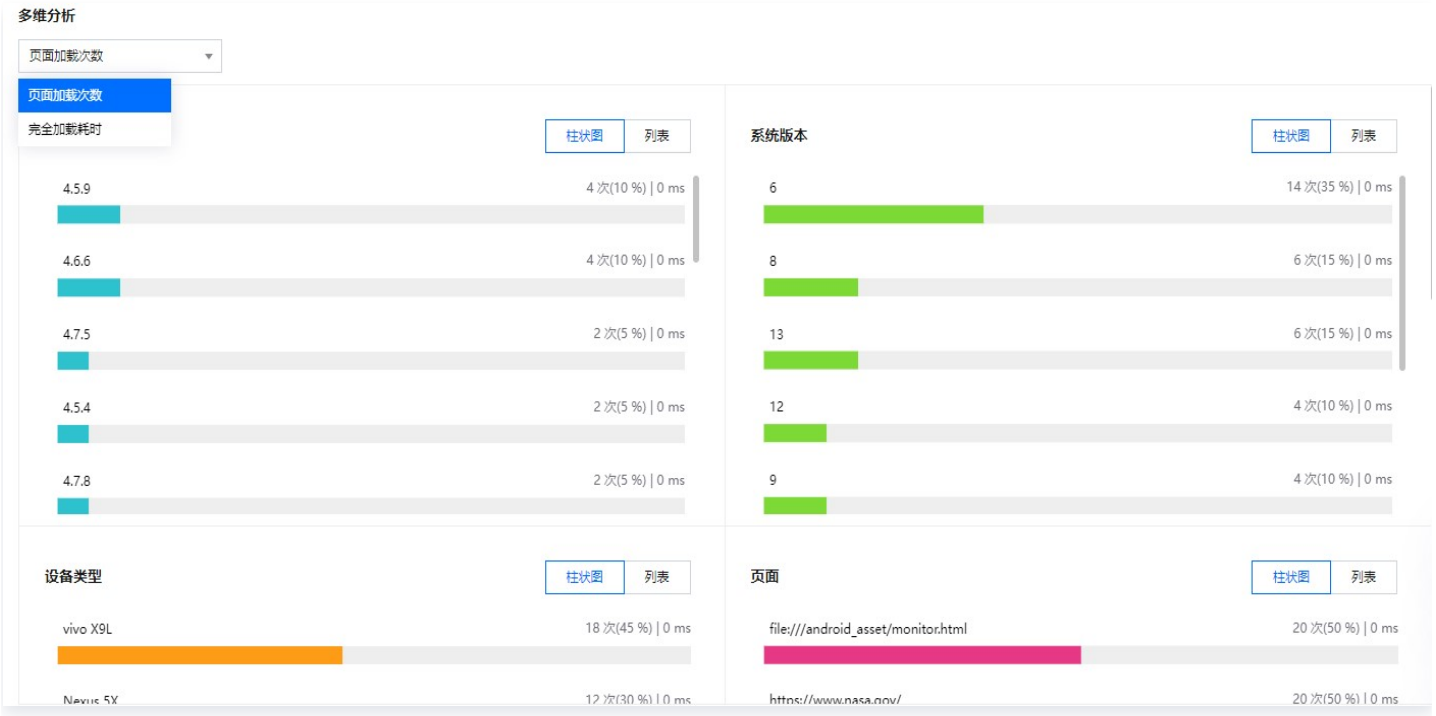
通过页面加载耗时、慢加载占比以及 JS 错误率等进行 WebView 指标分析，并通过问题列表对 WebView 以及 JSError 问题进行下钻。

功能入口

1. 登录 终端性能监控控制台 。
2. 在左侧导航栏中选择 **Webview**，选择需要查看的业务系统、时间范围、应用版本、系统版本、设备类型等来分析 Webview 问题。

慢加载、JS 错误多维分析

多维分析基于应用版本、系统版本、设备类型、页面、网络类型、运营商和地区等多个维度分析关键指标，便于您聚焦的现象进行针对性的慢加载/ js 错误根因分析。

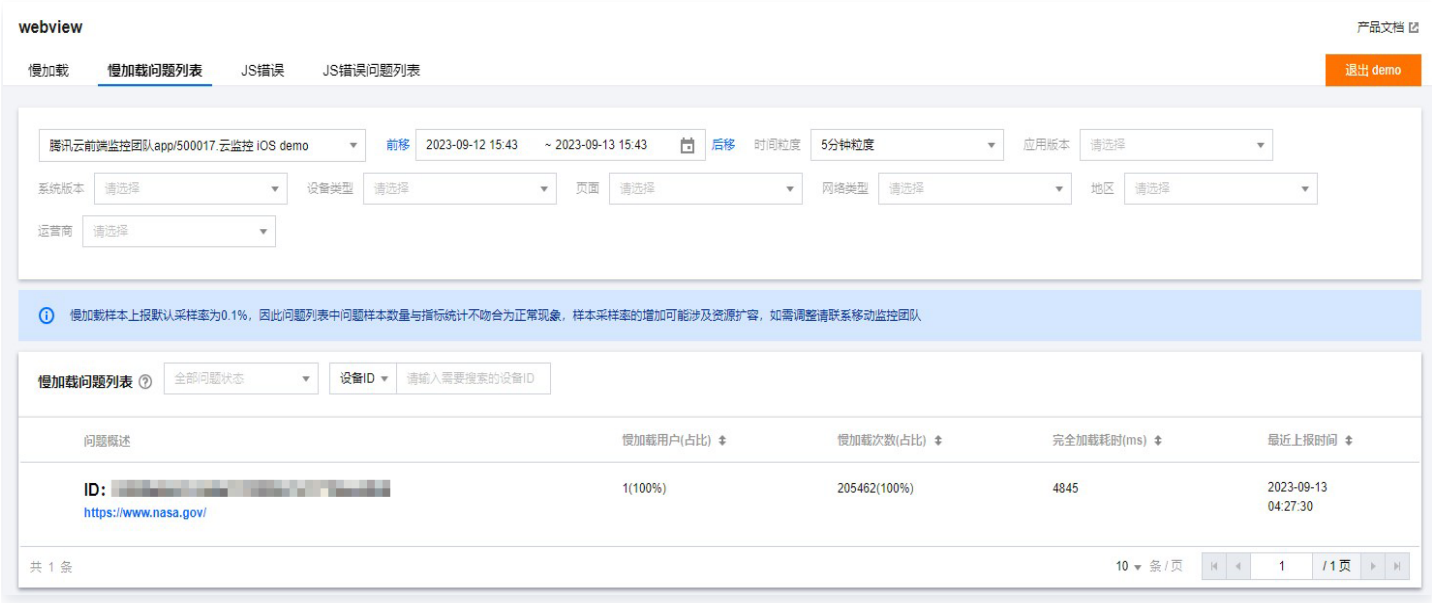


慢加载问题列表

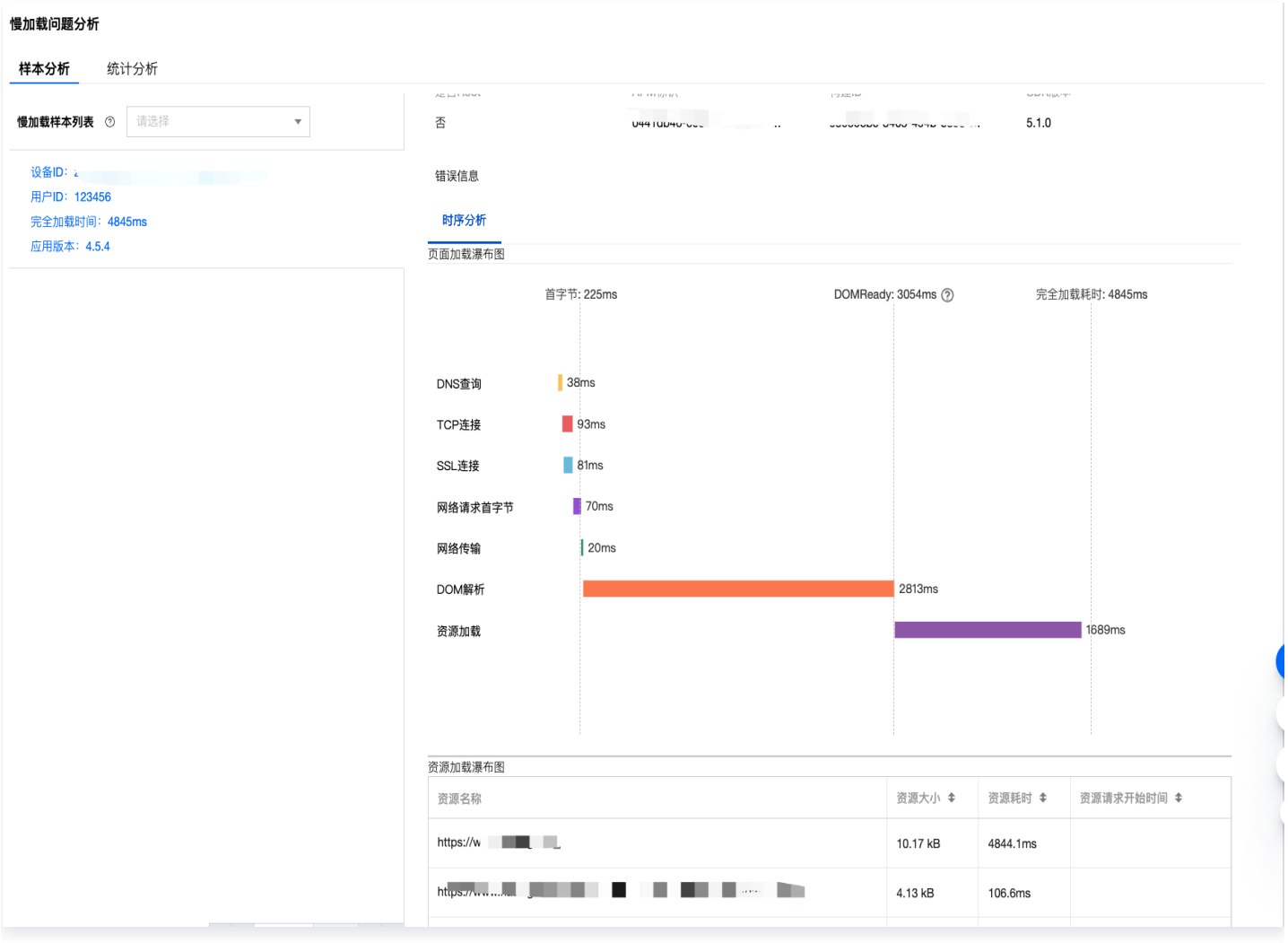
慢加载问题列表展示了所有设备的慢加载问题。您可以根据问题状态、问题设备 ID 快速筛选相关慢加载设备。您还可以单击**问题概述**查看慢加载问题详情，定位分析 App 慢加载根因。

> **说明：**

> 慢加载样本上报默认采样率为0.1%，因此问题列表中问题样本数量与指标统计不吻合为正常现象。



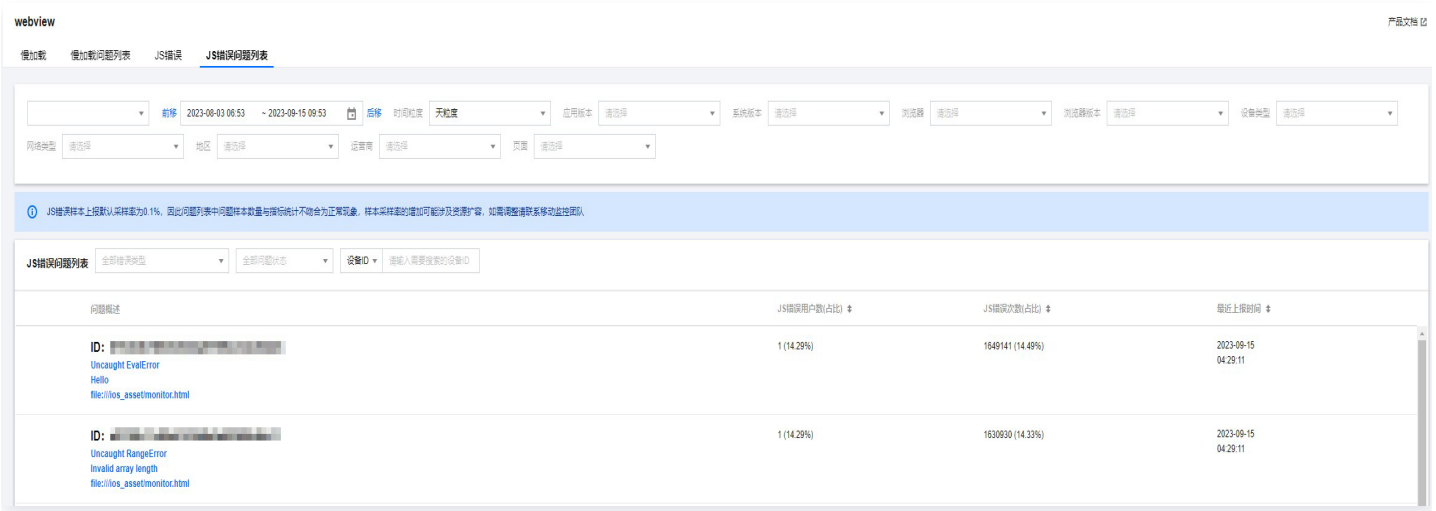
对于每一个页面加载样本，页面完全加载时间大于3500ms 为慢加载，这些慢加载样本会显示在问题列表中。



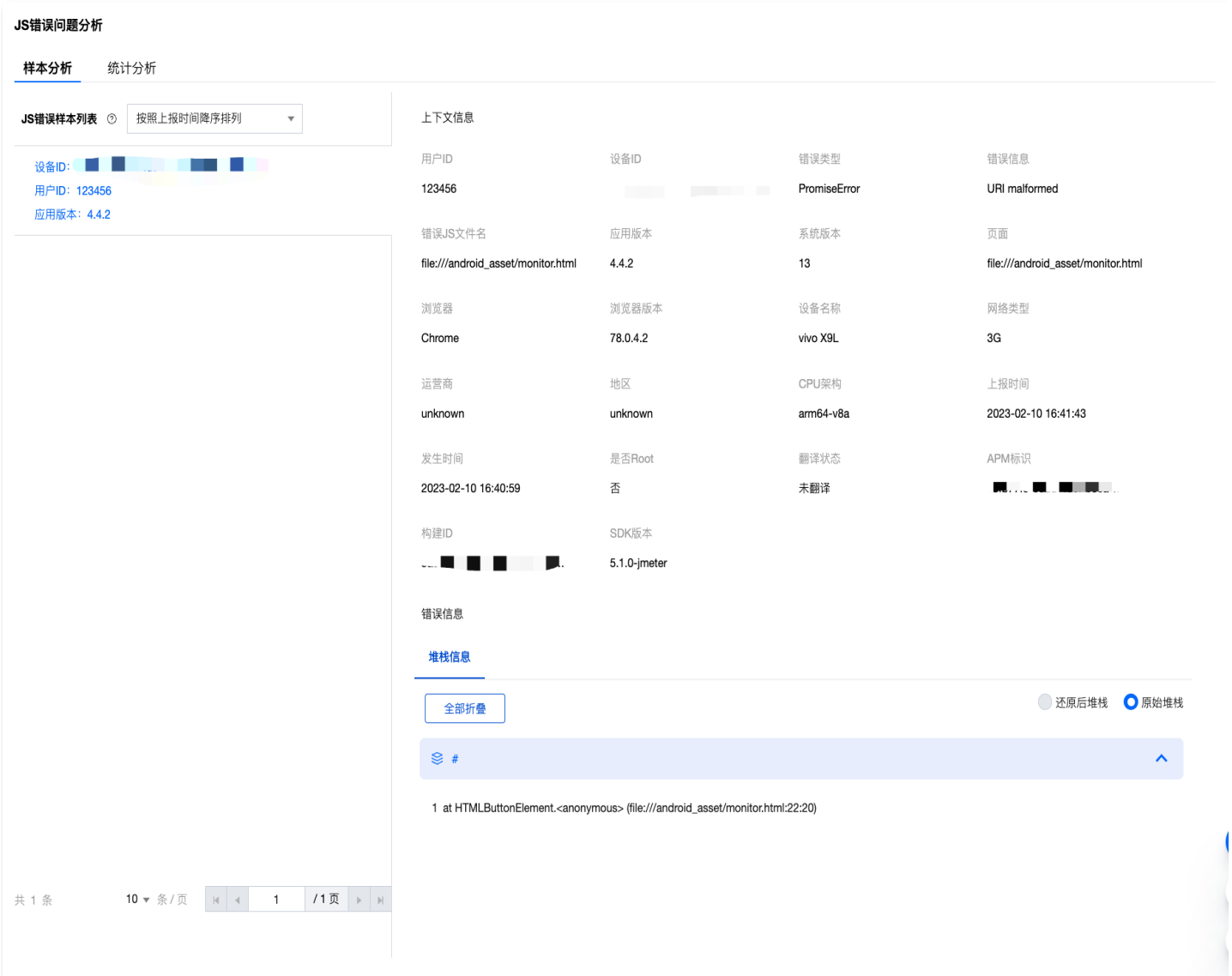
JS 错误问题列表

您可以在 JS 错误问题列表查看所有 JS 报错。

> **说明：**
> JS 错误问题上报默认采样率为0.1%，因此问题列表中问题样本数量与指标统计不吻合为正常现象。



您还可以单击问题概述查看 JS 错误问题详情，定位分析 JS 错误原因。



指标说明

相关指标说明如下表所示：

指标名称	指标说明
页面加载次数	打开或刷新页面的次数
完全加载耗时	指整个网页完全加载的时间
JS 错误次数	指定时间内的 JS 错误总数
JS 错误率	发生 JS 错误的用户数/访问 webview 页面的总用户数，由于计算资源限制，该指标分子及分母为未去重数据
JS 错误用户比例	指定时间范围内受到 JS 错误影响的用户数/访问 webview 页面的总用户数
JS 错误用户数	指定时间范围内受到 JS 错误影响的用户数

应用管理

最近更新时间：2024-06-07 15:31:51

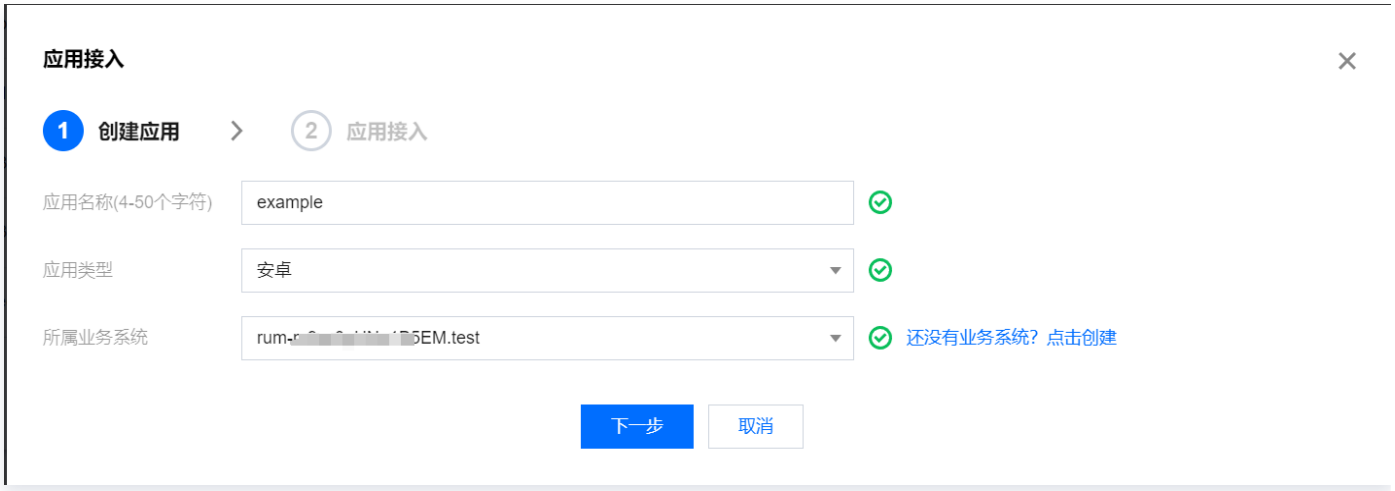
操作场景

在应用管理页您可以查看您已接入的终端应用信息，以及新的接入应用、应用 ID 、设置白名单信息。

操作步骤

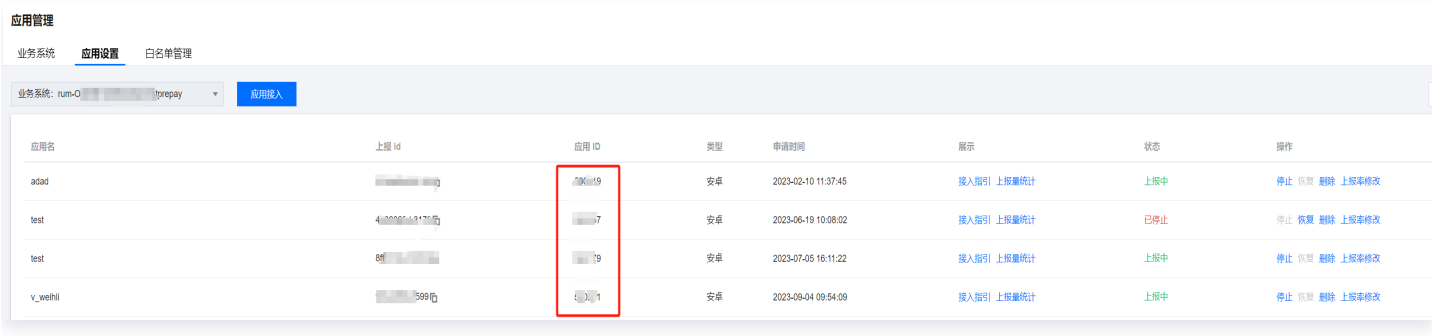
接入应用

1. 登录 终端性能监控控制台 。
2. 选择**应用管理 > 应用设置**。
3. 在**应用设置**页面单击**应用接入**，填写应用名称、选择 Android 或 IOS 应用类型、并选择相关业务系统。单击**下一步**。
4. 参见 IOS 接入 文档或 Android 接入文档 接入应用。



获取应用 ID

您可以在**应用管理 > 应用设置**页面，在应用设置列表中获取应用 ID 。



白名单配置

您可以在应用管理 > 白名单管理页面，点击白名单配置下方的添加。通过配置用户 ID/设备 ID 白名单可以避免数据上报受采样影响。即加入白名单的用户/设备不受采样率影响，将会全部采样。您可以选择用户或者设备类型，并填写相关 ID 即可。

全链路监控

最近更新时间：2024-06-26 14:23:32

全链路监控可通过 trace_id 将用户请求从客户端各个阶段到服务端各个服务节点之间的调用关系记录下来，形成一个完整的请求链路，帮助开发者和运维人员快速定位关键请求的性能瓶颈，提高系统的故障排除效率。

前提条件

- 已在终端性能监控 APP 服务中完成 应用接入 。
- 已在应用性能监控 APM 服务中完成 应用接入 。

> ⚠ 注意：
> - 目前仅支持通过 OpenTelemetry/SkyWalking 协议接入 APM 的应用实例与 APP 的应用实例进行全链路打通。应用语言无限制，请根据业务服务应用实际使用的语言完成 APM 接入。
> - 为了确保验证数据的上报不受网络模块默认采样（0.1%）的影响，可在 终端性能监控 > 应用管理 > 白名单管理 中，为测试设备配置白名单，白名单中的设备采集到的性能数据将会全量上报，配置步骤请参见 应用管理 – 白名单配置 。

接入步骤

步骤1：在终端性能监控 SDK 中开启网络监控功能

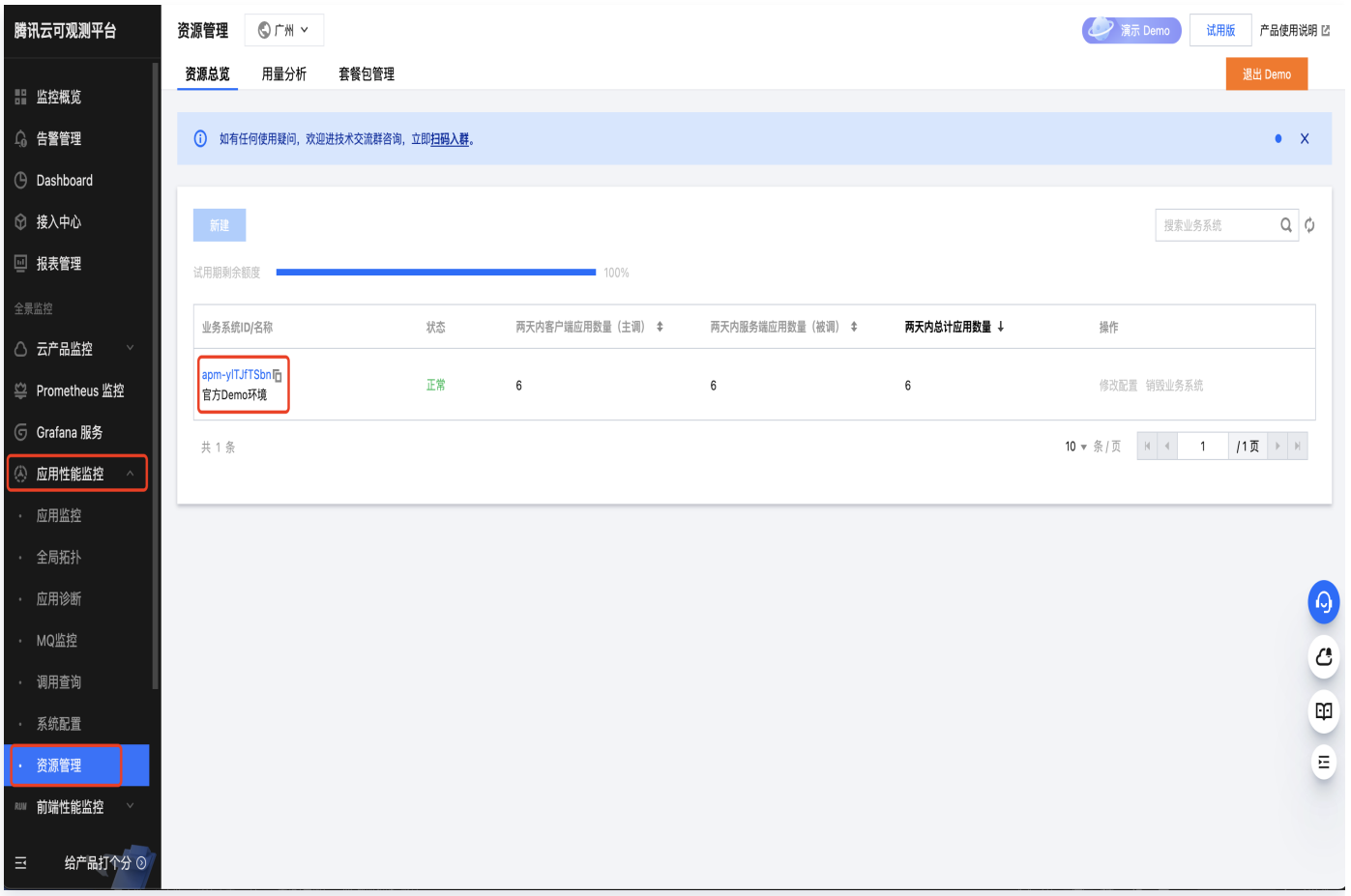
详情请参见 安卓应用场景 > 网络监控 与 iOS 应用场景 > 监控功能启用 。

步骤2：在终端性能监控 SDK 中开启全链路监控功能

详情请参见 安卓应用场景 > 初始化 SDK > 全链路监控开关 与 iOS 应用场景 > 初始化 SDK > 全链路监控开关 。

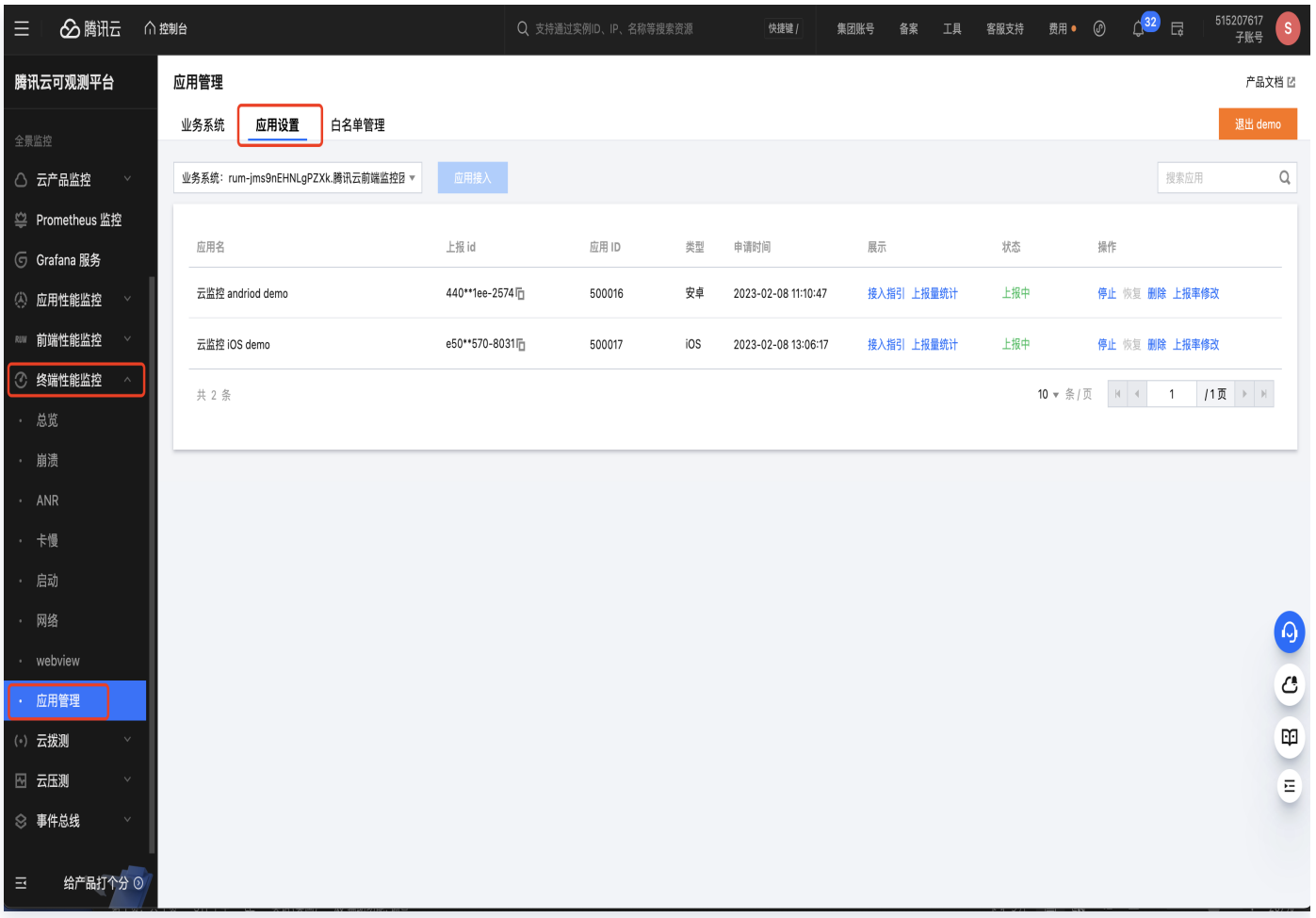
步骤3：获取应用性能监控业务系统ID参数

1. 登录 应用性能监控 。
2. 侧边栏点击**资源管理 > 资源总览**。
3. 选择您需要和终端性能监控关联的业务系统并复制其业务系统 ID。

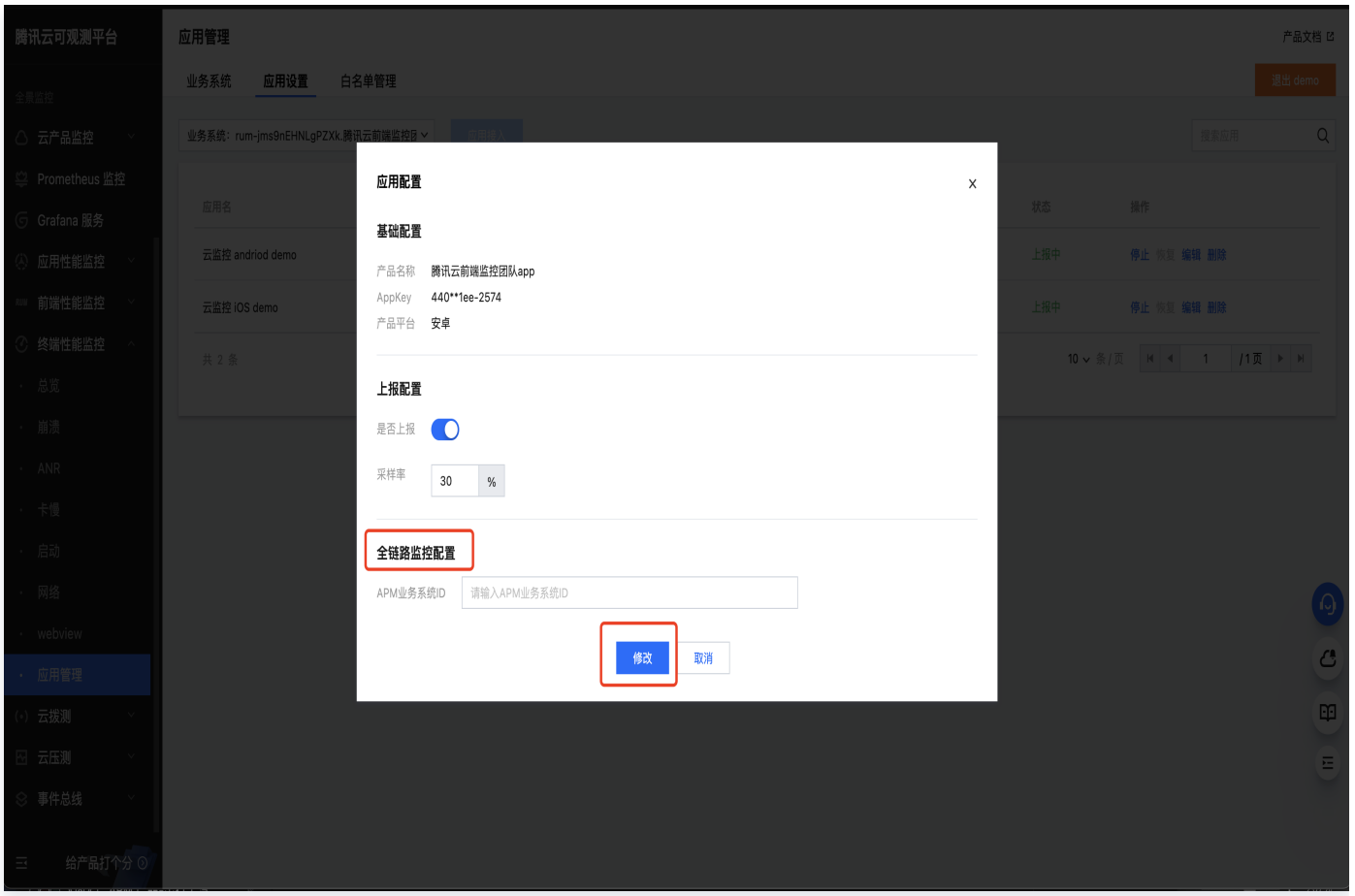


步骤4：完成客户端与服务端实例关联

1. 侧边栏切换到**终端性能监控**，点击**应用管理 > 应用设置**。



2. 选择您需要和应用性能监控关联的终端应用并点击编辑操作，将步骤 3.3 中获取的 APM 业务系统 ID 填入，完成配置修改即可实现实例关联。

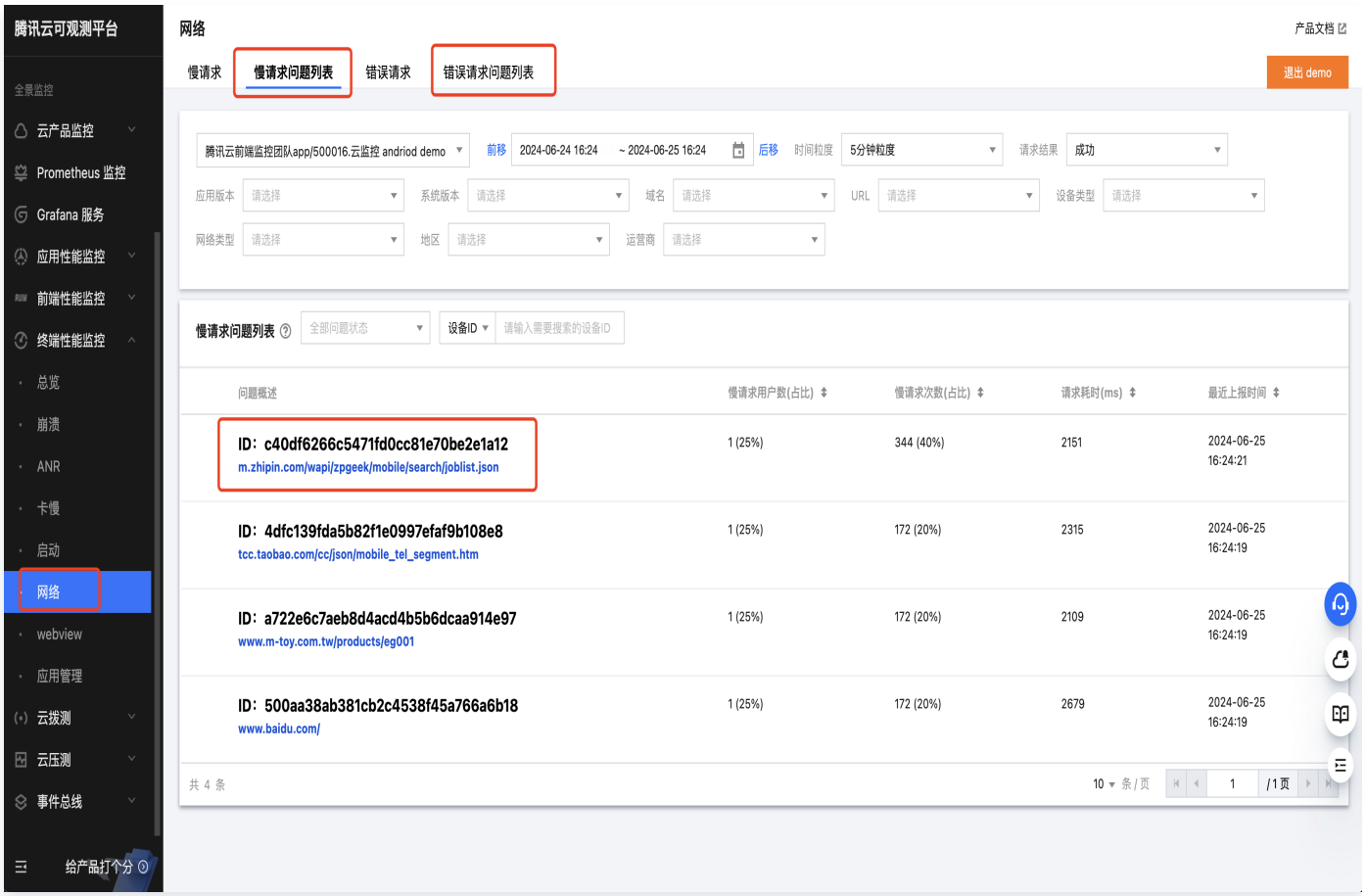


步骤5：（可选）严格服务端安全策略下的放行

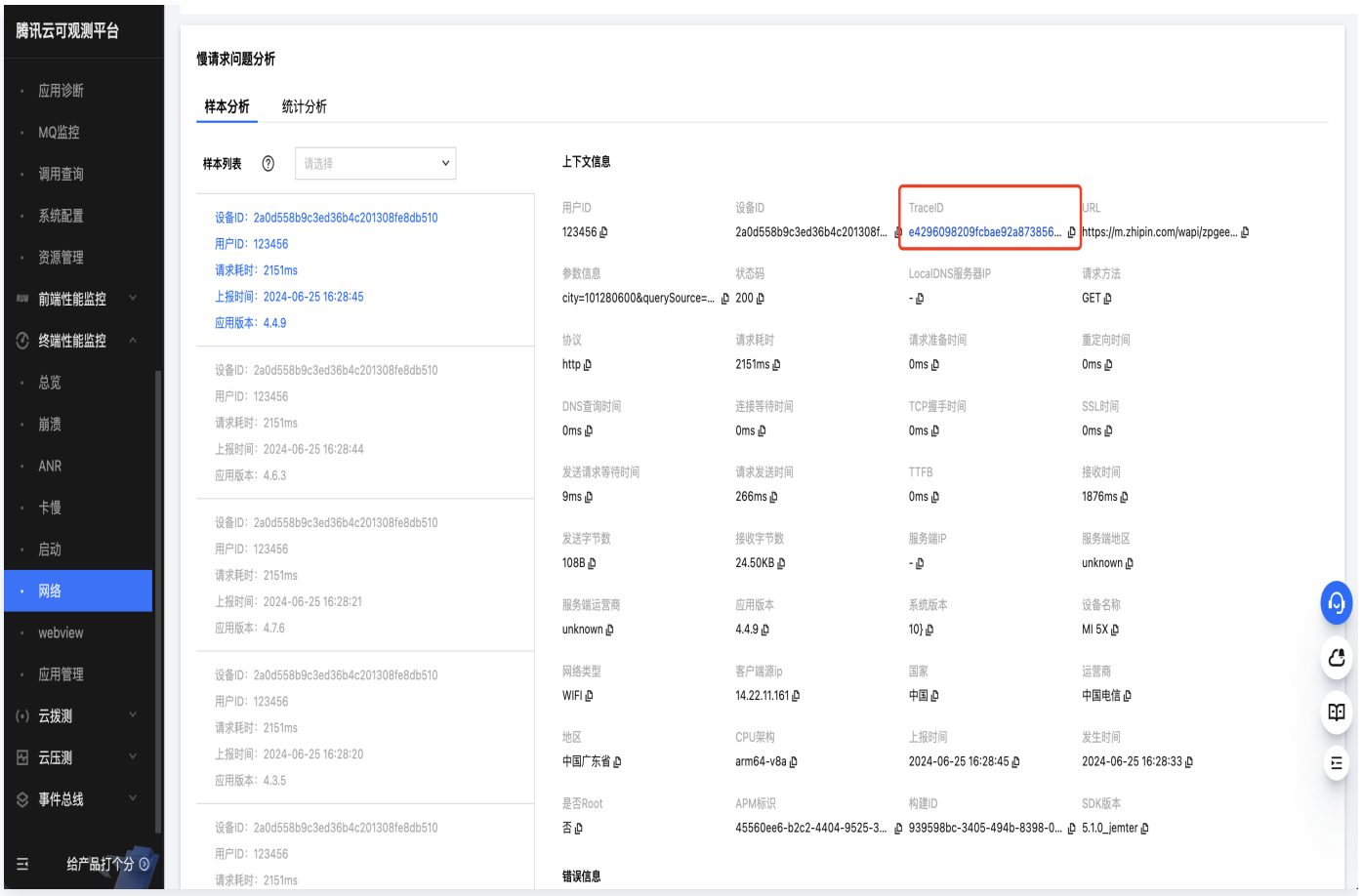
为了实现全链路 trace 的关联，根据您使用的协议（OpenTelemetry/SkyWalking），我们会在您的客户端请求服务端的 header 里面添加 transparent/sw8 字段。若您的服务端采用严格字段协议校验等安全策略，我们添加的字段可能会导致客户端请求服务端失败，我们主张您在开展全链路打通验证前与服务端同事针对该场景进行沟通，确保服务端对这些添加了 transparent/sw8 字段的请求进行放行。

步骤6：全链路打通验证

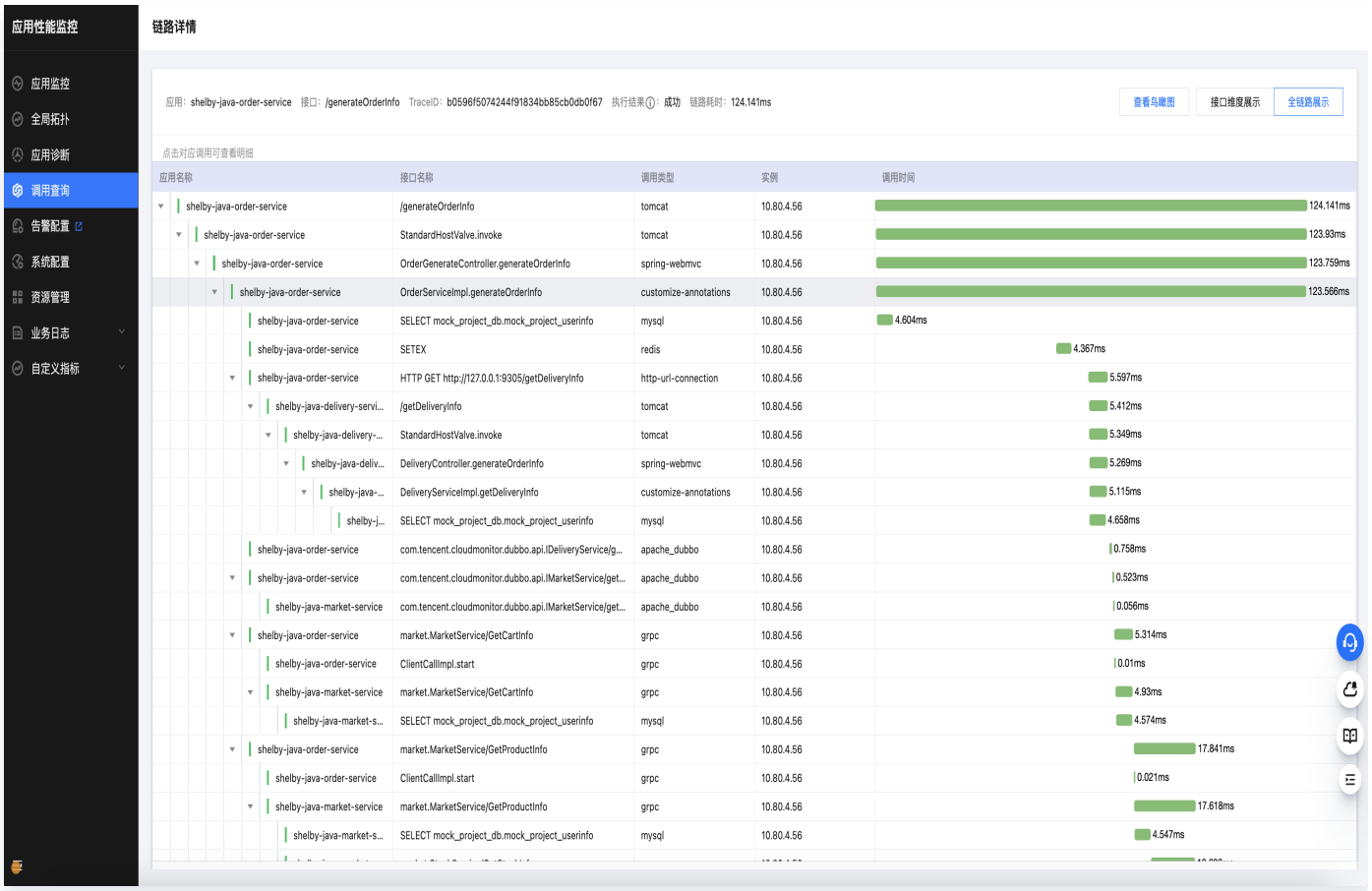
1. 上报慢请求/异常请求测试数据。
2. 侧边栏点击**网络**，在**慢请求问题列表**与**异常请求问题列表**页签中可查看问题信息。



3. 点击新上报的问题列表行下钻问题详情，在上下文模块可查看到 trace_id。



4. 点击跳转下钻到 APM 产品的链路追踪详情中，则全链路打通验证成功。



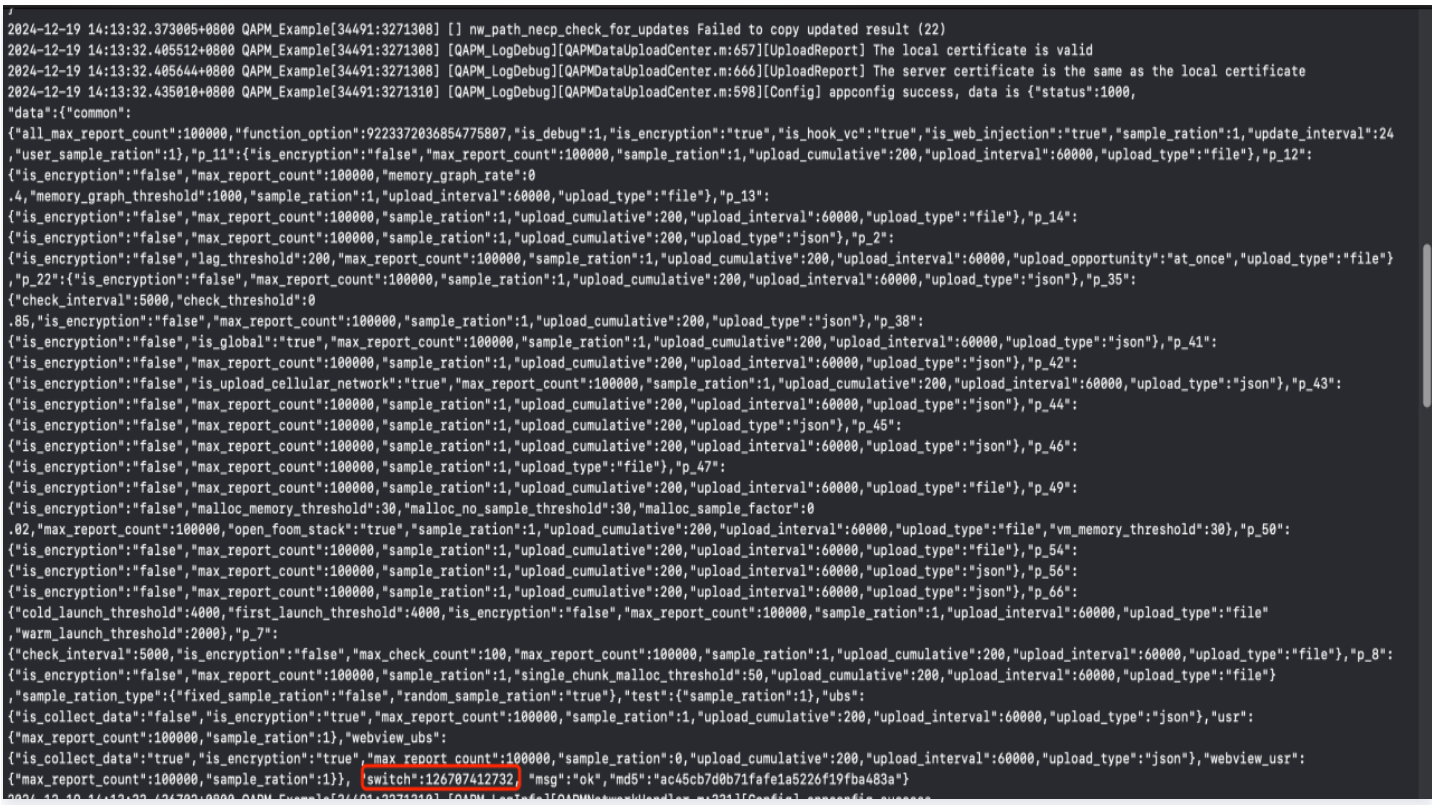
常见问题

最近更新时间：2024-12-27 11:58:52

接入 SDK 之后，为什么 web 端页面没有对应的数据信息上报?

请先检查初始化日志信息。可以触发示例 Demo，根据接入文档的上报日志信息查看对应信息，开启所有功能的首次初始化日志如下：


```
2024-12-19 14:13:32.011197+0800 QAPM_Example[34491:3271251] qapm sdk version : 5.4.4
2024-12-19 14:13:32.011396+0800 QAPM_Example[34491:3271251] [QAPM_LogDebug][QAPM.m:52][Config] setting host: https://app.rumt-zh.com
2024-12-19 14:13:32.011967+0800 QAPM_Example[34491:3271251] [QAPM_LogEvent][QAPMPathUtil.m:99][Config]
getCertificateUrl::https://app.rumt-zh.com/entrance/v1/requestCer/?app_key=d8165f39&p_id=5105&deviceid=qwerdf&uin=qwerdf&version=2.4.42-qapm-sdk-debug&sdk_ver=5.4
.4&device=iPhone10,3&os=Version 16.4 (Build 20E247)&platform=ios&cer_code=
2024-12-19 14:13:32.284462+0800 QAPM_Example[34491:3271310] [QAPM_LogDebug][QAPMDataUploadCenter.m:598][Config] EncodeCerString success, data is
{"EncodeCerString":"MIIGjzCCBXegAwIBAgIQAnFssUB2t4mTQqULrHAliTANBgkqhkiG9w0BAQsFADBMMQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMSYwJAYDVQQDEx1EaWdpQ2VydCBTZWN1cmUgU2l0ZSBDTiBDQSBHMzAeFw0
yNDAxMTYwMDAwMDBaFw0yNTAxMzEyMzU5NTlaMIGNMQswCQYDVQQGEwJDTjEbMBkGA1UECBMSR3Vhbmdkb25nIFByb3ZpbmNlMREwDwYDVQQHEwhTaGVuemhlbjE2MDQGA1UEChMtVGVuY2VudCBUZWNobm9sb2d5ICh
TaGVuemhlbikgQ29tcGFueSBMaW1pdGVkMRYwFAYDVQQDDA0qLnJ1bXQtemguY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxxq/K9fG3hNM+/ehrWAVZysWmfiALRp5bGQB6486xOU162BLzy50CQ/YKS6ObbqChvVd7Z8uvzwKs1/edL0QSgJfBojieRku0OWOLBjvg+h5O
XV5SOWye1ZxxN5Pjz6DXqdsxt3Piv41E1YRyuJXPFcy6t0+yuvlrir+FJ2bqNDGu1s1P/2BfLkmyg0L4mBm2cBc65gBLjQFQDkucGmiQAPtJEF7XxQrc0pQS0Lj/8CYMl6WBlev625UqWfGEPG3HKxUONxovsFOvj6llgvfAB7Y0aIqjuy7oKf11BK4BG8i
6rl38+mBqo7OO8WsdSORGiM6NXUmu85rdB7TrneJBQIDAQABo4IDKTCCAyUwHwYDVR0jBBgwFoAURNnISjOO01KNp5KUYR+ayKW37MswHQYDVR0OBBYEFK1h4NSETH+Wkkzrtd9/MkUhs56PMCUGA1UdEQQeMByCDSoucnVtdC16aC5jb22CC3J1bXQtemg
uY29tMD4GA1UdIAQ3MDUwMwYGZ4EMAQICMCkwJwYIKwYBBQUHAgEWG2h0dHA6Ly93d3cuZGlnaWNlcnQuY29tL0NQUzAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMEQGA1UdHwQ9MDswOaA3oDWGM2h0dHA6Ly9jcm
wuZGlnaWNlcnQuY24vRGlnaUNlcnRTZWN1cmVTaXRlQ05DQUczLmNybDB4BggrBgEFBQcBAQRsMGowIwYIKwYBBQUHMAGGF2h0dHA6Ly9vY3NwLmRpZ2ljZXJ0cy5kaWdpY2VydC5jbi9EaWdpQ2Vyd
FNlY3VyZVNpdGVDTkNBRzMuY3J0MAwGA1UdEwEB/wQCMAAwggF9BgorBgEEAdZ5AgQCBIIBbQSCAWkBZwB2AM8RVu7VLnyv84db2Wkum+kacWdKsBfsrAHSW3fOzDsIAAABjRIzbZMAAAQDAEcwRQIhAO0rzenZ/M/esxDf/XgeYVPu3J1sR24mW/jzs7Pg
66/5AiAfgZau1dFDa5JBk0qZThhBiZD3GJOUtE1pAyDiHaQe2AB1AH1ZHhLheCp7HGFnfF79+NCHXBSgTpWeuQMv2Q6MLnm4AAABjRIzbUYAAAQDAEYwRAIgec6rEiWhJMfcBxVEopQIjumqLsm3FVa76olbXUeTFvACIAbKs2vnxl5Xk+cgGqRuI7+h51F
qnDMnCcifAXkDah5WAHYA5tIxY0B3jMEQQQbXcbnOwdJA9paEhvu6hzId/R43jlAAAAGNEjNtcAAABAMARzBFAiEAmX/vafuqIJfw2UTfAI4GzyOu6jacwWA5QueY5piAz1kCIGU1cGmpsZIhipE8KD/SHu7VH4sYXyL49kR37l15vqAdMA0GCSqGSIb3DQ
EBCwUAA4IBAQBqspCf1rCLo5RGpNWX33G6gJ+sQiBXdNs5qnB4vCozgj9qGpHxOIvWdOndU02JOR8Ml/NwrfDk4+/RJiAj8owR7vaRwCWsY2wN1k9acTL5CpTb/aOIpFcB7QM1weh+jBka7yWWS0zGBhd4OqaR+k/jIU2nCeY9oRxmxytzk5GL+Jo6Czdm2
x3PantjeyAmh1LE9EaNbs4qfse76a4XObKjRdITx8Y9O5/PfASyC7ZmxSx0NaaP4pca+DKtoNa35zp6IN7hhShmZykI/PV2gifb2/i0Q1Mo+f9iPovIbG0gD1Zwi7SSYBUs2WAVOh3rE1tjsiAQ/ImjmWQAiq92eGC"
,"EncodeCerBytesMd5":"8fce23d4e1f4ac154e3f74eb613d7257","status":"1000"}
2024-12-19 14:13:32.285281+0800 QAPM_Example[34491:3271310] [QAPM_LogInfo][QAPMNetworkHandler.m:331][Config] EncodeCerString success
2024-12-19 14:13:32.291429+0800 QAPM_Example[34491:3271310] [QAPM_LogEvent][QAPMPathUtil.m:55][Config]
getAuthorizeUrl::https://app.rumt-zh.com/entrance/5105/authorize/?app_key=d8165f39&p_id=5105&deviceid=qwerdf&plugin_ver=1&sdk_ver=5.4.4&platform=ios&version=2.4
.42-qapm-sdk-debug&device=iPhone10,3&os=Version 16.4 (Build 20E247)&uin=qwerdf
2024-12-19 14:13:32.331570+0800 QAPM_Example[34491:3271310] [QAPM_LogDebug][QAPMDataUploadCenter.m:657][UploadReport] The local certificate is valid
2024-12-19 14:13:32.331711+0800 QAPM_Example[34491:3271310] [QAPM_LogDebug][QAPMDataUploadCenter.m:666][UploadReport] The server certificate is the same as the local certificate
2024-12-19 14:13:32.361706+0800 QAPM_Example[34491:3271306] [QAPM_LogDebug][QAPMDataUploadCenter.m:598][Config] token success, data is
{"token":"d8165f39&1734588812&4253039e5173644fdf79bbb56159a1f07dd11c18e12839b954dd8482b2ebc2d4"}
2024-12-19 14:13:32.361877+0800 QAPM_Example[34491:3271306] [QAPM_LogInfo][QAPMNetworkHandler.m:331][Config] token success
2024-12-19 14:13:32.369268+0800 QAPM_Example[34491:3271306] [QAPM_LogEvent][QAPMPathUtil.m:111][Config] getConfigUrl::https://app.rumt-zh.com/appconfig/v7/config/5105/
```



> **⚠ 注意：**
> 启动、崩溃/anr、webview 的 jserror 这些功能是全量采集，卡顿、http 网络、webview 的慢请求、掉帧是抽样采集，测试流程内可配置白名单验证数据的上报，具体配置白名单流程请参照问题9进行添加。

启动监控后，为什么没有数据显示?

1. 请确保在工程的 main 函数进行打点操作，并已加入 `[QAPMLaunchProfile didEnterMain]` 函数。
2. 启动列表无数据显示。
   - 冷启动和首次启动列表数据默认后台配置下发启动阈值是4000ms，验证数据时可以在首屏显示阶段主线程 Sleep 4s左右；
   - 热启动列表数据默认后台配置下发启动阈值是2000ms，且是退后台超3min返回前台，验证数据时可以在后台返回前台时 Sleep 2s左右。

卡慢问题列表为什么没有数据显示?

接入成功后可以多次操作 APP 界面，一般主线程卡顿了200ms就会上报，您可在 卡慢 页面的**卡慢问题列表**中查看数据。

流畅度列表为什么无数据显示?

卡慢 中的流畅度列表展示的是掉帧率，需在对应的 viewController 进行打点操作，且会在下次启动时上报数据，具体操作请参见 集成和初始化 。

为什么 webview 无数据展示?

webview 借助的是 framework 里面的 js_sdk.js 进行数据采集，需要在 webview 页面的 didFinishNavigation 方法下进行打点。

为什么 http/webview 网络数据未展示?

触发正常的 http 网络请求或者 webview 页面即可，然后在 网络 页面查看 HTTP 数据，在 webview 页面查看 webview 数据。

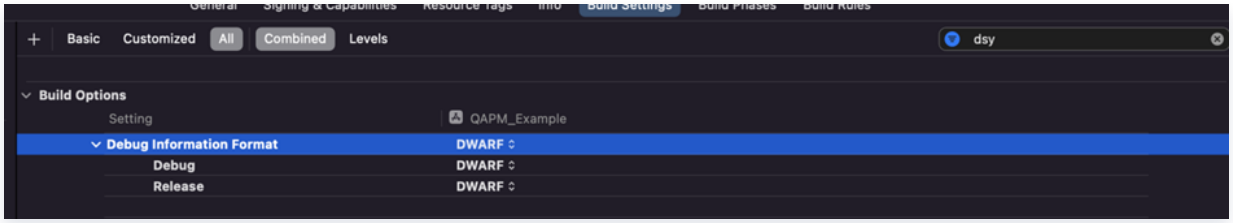
> **说明：**
> 为了能够看到 webview 和网络监控等慢请求数据，在测试手机上可以将**设置项 > 开发者**中的 **Network Link Conditioner** 打开，选择弱网进行测试。

为什么崩溃列表存在部分类型无数据?

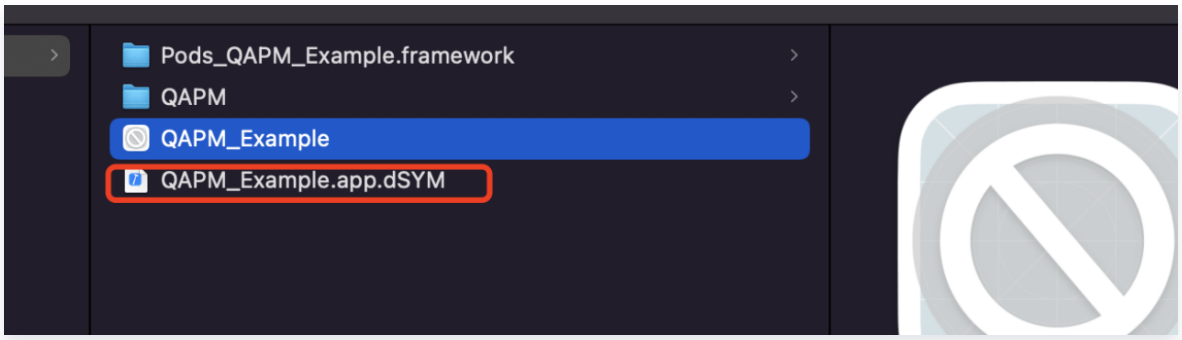
1. foom 下的爆内存 crash 在 appstore 环境下有0.02的抽样，需要触发 oom 后，在下次启动才上报数据。
2. deadlock 下的 crash 问题是在发生死锁5s时长后，且不会恢复到原有状态，才会在下次启动上报数据。
3. normalCrash 问题上报在崩溃线程的栈顶发现不是自己的业务堆栈信息，请确保工程里面没有对相应的代码编译。
   - 检查业务代码是否有在主线程做 crash 堆栈拦截处理。
   - 检查业务工程是否有添加第三方监控 crash 性能的 SDK，无论是否初始化。

堆栈没有翻译，如何处理?

1. 符号表生成方式以及压缩方法、debug 模式下如下图所示生成、release 模式回默认打开此项。



2. 工程创建完成后，会在工程默认的 Products 下生成符号表文件，可以使用 Show in Finde 的方式进入，如下图所示的 `QAPM_Example.app.dSYM` 文件即为主工程的符号表文件。



3. 符号表压缩。

   一般的业务工程会用到第三方库文件，当用到的是动态库（如网络 AFNetWorking）时会在 release 环境下生成第三方符号表文件，为了确保翻译的准确性，请将第三方符号表和业务符号表放置同一个文件夹进行进行压缩上传。

   > **说明：**
   > 在压缩多符号表时，需要全选符号表然后进行压缩，并用英文命名 zip 文件，不能直接压缩文件夹。

4. 符号表上传。

   符号表准确性校验，可以通过 `dwarfdump -uuid` 指令读取出符号表 uuid，然后和卡顿或者crash问题列表详情页面的构建 ID 进行对比，两者一致如下图所示即可上传 zip 压缩包。

数据上报缺失时如何处理？

由于部分功能默认情况下存在功能抽样，所以为了保证所有数据能正常上报，请在 终端性能监控 > 应用管理 > 白名单管理 中添加白名单操作，确保所有功能命中抽样。
例如这里添加了用户 “qwerdf” 的白名单，那么在初始化代码的地方应填写 `[QAPMConfig getInstance].userId = @"qwerdf"` 。


新增白名单账号
账号类型
用户
用户ID
qwerdf
备注(选填)
请输入用户白名单的备注
确定
取消