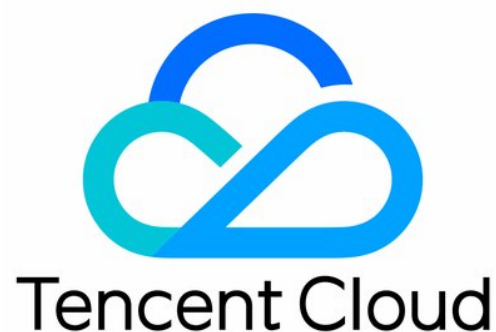


# **Tencent Cloud Observability Platform**

## **Mobile App Performance Monitoring Product Documentation**



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Mobile App Performance Monitoring

Overview

Access Guide

Android Use Cases

Integration and Initialization

Feature Configuration

Network Monitoring

WebView, JsError, and Web Network Monitoring

Crash and ANR Monitoring

Lag and Frame Rate Monitoring

Startup Monitoring

Operation Guide

Crash

ANR

Network

Webview

Application Management

# Mobile App Performance Monitoring Overview

Last updated : 2024-05-14 12:36:06

## Overview

Mobile App Performance Monitoring is a comprehensive tool for positioning, detecting app performance, and user experience, automatically analyzing suspicious performance defects in multiple dimensions. It helps you accurately measure the app performance and discover various issues with low cost and high efficiency.

## App Monitoring Feature Description

Feature Name	Description
Crash	By aggregating key characteristics of individual crash cases, it facilitates the locating and analysis of the root cause.
Startup	Supports startup metric analysis such as startup duration and slow startup percentage, allowing you to locate and analyze the root cause of slow startups through the slow startup issue list.
Latency	Metrics like smoothness help you analyze the performance of app pages.
ANR	Multi-dimensional restoration of the real experience of online users. It collects ANR issues encountered during the real use of the app and the thread stack information upon the occurrence of the issue to extract key features for clustering.
Network	Supports network problem analysis based on request duration, slow request proportion, network error rate, and other metrics.
WebView	Supports WebView metric analysis based on page load time, slow loading proportion, and JS error rate, and dives into WebView and JS error issues through the issue list.

## Data Storage Description

Individual case data (slow startup and requests): stored for 60 days

15-minute metric data (metric analysis interface): stored for 30 days

1-hour metric data (metric analysis interface): stored for 30 days

# Access Guide

## Android Use Cases

### Integration and Initialization

Last updated : 2024-05-14 12:36:06

## Overview

This document guides you through integrating and initializing with the Android SDK.

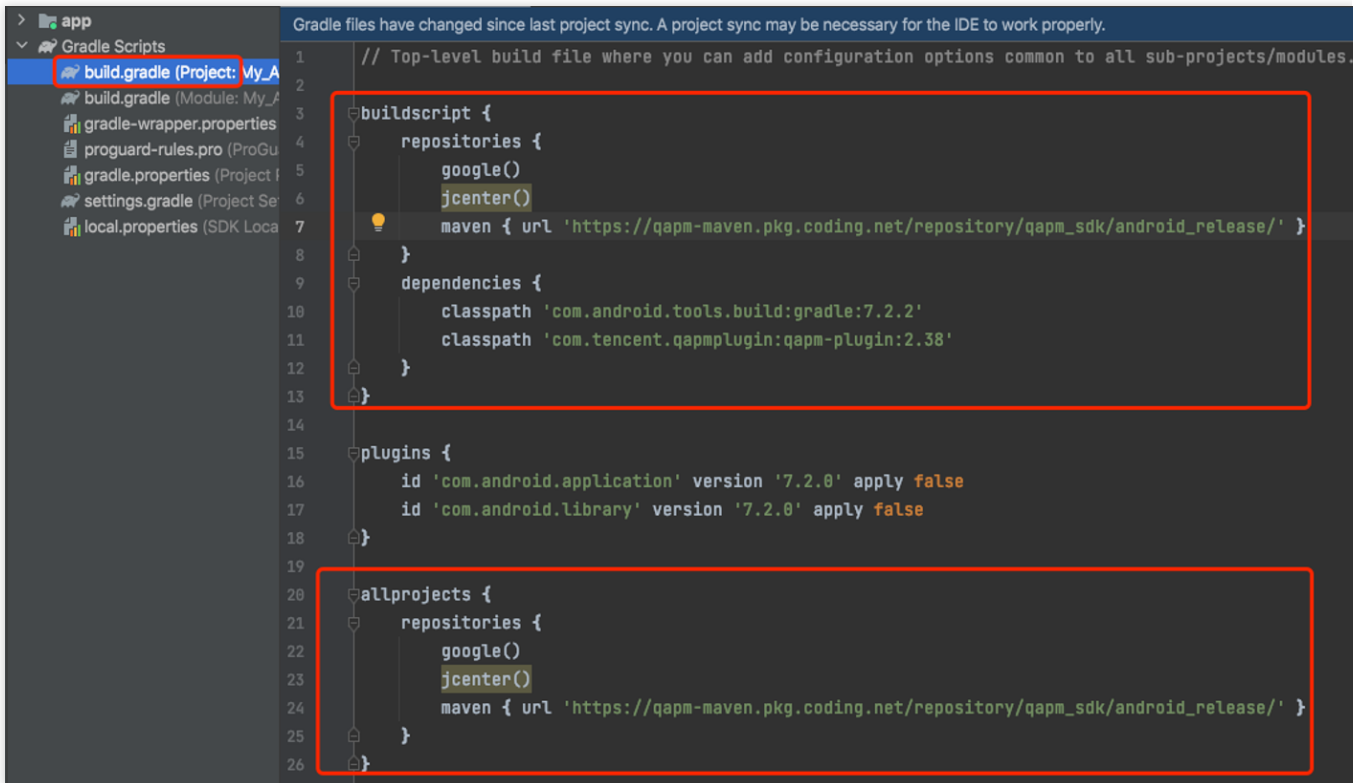
## Directions

### Step 1: Configure Gradle integration.

#### 1. Add Maven dependency in the project-level build.gradle.

- i. Add buildscript and allprojects (For Gradle 7.0 and later, adding allprojects is not necessary).

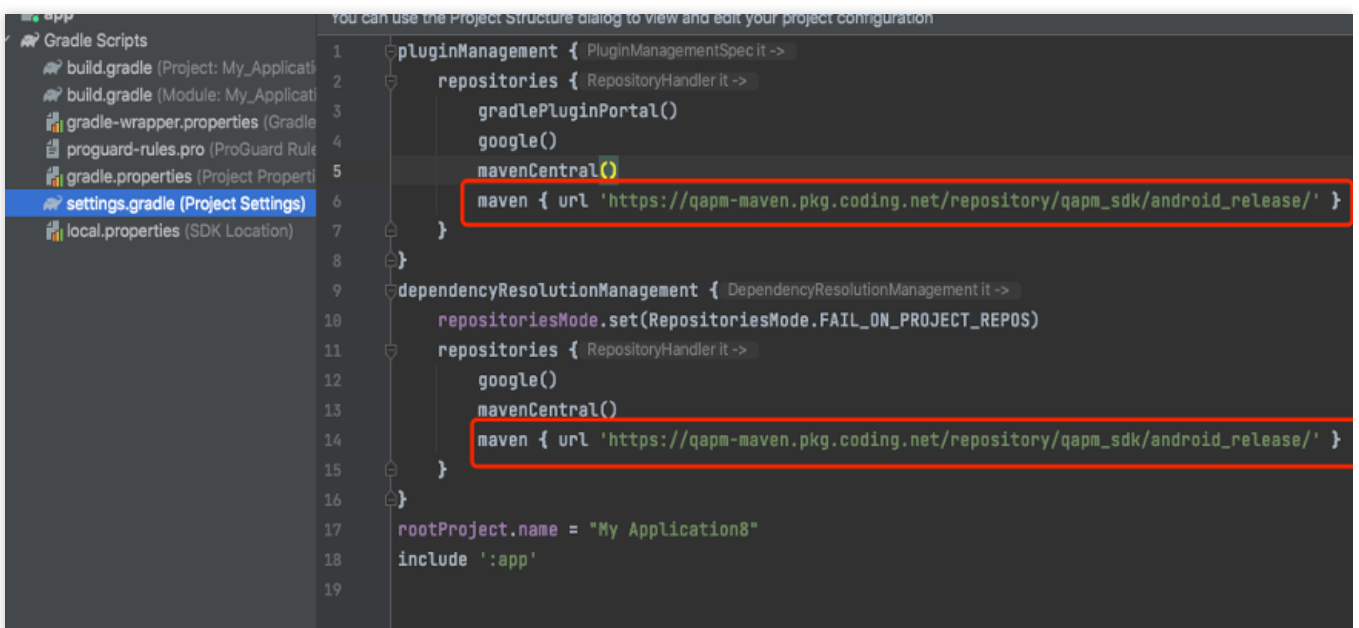
**Configuration for Gradle 7.0 and earlier is as follows:**



```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2
3 buildscript {
4     repositories {
5         google()
6         jcenter()
7         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
8     }
9     dependencies {
10        classpath 'com.android.tools.build:gradle:7.2.2'
11        classpath 'com.tencent.qapmplugin:qapm-plugin:2.38'
12    }
13 }
14
15 plugins {
16     id 'com.android.application' version '7.2.0' apply false
17     id 'com.android.library' version '7.2.0' apply false
18 }
19
20 allprojects {
21     repositories {
22         google()
23         jcenter()
24         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
25     }
26 }
```

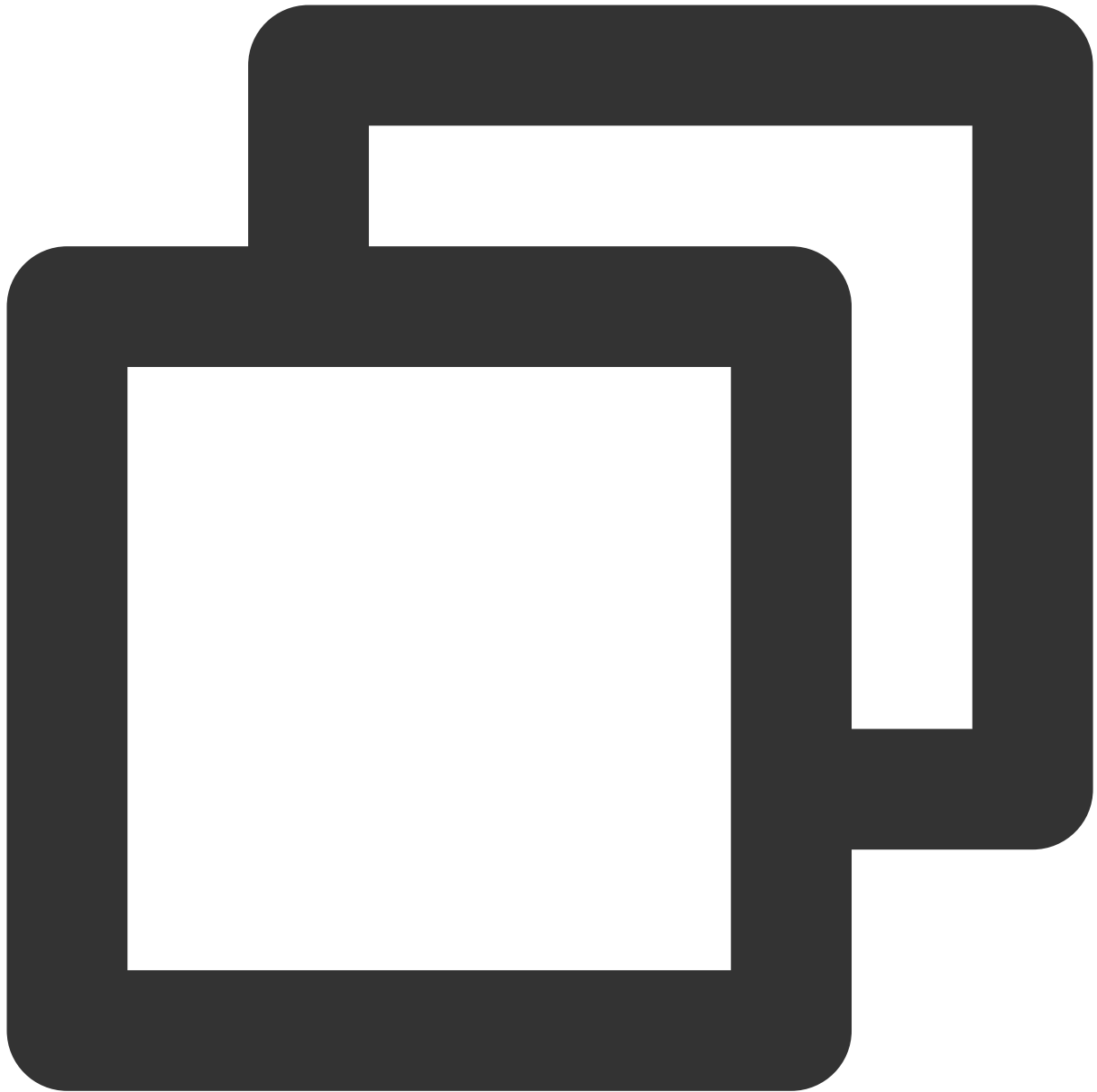
### Configuration for Gradle 7.0 and later is as follows:

Gradle 7.0 and later do not support allprojects. Maven dependency of allprojects must be configured in setting.gradle, as shown below.



```
1 pluginManagement { PluginManagementSpec it ->
2     repositories { RepositoryHandler it ->
3         gradlePluginPortal()
4         google()
5         mavenCentral()
6         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
7     }
8 }
9 dependencyResolutionManagement { DependencyResolutionManagement it ->
10     repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
11     repositories { RepositoryHandler it ->
12         google()
13         mavenCentral()
14         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_release/' }
15     }
16 }
17 rootProject.name = "My Application8"
18 include ':app'
19 }
```

Refer to the code:



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

ii. In buildscript, `com.android.tools.build:gradle:*.*.*` should be filled in with your Gradle plugin version, as shown below.

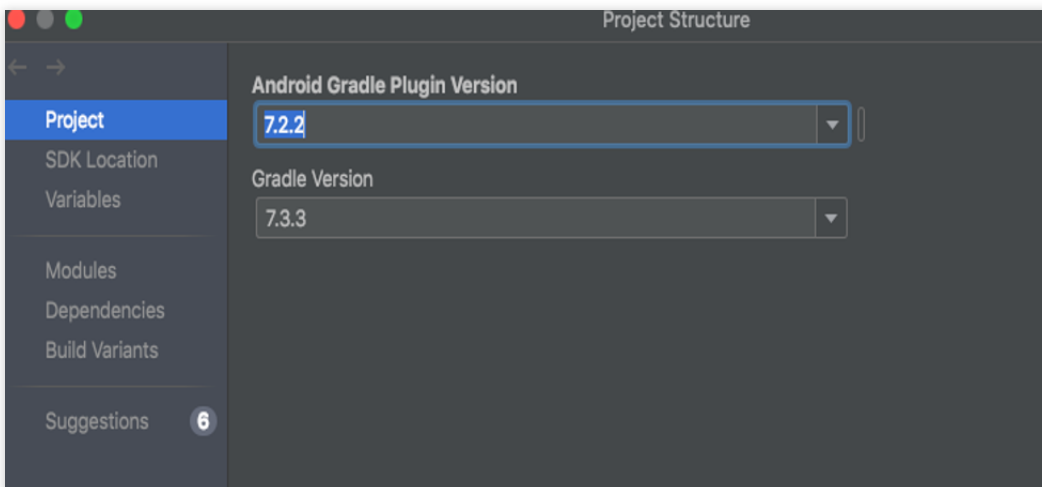


```
buildscript {  
    repositories {  
        google()  
        jcenter()  
        maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_re...' }  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:7.2.2'  
        classpath 'com.tencent.qapmplugin:qapm-plugin:2.38'  
    }  
}
```

gradle plugin version

**Note:**

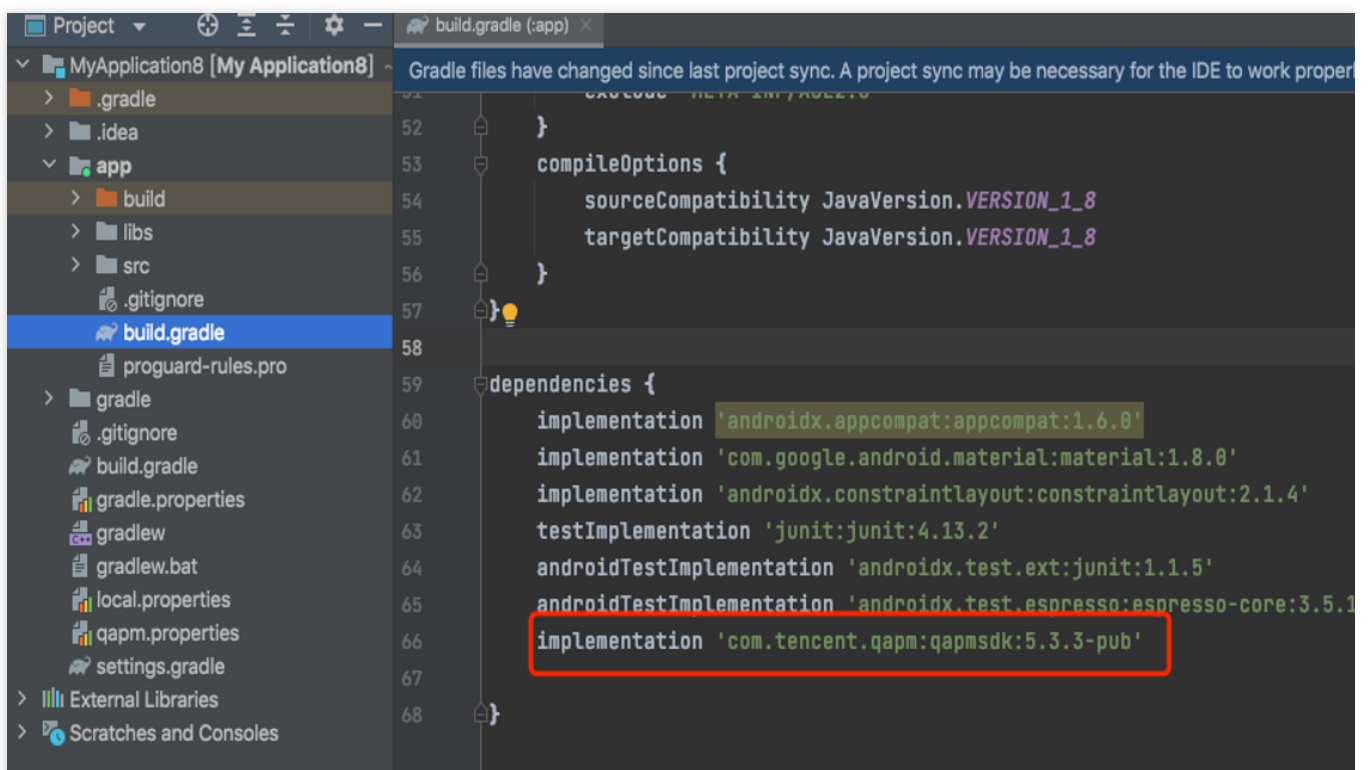
1. The Gradle version and Gradle plugin version can be viewed in the menu file > project structure, as shown below.



2. The corresponding relationship between Gradle and Gradle plugin is as shown below.

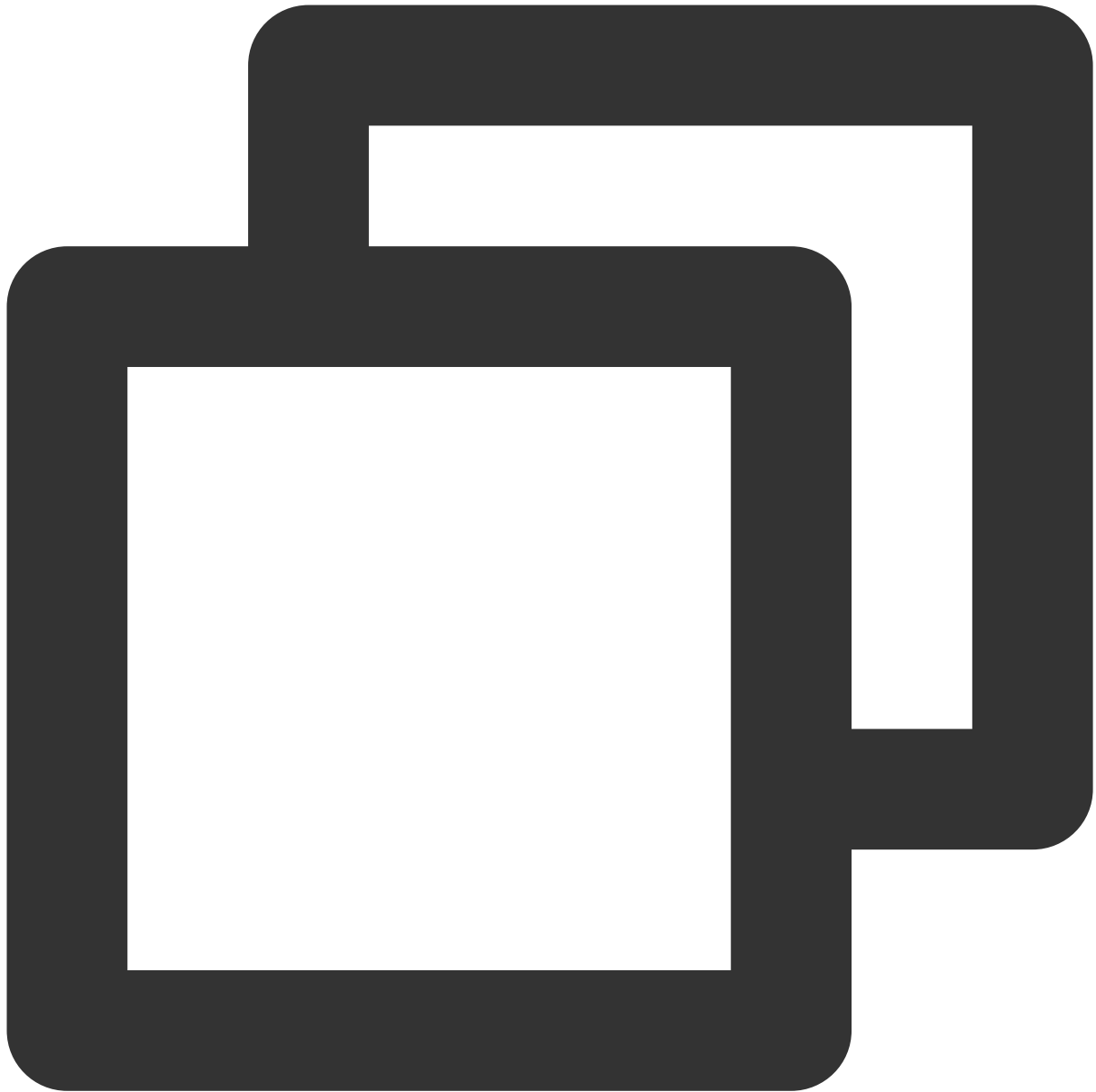
plugin version	minimum required Gradle version
8.1	8
8	8
7.4	7.5
7.3	7.4
7.2	7.3.3
7.1	7.2
7	7
4.20+	6.7.1

2. Introduce the module in the app's build.gradle (For studio 3.0 and earlier, use the compile reference header).



```
52 }
53
54 compileOptions {
55     sourceCompatibility JavaVersion.VERSION_1_8
56     targetCompatibility JavaVersion.VERSION_1_8
57 }
58
59 dependencies {
60     implementation 'androidx.appcompat:appcompat:1.6.0'
61     implementation 'com.google.android.material:material:1.8.0'
62     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
63     testImplementation 'junit:junit:4.13.2'
64     androidTestImplementation 'androidx.test.ext:junit:1.1.5'
65     androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
66     implementation 'com.tencent.qapm:qapmsdk:5.3.3-pub'
67 }
68 }
```

Refer to the code:



```
implementation 'com.tencent.qapm:qapmsdk:5.3.9-pub'
```

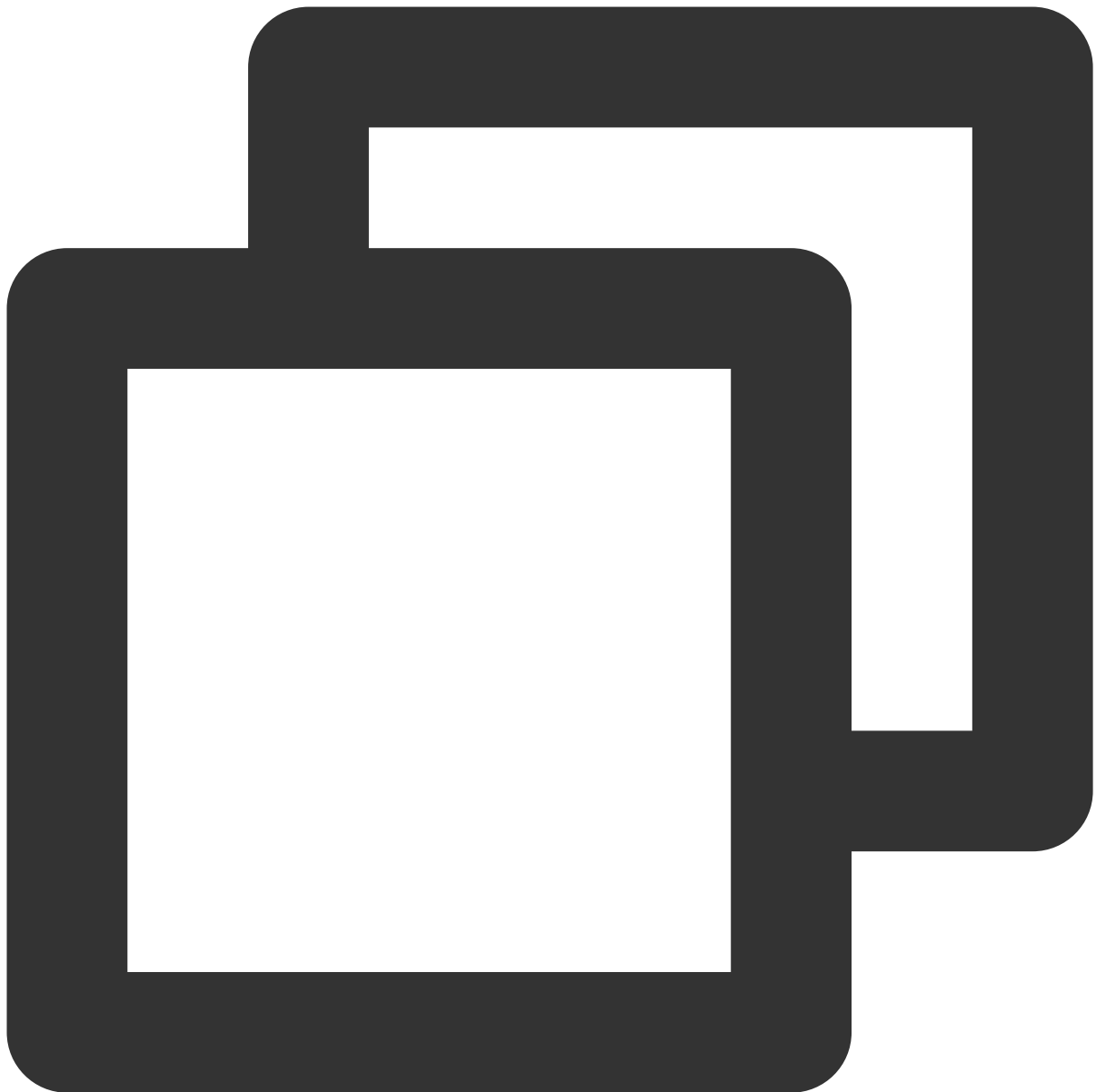
### 3. Introduce Kotlin dependencies.

- i. Add the following code under the project-level build.gradle:



```
1 // Top-level build file where you can add configuration options common to all sub-projects.
2
3 buildscript {
4     ext.kotlin_version = '1.3.41'
5     repositories {
6         google()
7         jcenter()
8         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android_release' }
9     }
10    dependencies {
11        classpath 'com.android.tools.build:gradle:7.2.2'
12        classpath 'com.tencent.qapmplugin:qapm-plugin:2.38'
13        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
14    }
15 }
16
```

Refer to the code:



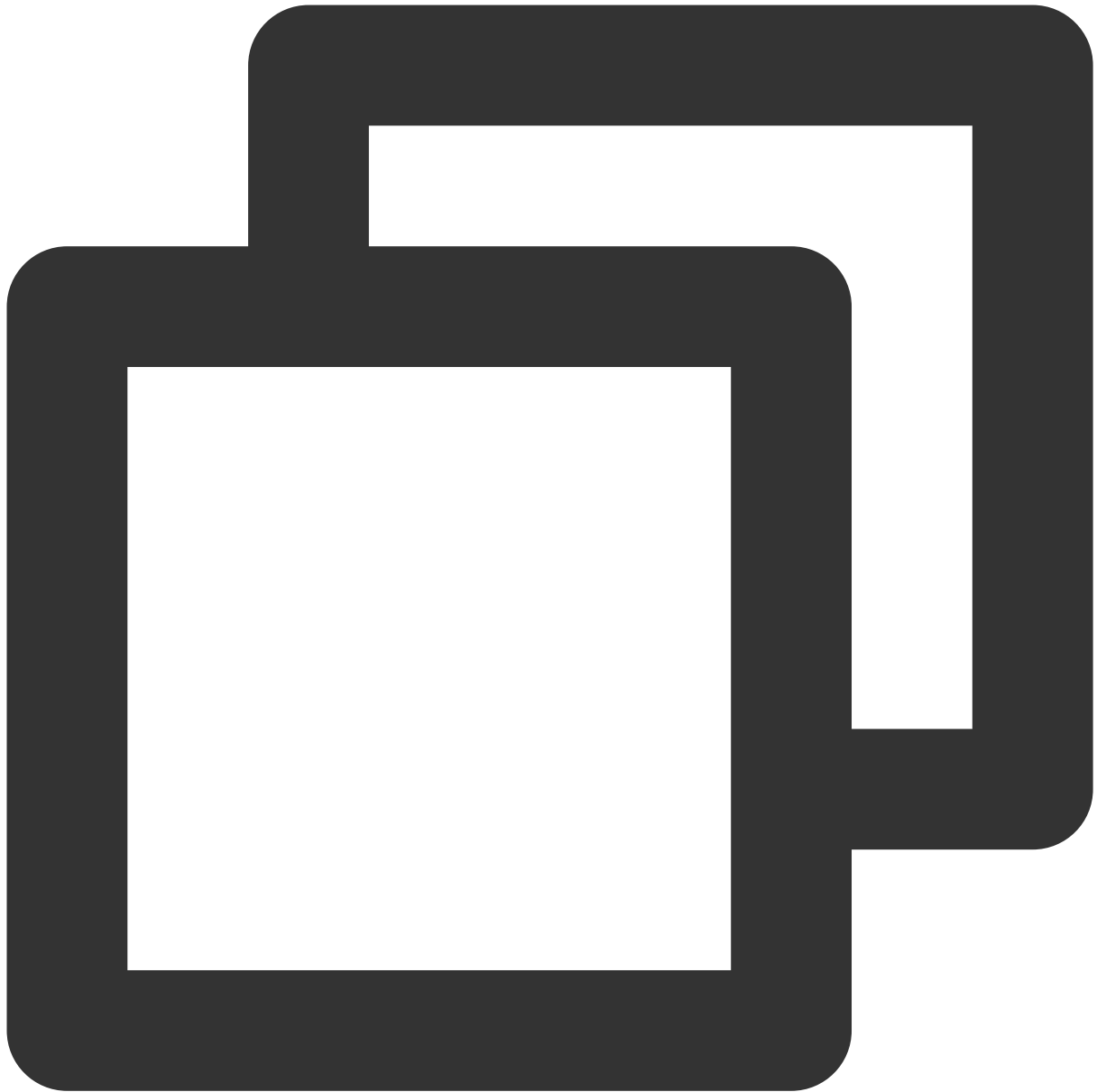
```
ext.kotlin_version = '1.3.41'  
classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
```

ii. Add the following code to the app's build.gradle:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

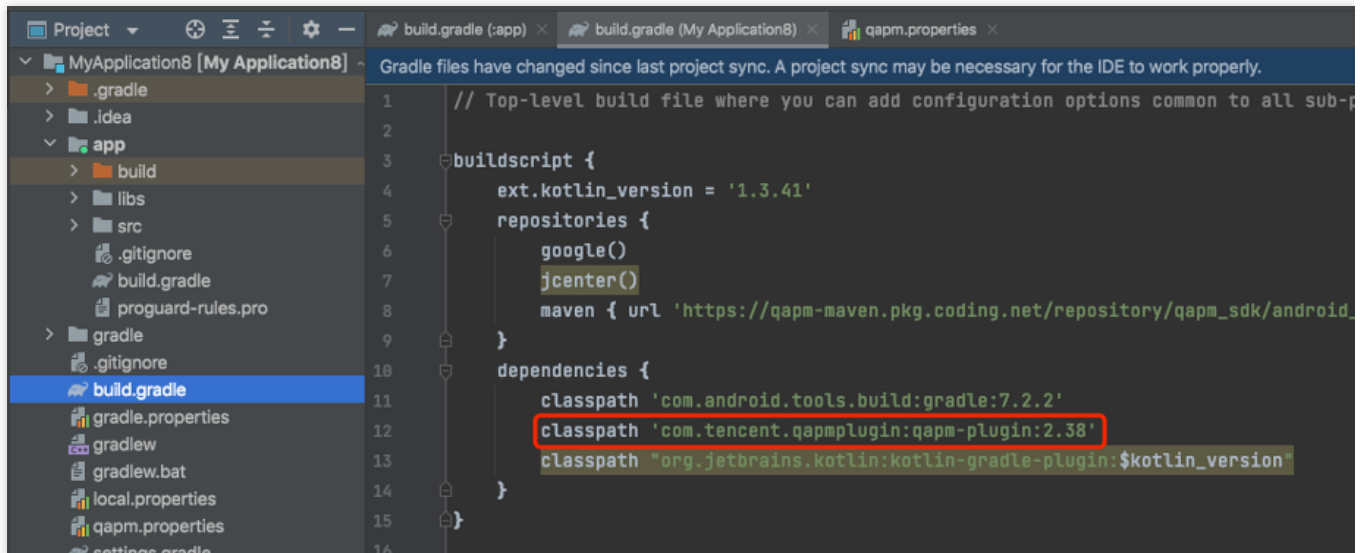
android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.sdkapp"
        minSdkVersion 16
        targetSdkVersion 26
    }
}
```

Refer to the code:



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

**4. Add the following configuration in the project-level build.gradle file.**



```
1 // Top-level build file where you can add configuration options common to all sub-p
2
3 buildscript {
4     ext.kotlin_version = '1.3.41'
5     repositories {
6         google()
7         jcenter()
8         maven { url 'https://qapm-maven.pkg.coding.net/repository/qapm_sdk/android.'
9     }
10 }
11 dependencies {
12     classpath 'com.android.tools.build:gradle:7.2.2'
13     classpath 'com.tencent.qapmplugin:qapm-plugin:2.38'
14     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
15 }
16
```

Refer to the code:

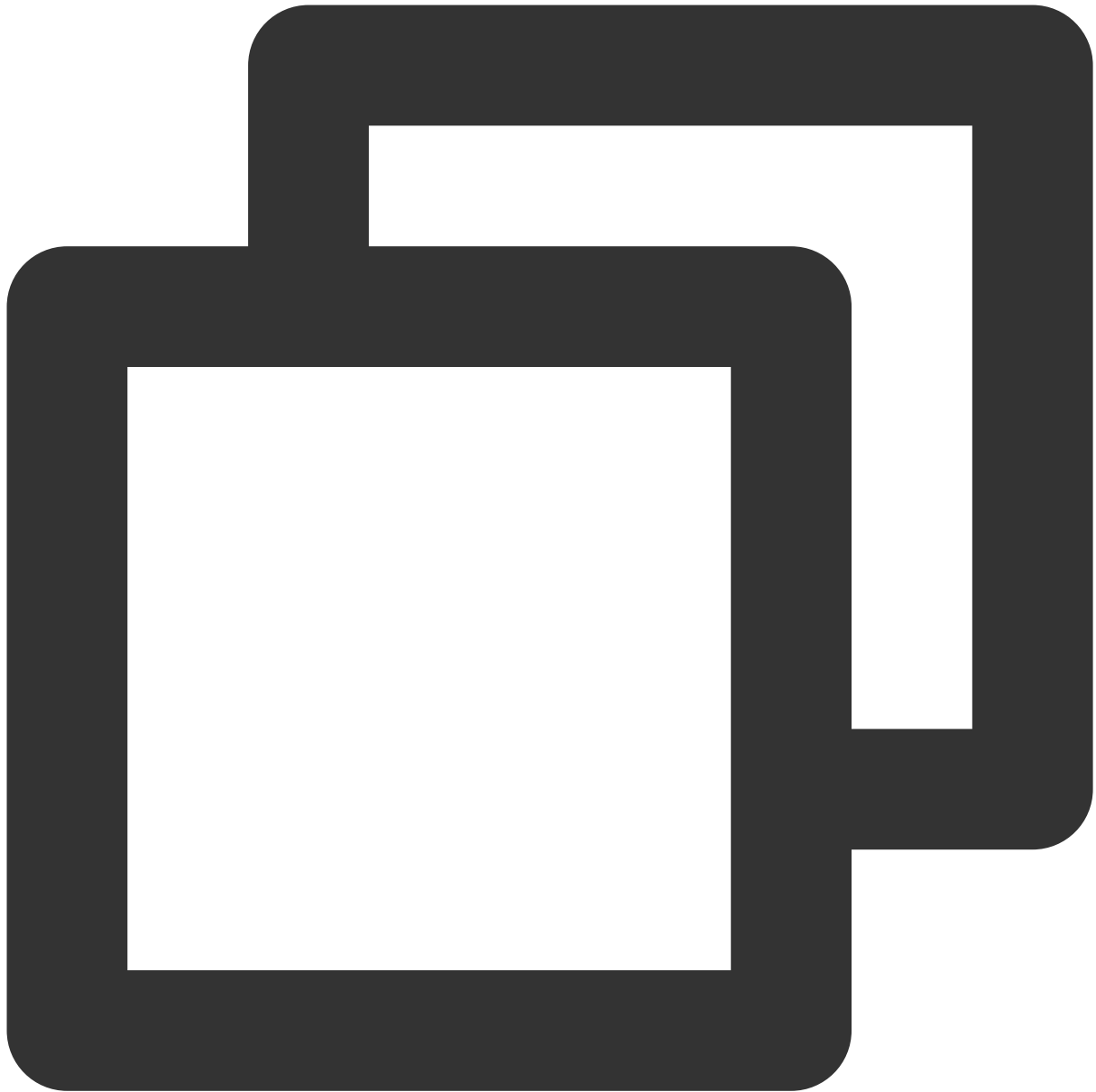




```
classpath 'com.tencent.qapmplugin:qapm-plugin:2.39'
```

**5. Add the following configuration in the app's build.gradle file.**

Refer to the code:



```
apply plugin: 'qapm-plugin'  
QAPMPluginConfig{  
    // Optional, and empty by default. Enter attachBaseContext in the class where the  
}
```

**Note:**

If items 4-5 are not configured, it will affect the data reporting of "App launch".

**Step 2: Configure parameters.**

1. Add the following permissions in AndroidManifest.xml.



```
<!--Required for information reporting-->  
<uses-permission android:name="android.permission.INTERNET" />  
<!--Required for information collection-->  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

2. To avoid confusion with the SDK, add the following configuration in the app's proguard-rules.pro file:



```
-keep class com.tencent.qapmsdk.**{*;}  
# If network monitoring is needed, ensure okhttp3 is not obfuscated.  
-keep class okhttp3.**{*};
```



### Step 3: Initialize the SDK.

1. Log in to [TCOP](#) console, navigate to the **Mobile App Performance Monitoring** page, select [Application Management > Application Settings](#), and then obtain the Appkey (reporting ID).

### Application Management

Business System   **Application Settings**   Allowlist Management

Business System: rum-lruQGZfQxnBZ0K.现网测试   [Application Access](#)

Application Name	Report ID	Application ID
现网测试- Android	7f7073be-49 	500347
云监控- android demo	cdf07086-10344 	500348

Total items: 2

2. Copy the code below and modify some of the fields. The following items are mandatory interface settings; for additional interface configurations, refer to the initialization interface analysis (Initializing QAPM in Application is recommended).



```
// Set the mobile phone model and device ID.  
// Device identifier required, in the form of any string. deviceId (mandatory).  
// The deviceId can be used to enable an allowlist, avoiding data sampling (Except  
QAPM.setProperty(QAPM.PropertyKeyDeviceId, "Device identifier");  
// The mobile phone model is required (mandatory).  
QAPM.setProperty(QAPM.PropertyKeyModel, "Enter mobile phone model");  
  
// Set Application (mandatory).  
QAPM.setProperty(QAPM.PropertyKeyAppInstance, getApplication());  
// Set AppKey (mandatory, used to distinguish the reporting products. The value is  
QAPM.setProperty(QAPM.PropertyKeyAppId, "YourAppKey");
```

```
// Set the product version, which is used for background retrieval field (mandatory
QAPM.setProperty(QAPM.PropertyKeyAppVersion, "YourApp Version");
// Set the UUID to pull the obfuscated stack's mapping (mandatory. If QAPM Symbol T
// This variable is generated during the build time, ignore if it shows errors.
QAPM.setProperty(QAPM.PropertyKeySymbolId, BuildConfig.QAPM_UUID);
// Set the user ID, in the form of any string, which is used for background retriev
// The userId can be used to enable an allowlist, avoiding data sampling (except fo
QAPM.setProperty(QAPM.PropertyKeyUserId, "123456");
// Set the Log level (optional). For the production environment version, set it to
QAPM.setProperty(QAPM.PropertyKeyLogLevel, QAPM.LevelInfo);
// Set the QAPM external network reporting domain (required). Chinese Mainland: htt
QAPM.setProperty(QAPM.PropertyKeyHost, "https://app.rumt-zh.com");
QAPM.setProperty(QAPM.PropertyKeyHost, "https://app.rumt-sg.com");
// Enable QAPM.
QAPM.beginScene(QAPM.SCENE_ALL, QAPM.ModeStable);
```

#### Note:

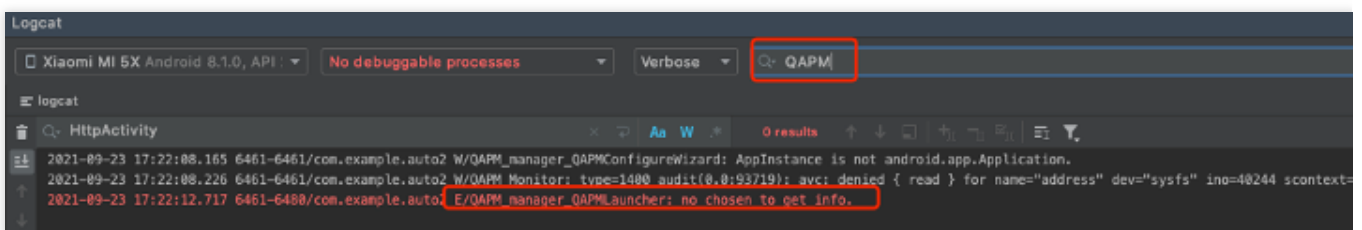
The AppKey can be obtained from the **Mobile App Performance Monitoring > Application Management** page as in Step 1.

Crash and start data are reported in full, while other data is sampled due to its large volume, at a rate of 0.1% (one in a thousand). To report all data, an allowlist can be enabled, and the app will change the sampling rate at the next start. The preset userId or deviceId can be added to the allowlist through the [Application Management](#) page to enable the allowlist.

Multiple processes need to initialize QAPM, separately.

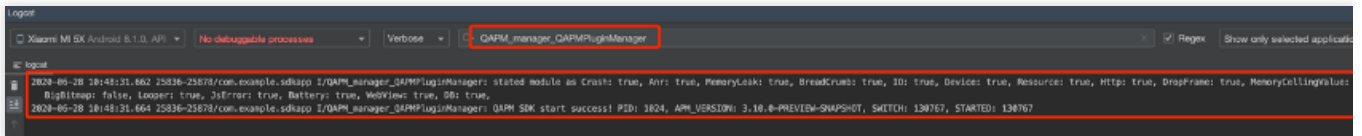
#### Step 4: Access verification.

1. If the following log is printed, it indicates that the user has not been sampled, and the sampling rate needs to be reset:



See TAG: QAPM\_manager\_QAPMLauncher

2. If the following log is printed, it indicates that the initial access succeeded. You can verify data reporting and try to enable the advanced feature:



See TAG: QAPM\_manager\_QAPMPluginManager

## Initialized Interface Analysis

API Name	Parameter	Parameter Description	Notes
public static QAPM setProperty(int key, Stringvalue) <b>Functionality:</b> Set QAPM parameters.	key	<b>Required.</b> The Key that needs to be set.	-
	QAPM.PropertyKeyLogLevel	<b>Optional.</b> Enable log level. (It is recommended to use QAPM.LevelDebug for Debug versions and QAPM.LevelWarn for release versions).	
	QAPM.PropertyNeedTranslate	<b>Optional.</b> Whether stack translation is needed, which by default is required. If the apk is not obfuscated, then pass in 'false'. Otherwise the frontend may display everything as 'unTranslated'.	
public static boolean beginScene(String sceneName, int mode)	sceneName	<b>Required.</b> Scene name.	For official versions, it is recommended to enable QAPM.ModeStable; for development versions, QAPM.M is recommended.
	mode	<b>Required.</b> The feature to be	



<p><b>Functionality:</b> Enable monitoring.</p>		enabled.	<p>By default, the ModeStable featu enabled. You can add the needer features on ModeStable by using operation, for example, to enable and memory ceiling: beginScene("Stable&amp;Ceiling", QAPM.ModeStable  QAPM.ModeCeiling). The XOR c can be used to exclude unnecess features, such as disabling the ne QAPM.ModeStable^QAPM.Modi</p>
	QAPM.ModeStable	<p><b>Optional.</b> Enable all features (Recommended for external releases. Includes interval performance, crash, ANR, WebView page load, JsError, and network).</p>	
	QAPM.ModeWebView	<p><b>Optional.</b> Enable WebView page load monitoring.</p>	
	QAPM.ModeJsError	<p><b>Optional.</b> Enable WebView JS exception monitoring.</p>	
	QAPM.ModeHTTPInWeb	<p><b>Optional.</b> Enable WebView network monitoring.</p>	
	QAPM.ModeHTTP	<p><b>Optional.</b> Enable network monitoring.</p>	
<p>public static boolean endScene(String sceneName, long mode) <b>Functionality:</b> End monitoring (Only effective for frame drops and interval performance collection).</p>	sceneName	<p><b>Required.</b> The name of the scene to be turned off (Required to correspond to the beginScene).</p>	-
	QAPM.ModeDropFrame	<p><b>Optional.</b> Turn off frame drop monitoring.</p>	
	QAPM.ModeResource	<p>Optional. Turn off interval performance monitoring.</p>	

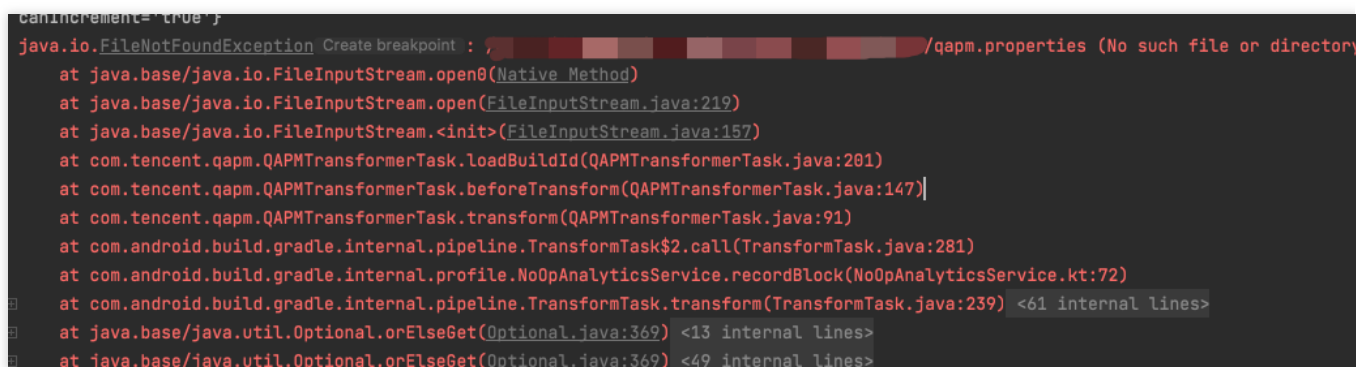
## Others

### Note:

When compiling and packaging the app through the qapm plugin, the app needs a UUID as the Build ID. If there is a qapm.properties file in the project directory, and the value of the qapm\_uuid property exists, this value will be used as the Build ID; otherwise, the plugin will randomly generate a Build ID.

qapm-plugin Version 2.39 and earlier will report an IO Error during the app compiling process:

```
java.io.FileNotFoundException, qapm.properties (No such file or directory) .
```



```
canIncrement="true"}
java.io.FileNotFoundException Create breakpoint : /qapm.properties (No such file or directory)
  at java.base/java.io.FileInputStream.open0(Native Method)
  at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
  at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
  at com.tencent.qapm.QAPMTransformerTask.loadBuildId(QAPMTransformerTask.java:201)
  at com.tencent.qapm.QAPMTransformerTask.beforeTransform(QAPMTransformerTask.java:147)
  at com.tencent.qapm.QAPMTransformerTask.transform(QAPMTransformerTask.java:91)
  at com.android.build.gradle.internal.pipeline.TransformTask$2.call(TransformTask.java:281)
  at com.android.build.gradle.internal.profile.NoOpAnalyticsService.recordBlock(NoOpAnalyticsService.kt:72)
  at com.android.build.gradle.internal.pipeline.TransformTask.transform(TransformTask.java:239) <61 internal lines>
  at java.base/java.util.Optional.orElseGet(Optional.java:369) <13 internal lines>
  at java.base/java.util.Optional.orElseGet(Optional.java:369) <49 internal lines>
```

This error only occurs during compilation and does not affect the running of the app.

# Feature Configuration

## Network Monitoring

Last updated : 2024-05-14 12:36:06

### Enabling Feature

Network monitoring requires the use of the qapm-plugin for instrumentation and is by default inserted at various entry and exit points of the network layer.

### Prerequisites

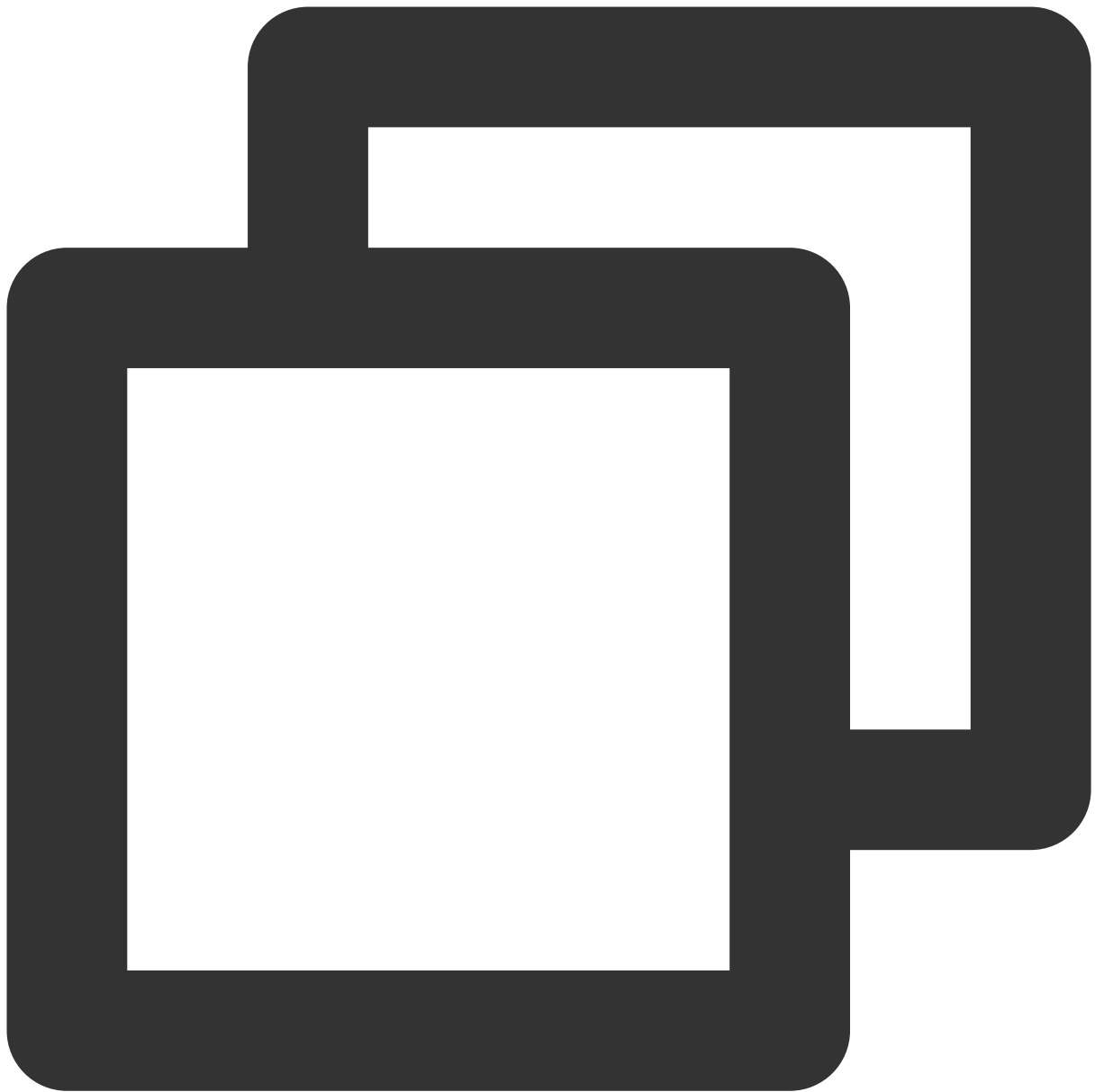
A user allowlist or device allowlist, which is used to initialize the SDK, has been added through the [Mobile App Performance Monitoring > Application Management > Allowlist Management](#) page.

The qapm-plugin has been configured in the app-level build.gradle. See [Integration and Initialization](#).

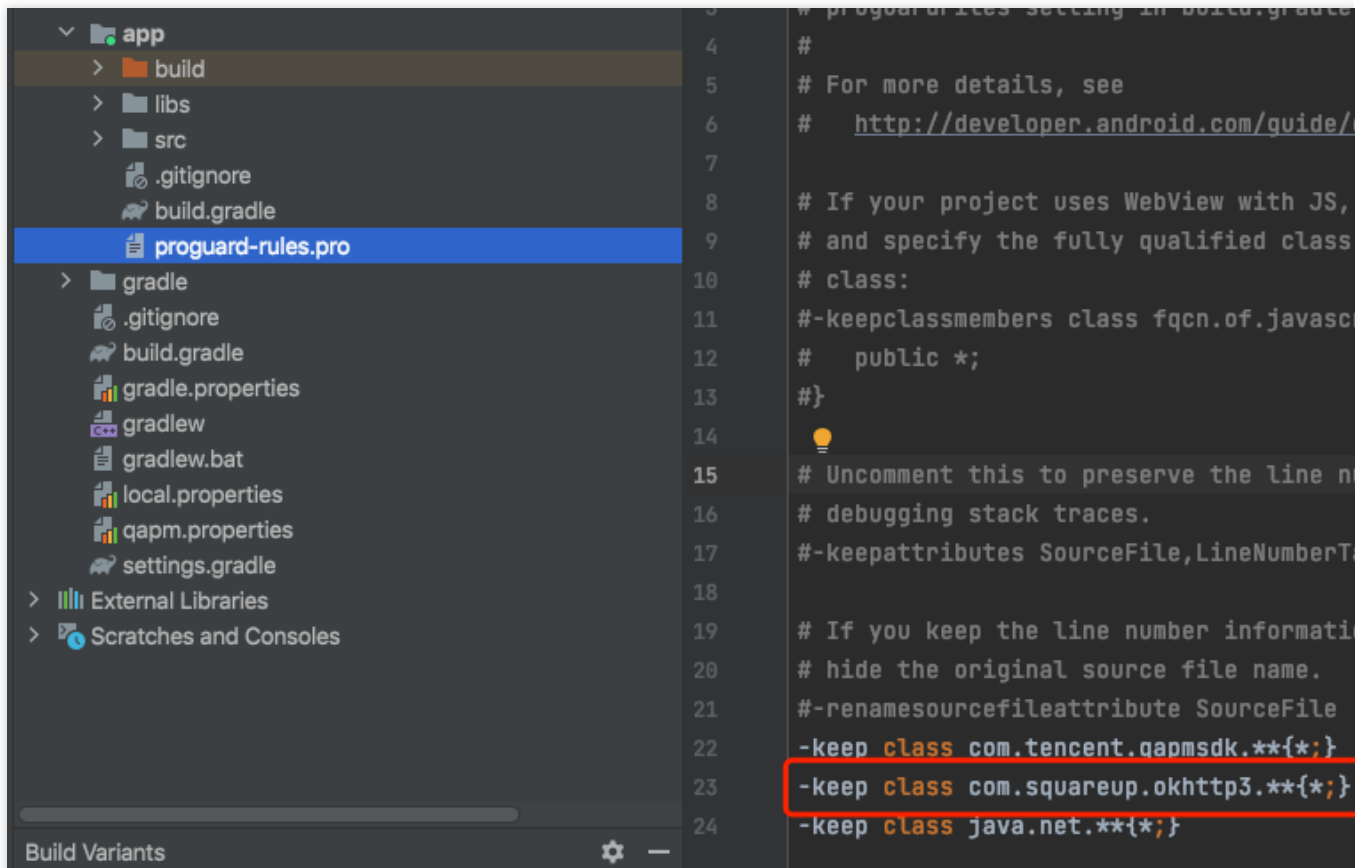
Currently, only okhttp3 monitoring is supported. The okhttp3 also requires okio version 1.14.0 or later.

### Configuration Process

Add obfuscation rules in the proguard-rules.pro file in the app directory to prevent okhttp3 code from being obfuscated.



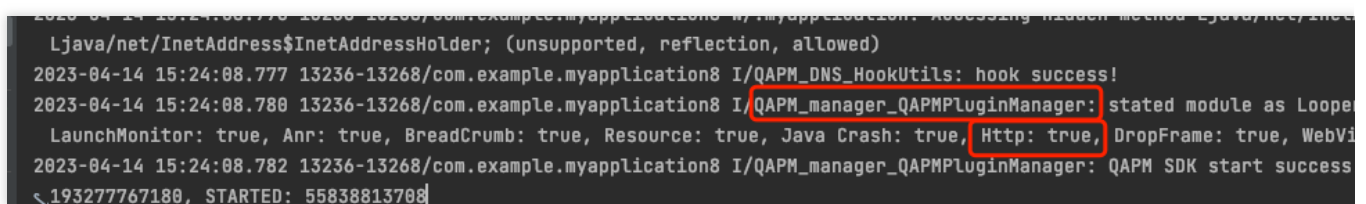
```
-keep class com.squareup.okhttp3.**{*;} 
```



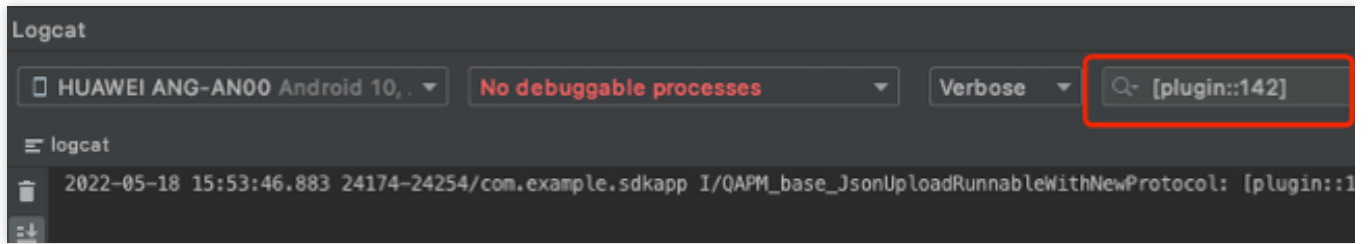
## Verifying Whether the Feature Is Working Properly

Retrieval tag: QAPM\_manager\_QAPMPluginManager

The log message that appears one minute after each network request indicates successful reporting of network data:



Retrieval tag: [plugin::142]

**Note:**

It requires the use of the qapm-plugin for instrumentation. Otherwise, it will not work.

The SDK is only responsible for capturing information related to network requests. The backend analyzes issue data, such as slow requests (with the request time being greater than xxs) and network errors (with the request response code being greater than 400).

Data can be viewed in [Mobile Performance Monitoring > Network > Slow Requests and Error Requests List](#).

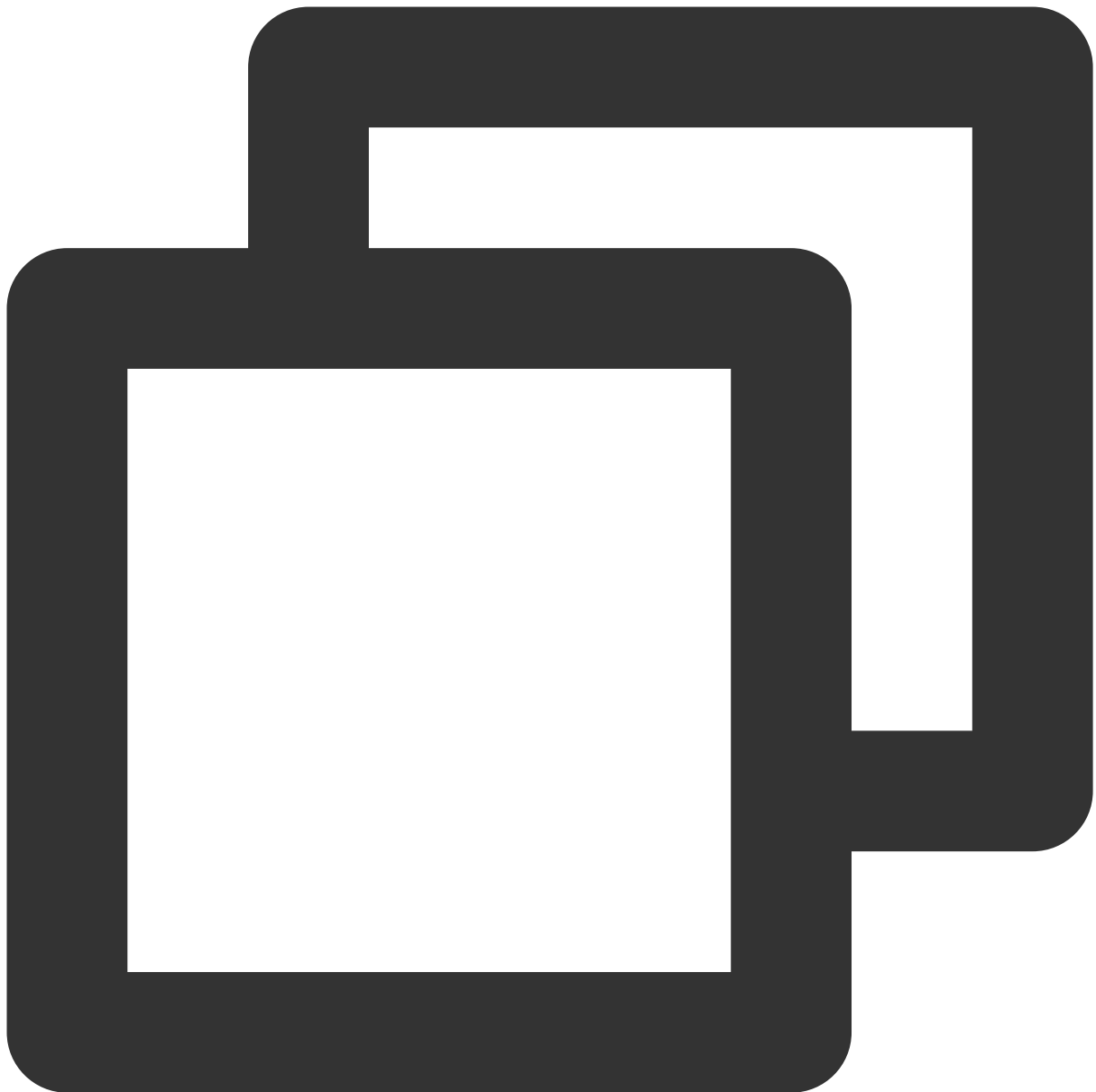
If the correct allowlist is not configured, the SDK will not enable network monitoring.

# WebView, JsError, and Web Network Monitoring

Last updated : 2024-05-14 12:36:06

## Enabling Feature

Initialization requires enabling WebView, JsError, and Web network monitoring. Below is how to enable these three features based on Stable. The code is as follows:

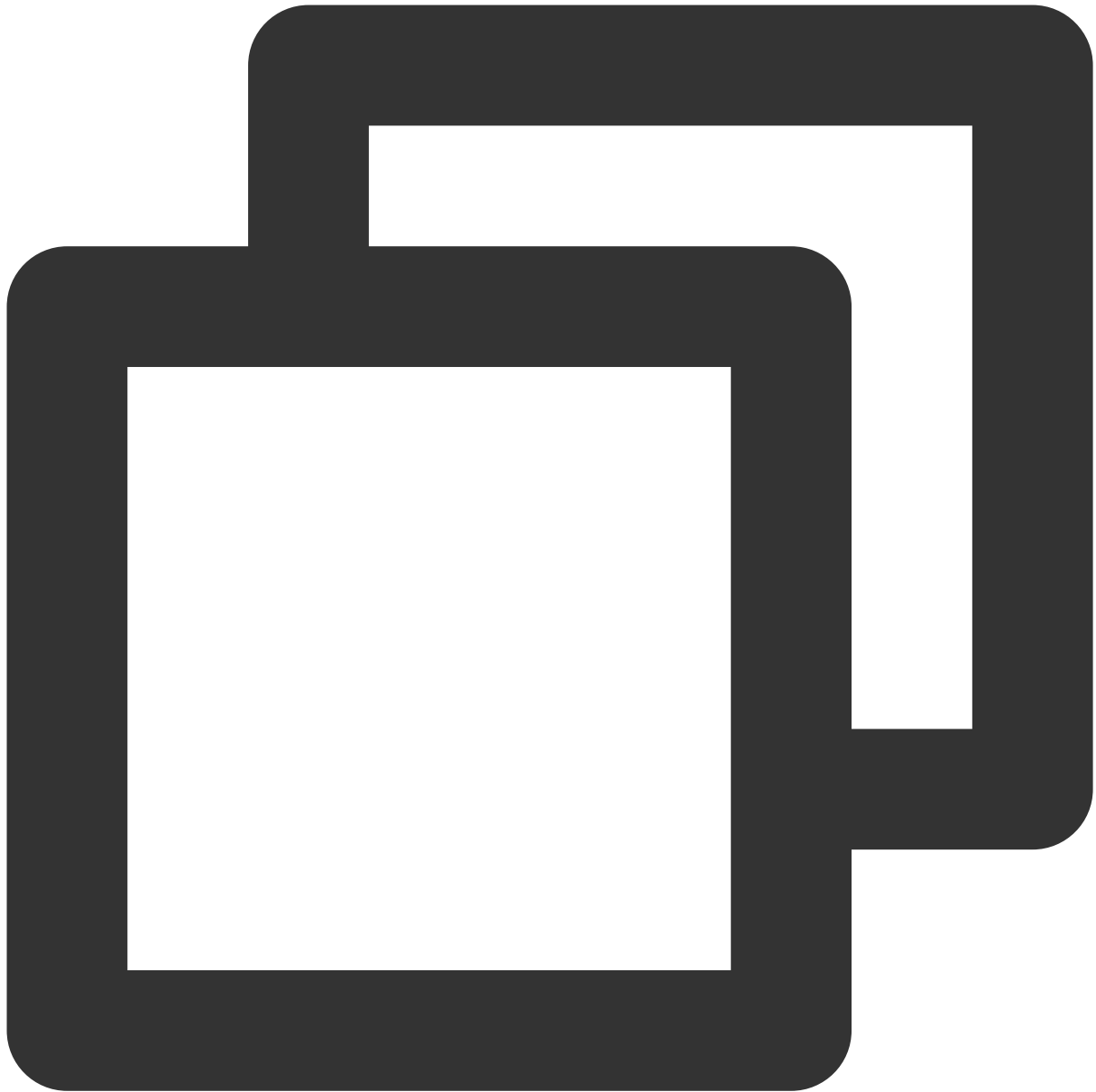


```
QAPM.beginScene(QAPM.SCENE_ALL, QAPM.ModeStable | QAPM.ModeWebView | QAPM.ModeJsErr
```

In addition, the following code needs to be configured:

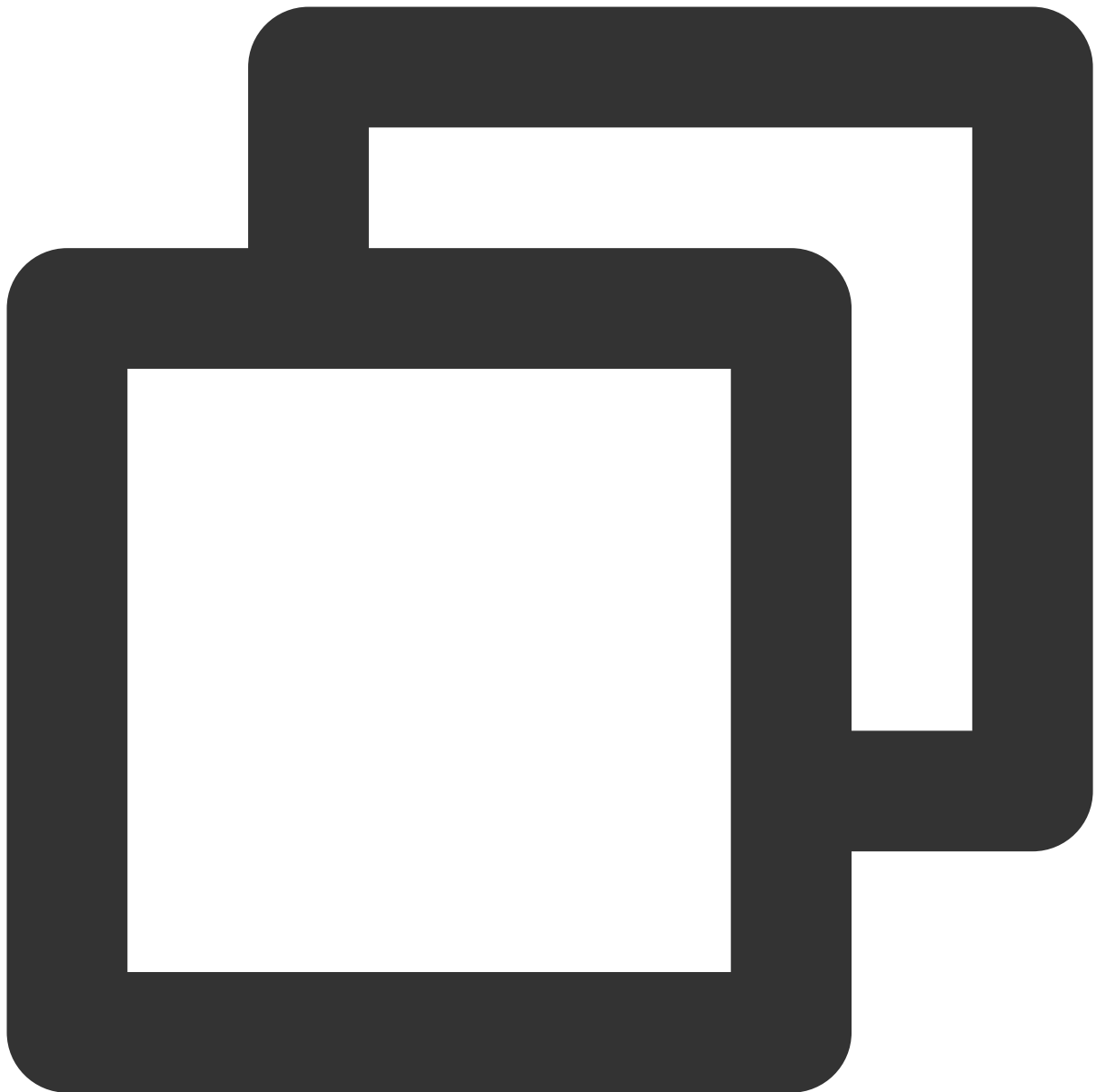
WebView Monitoring requires enabling interaction with JavaScript. Call the following code during WebView initialization to enable:





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

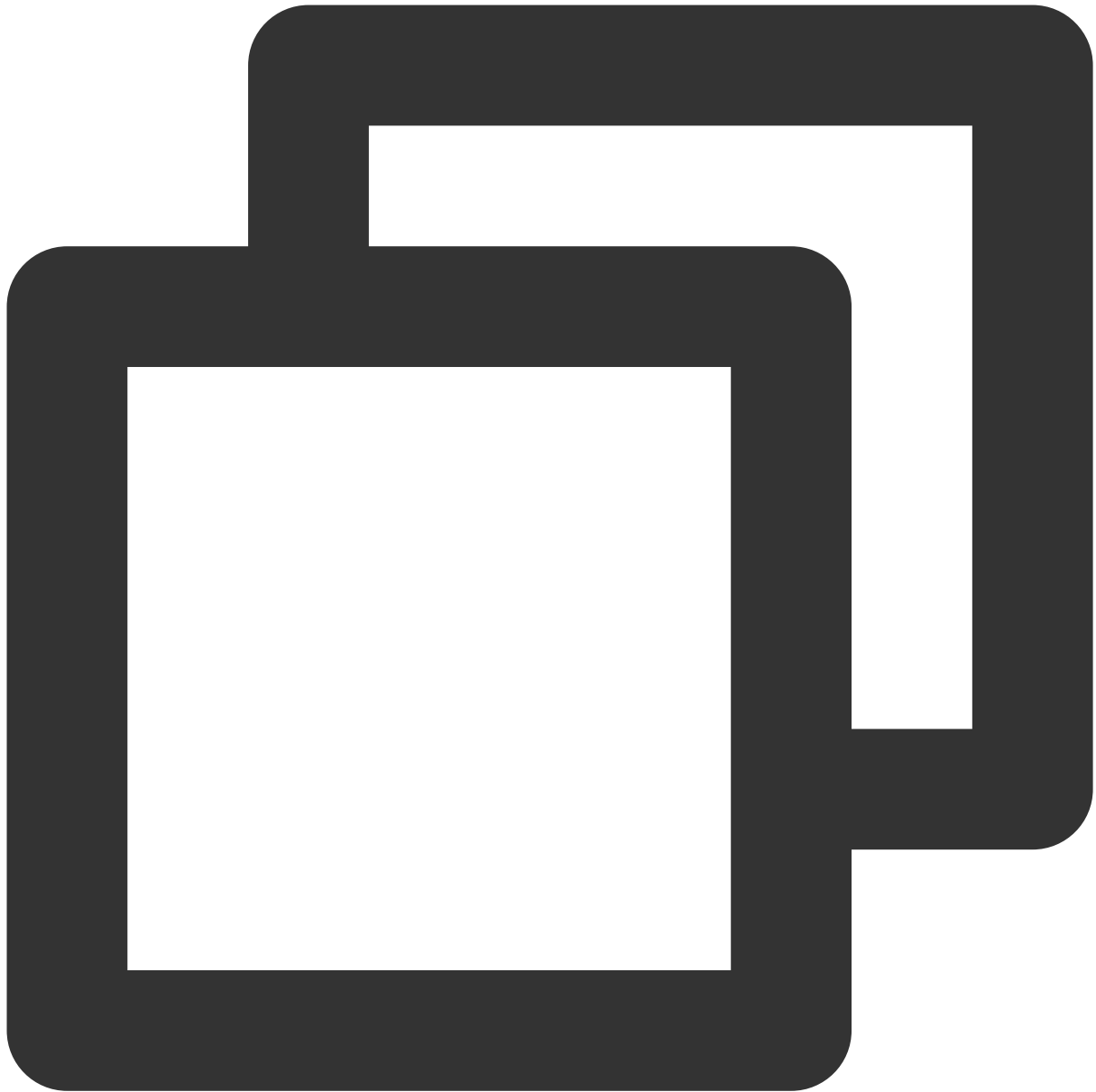
After WebView initialization, add a call interface channel for Java and JS. The purpose is to allow the JS layer to obtain some configuration information from the Java layer:



```
webView.addJavascriptInterface(QAPMJavaScriptBridge.getInstance(), "QAPMAndroidJsBri
```

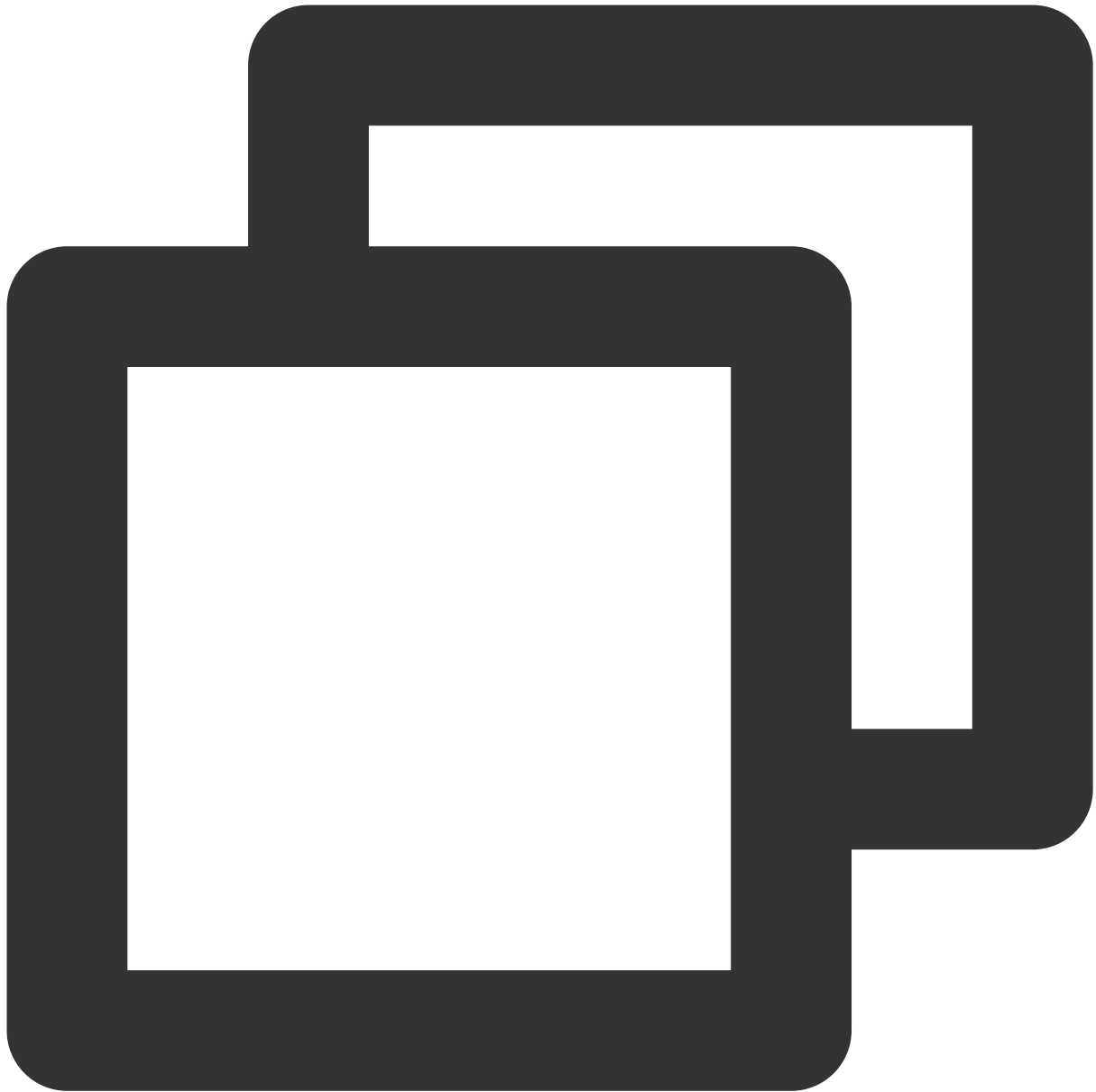
Add the following method in the WebView's `shouldInterceptRequest` code to intercept web-sdk and replace it with local SDK resources. **Make sure to call the following code at the earliest position in this callback.**

**If it is x5, use the following code:**



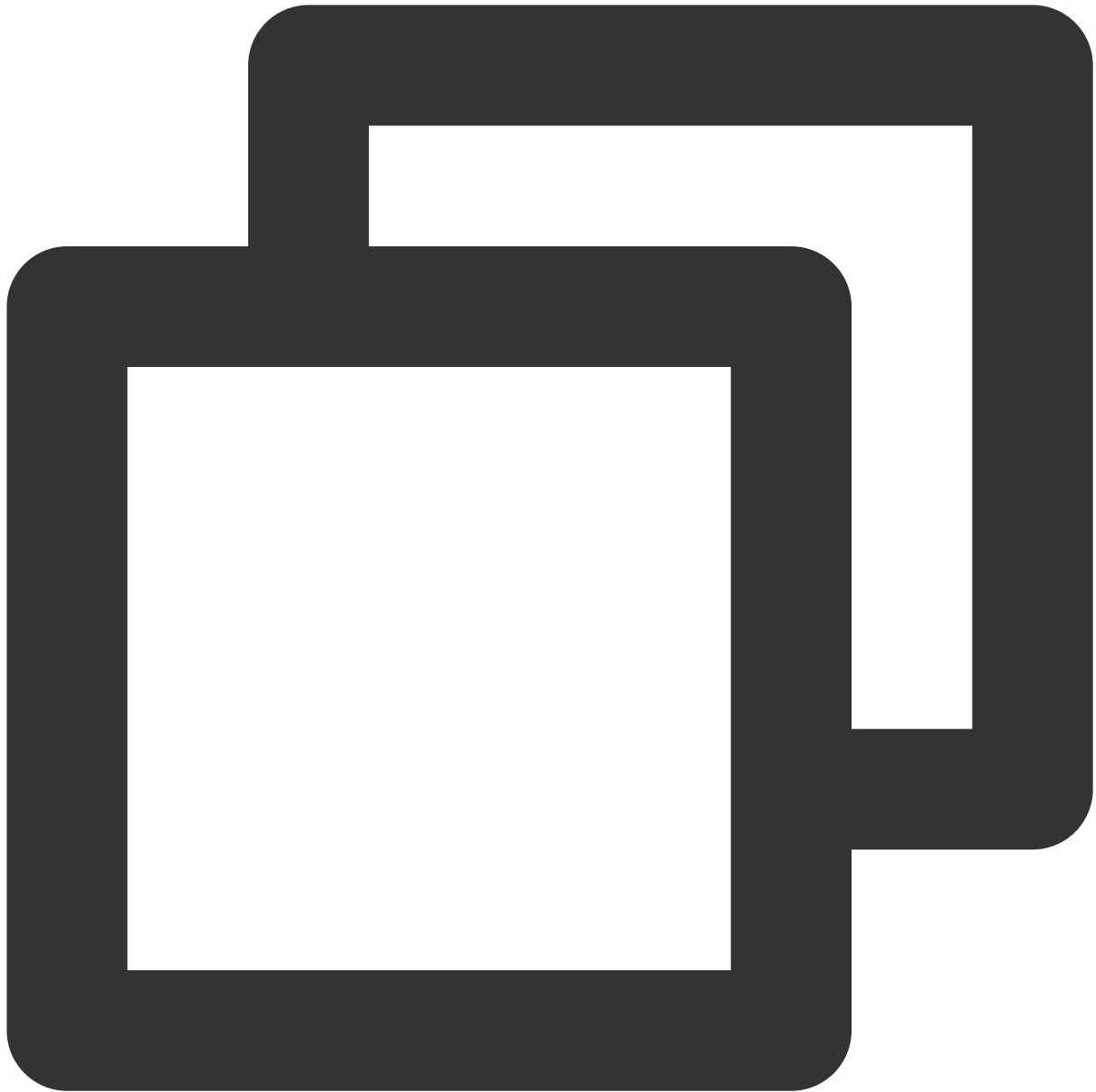
```
@Overridepublic
public WebResourceResponse shouldInterceptRequest (WebView webView, String s) {???
    Object response = QAPMJavaScriptBridge.getInstance().shouldInterceptRequestWith
    if (response != null) {????????
        return (WebResourceResponse)response;???
    }
    return super.shouldInterceptRequest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

**If it is Native WebView, use the following code:**



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

Add the following method in the WebView's onPageFinished code to inject JS script:

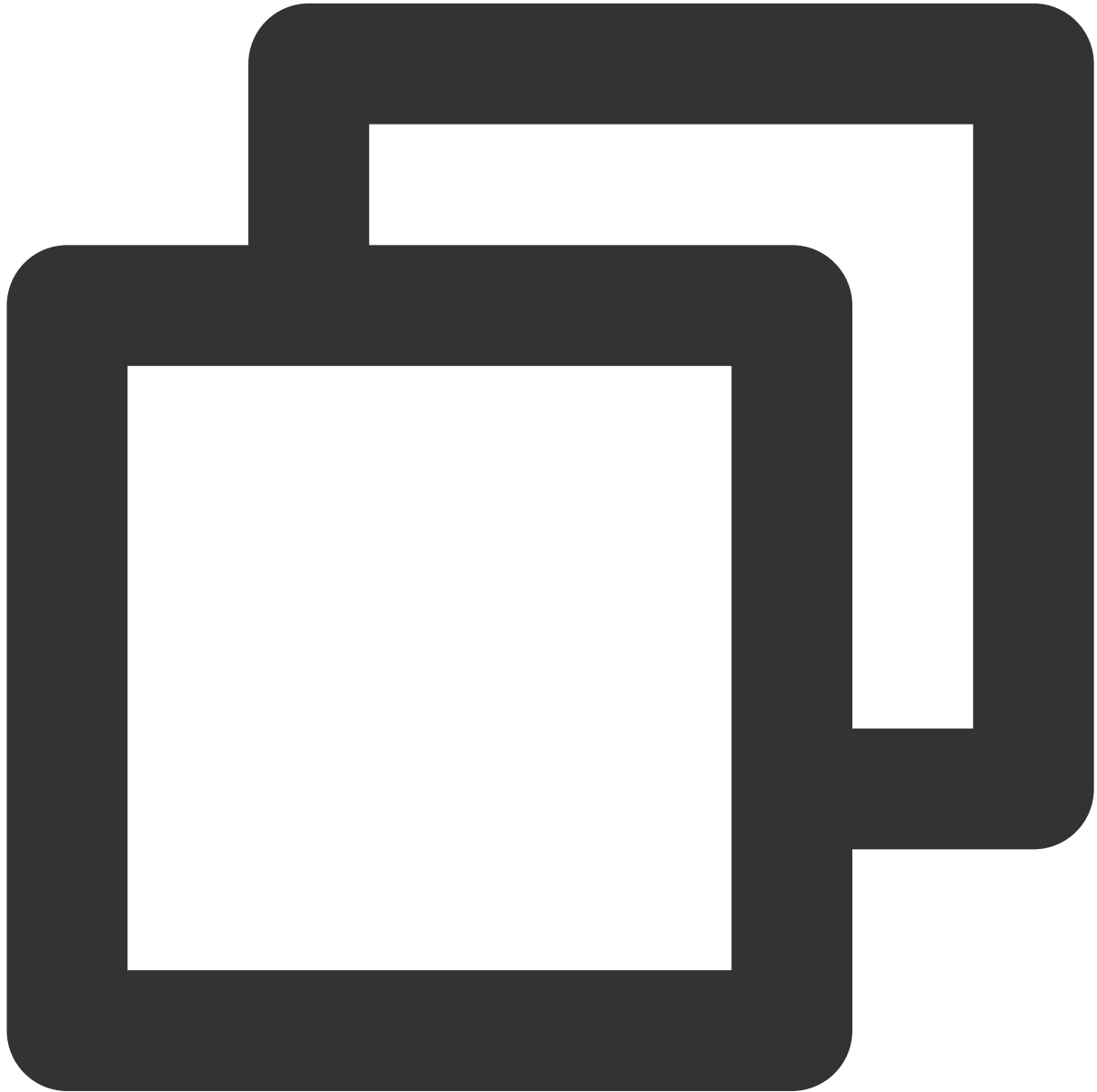


```
webView.setWebViewClient(new WebViewClient(){
    ??? @Override
    ??? public void onPageFinished(WebView view, String url) {
        ???????? super.onPageFinished(view, url);
        ???????? QAPMJavaScriptBridge.getInstance().initFileJS(view);
    ??? }
});
```

## Verifying Whether the Feature Is Working Properly

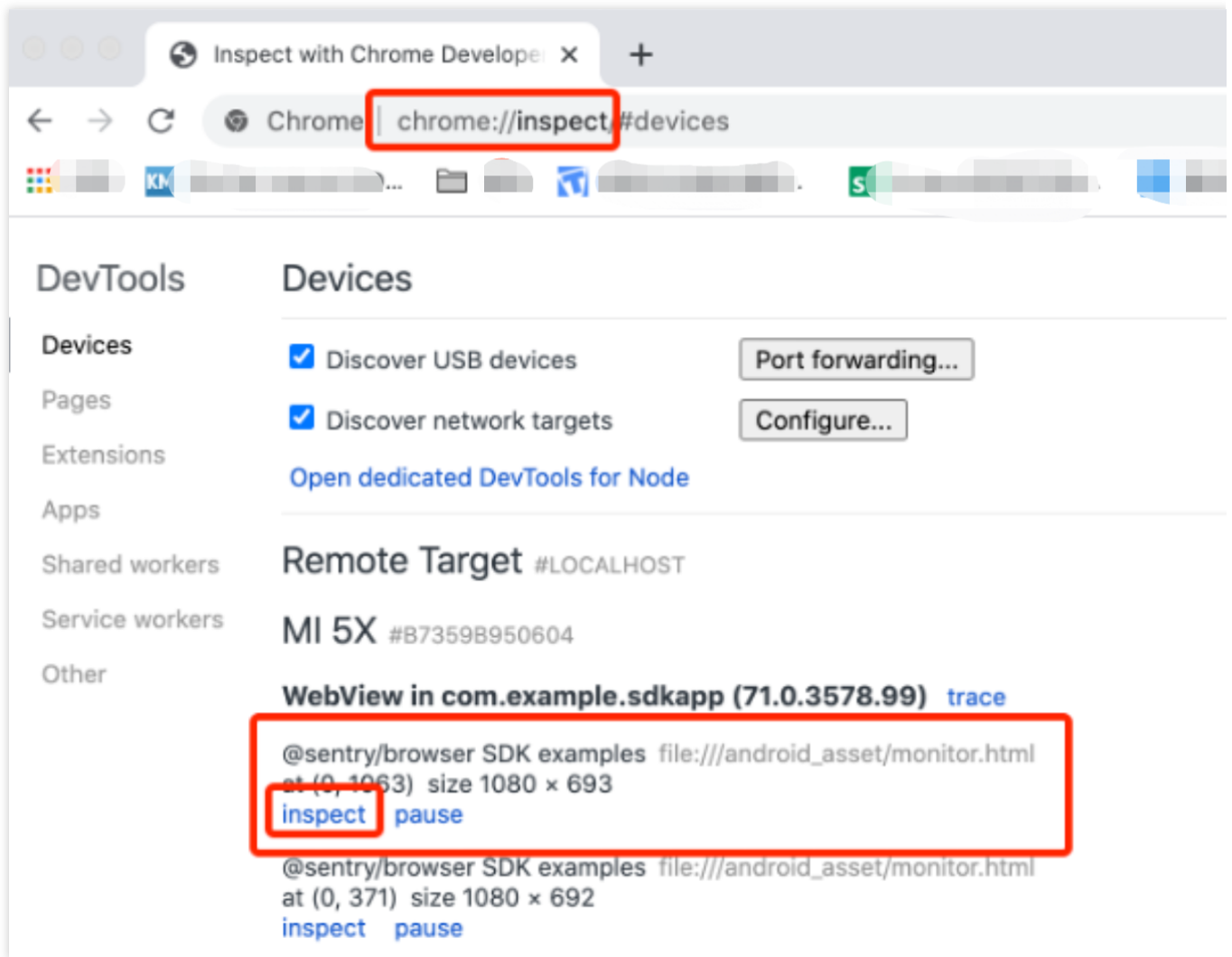
Native WebView, JsError monitoring:

1. Include the following code in the code (for remote debugging purposes).

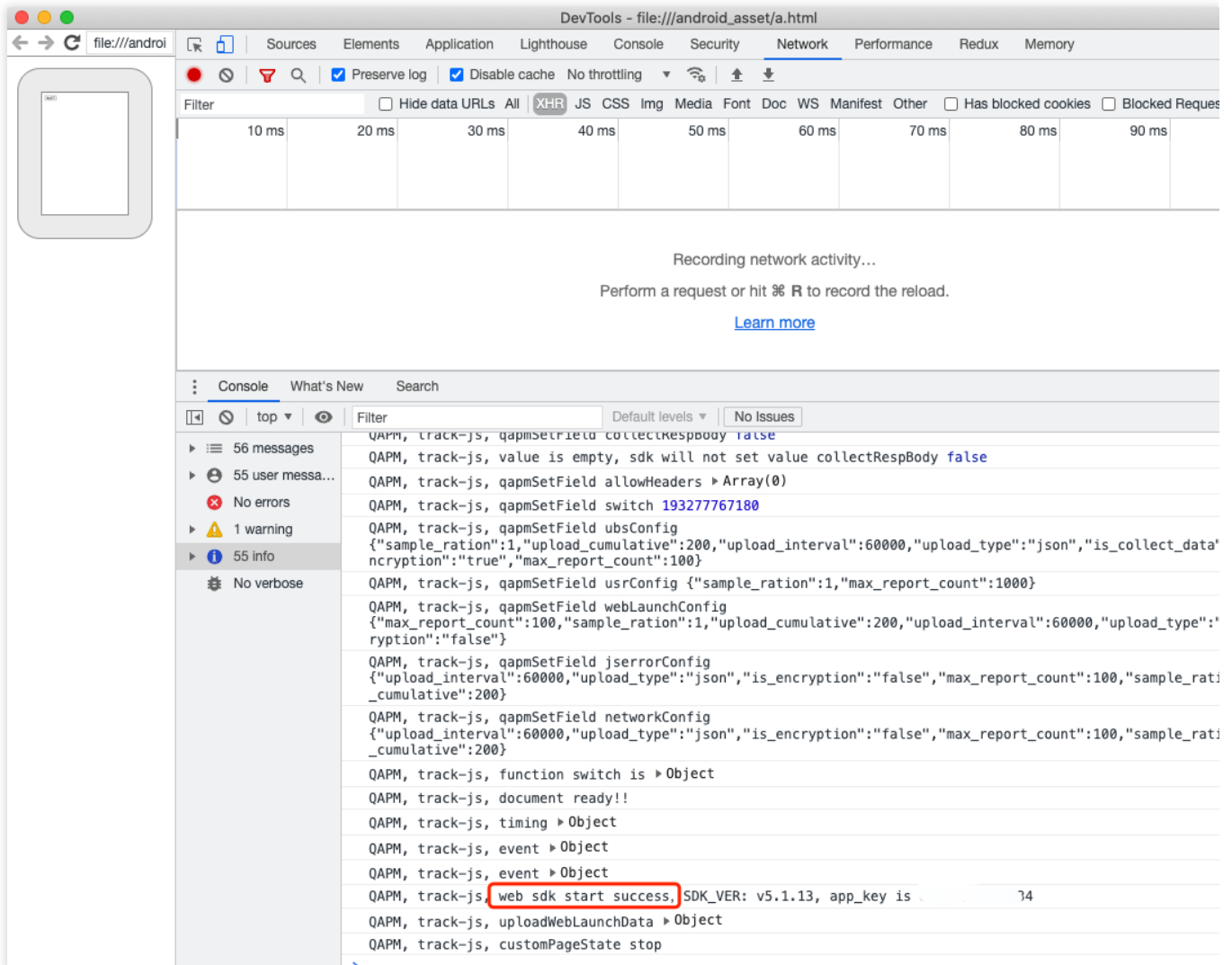


```
WebView.setWebContentsDebuggingEnabled(true);
```

2. Open Google Chrome, and enter `chrome://inspect` in the address bar. In the devices that appear, click **inspect**.

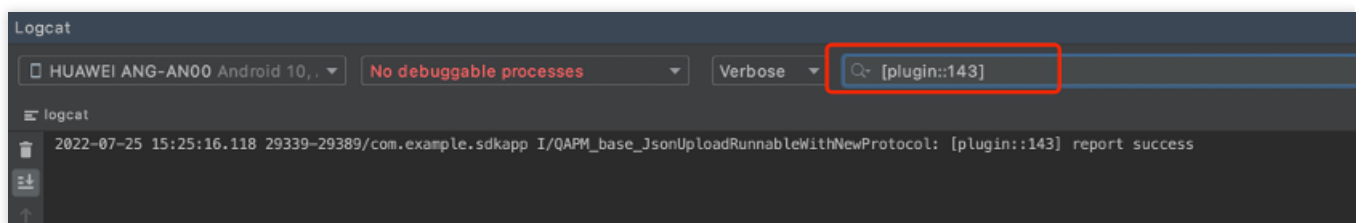


3. Find the Console module query log. If `web start success, vxxx` is displayed, it indicates the WebSDK inject succeeded.



4. Check whether all features are reporting normally. Consider JsError reporting as an example, as follows:  
Retrieval tag: [plugin::143].

The occurrence of a JsError, such as the following log message, indicates successful reporting of JsError data.



The other retrieval tags are as follows:

Page load: plugin::141 (reported immediately after each Web page load is complete).

Network request: plugin::154 (reported when there are network errors and slow requests).

**Note:**

1. To check whether WebView monitoring is normal, examine through browsers like Chrome for debugging.



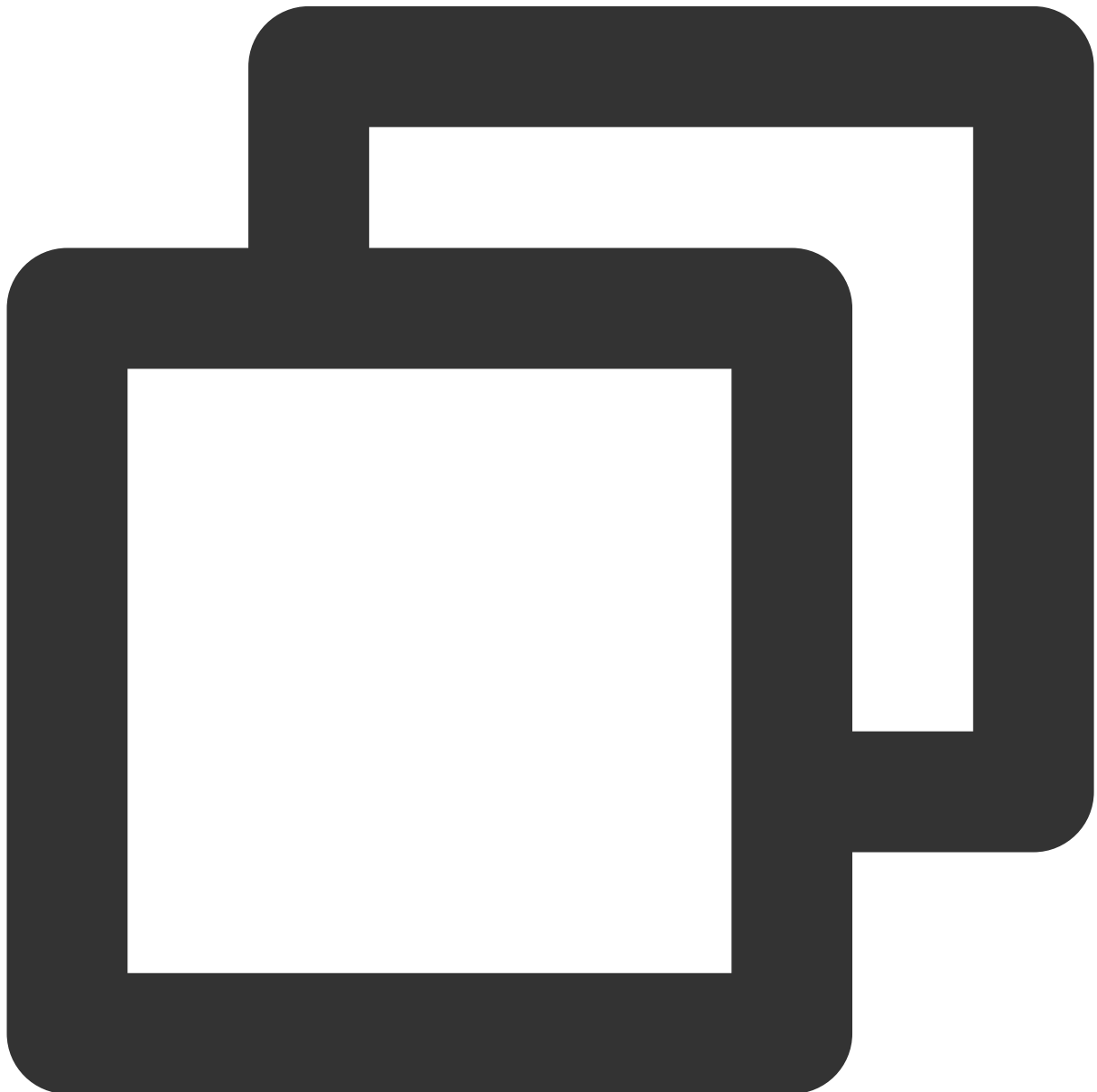
- 
2. Page load is reported in the Issue Case Details only if the page load duration exceeds 3.5s.
3. Network requests are reported in cases of network errors and slow networks.

# Crash and ANR Monitoring

Last updated : 2024-05-14 12:36:06

## Enabling Feature

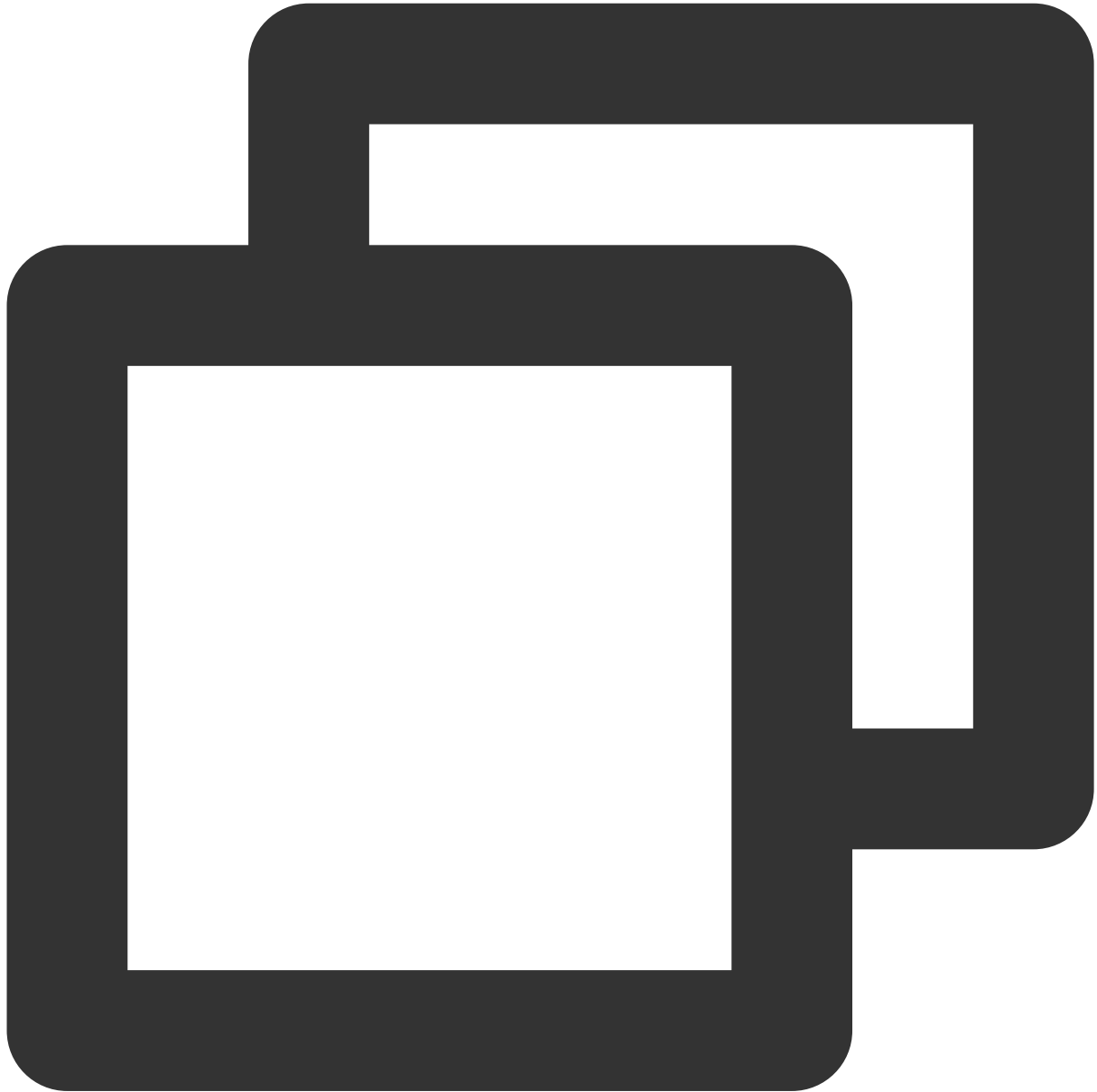
Initialization requires enabling Crash and ANR monitoring, which by default monitors Crash and ANR information.



```
// ModeStable mode by default includes Crash and ANR monitoring.
```

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

QAPM provides APIs for uploading custom log files in case of crashes or ANRs, if necessary. An example is as follows:

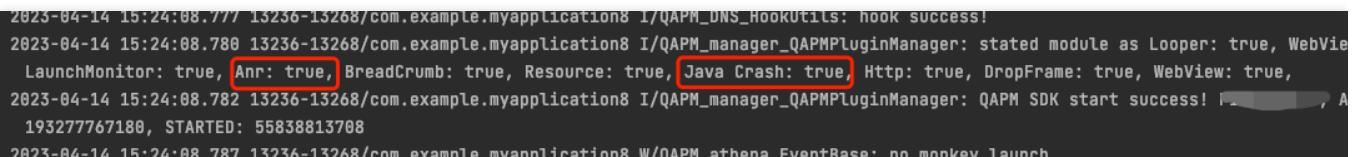


```
QAPM.setProperty(QAPM.PropertyExtraDataListener, new IExtraDataListener() {  
    // This callback is executed when an ANR occurs.  
    @Override  
    public List<String> onAnrExtraFileHandler() {  
        List<String> files = new ArrayList<>();  
        File[] fileArray = new File("xxxx").listFiles();//Enter the folder name at
```

```
        for (File file : fileArray) {
            files.add(file.getAbsolutePath());
        }
        return files;
    }
    // This callback is executed when a crash occurs.
    @Override
    public List<String> onCrashExtraFileHandler() {
        List<String> files = new ArrayList<>();
        File[] fileArray = new File("xxxx").listFiles();//Enter the folder name at
        for (File file : fileArray) {
            files.add(file.getAbsolutePath());
        }
        return files;
    }
});
```

## Verifying Whether the Feature Is Working Properly

Retrieval tag: QAPM\_manager\_QAPMPluginManager



2023-04-14 15:24:08.777 13236-13268/com.example.myapplication8 I/QAPM\_DNS\_HookUtils: hook success!  
2023-04-14 15:24:08.780 13236-13268/com.example.myapplication8 I/QAPM\_manager\_QAPMPluginManager: stated module as Looper: true, WebView: true, LaunchMonitor: true, Anr: true, BreadCrumb: true, Resource: true, Java Crash: true, Http: true, DropFrame: true, WebView: true,  
2023-04-14 15:24:08.782 13236-13268/com.example.myapplication8 I/QAPM\_manager\_QAPMPluginManager: QAPM SDK start success! [REDACTED], A  
193277767180, STARTED: 55838813708  
2023-04-14 15:24:08.787 13236-13268/com.example.myapplication8 W/QAPM\_athena\_EventBase: no monkey launch

Retrieval tag: QAPM\_crash

The following log message in case of crashes or ANRs indicates that QAPM has collected this exception:

The screenshot shows the Logcat interface in Android Studio. The search filter is set to 'QAPM\_crash'. The log output shows a stack trace starting with 'at android.view.View.performClick' and ending with 'at com.android.internal.os.ZygoteInit.main'. Below the stack trace, there are several log messages from the QAPM plugin, including 'Calling collector com.tencent.q...', 'Collector com.tencent.q...', and 'Handing Exception on to default Exce...'. The bottom status bar indicates 'Gradle build finished in 1 s 520 ms (5 minutes ago)'.

Retrieval tag: plugin::144

The following log message indicates that QAPM has successfully reported this exception. An example is as follows:

The screenshot shows the Logcat interface in Android Studio. The search filter is set to '[plugin::144]'. The log output shows a single log message: '2022-05-18 16:27:41.873 26953-27034/com.example.sdkapp I/QAPM\_base\_FileUploadRunnableWithNewProtocol: [plugin::144]'. The bottom status bar indicates 'No debuggable processes'.

The other crash retrieval tags are as follows:

ANR: [plugin::140].

NativeCrash: [plugin::146].

#### Note:

To avoid lagging, keep the logic in the interface callbacks as simple and straightforward as possible.

Uploaded files must be less than 20 MB in size. Files larger than the limit will not be uploaded. Select helpful log files.

Crash events can be viewed on the Mobile Monitoring Crash page, and ANR rates can be viewed in the Overview page.

# Lag and Frame Rate Monitoring

Last updated : 2024-05-14 12:36:06

## Prerequisites

Integration and Initialization has been [Integration and Initialization](#).

## Feature Configuration

### Enabling Monitoring

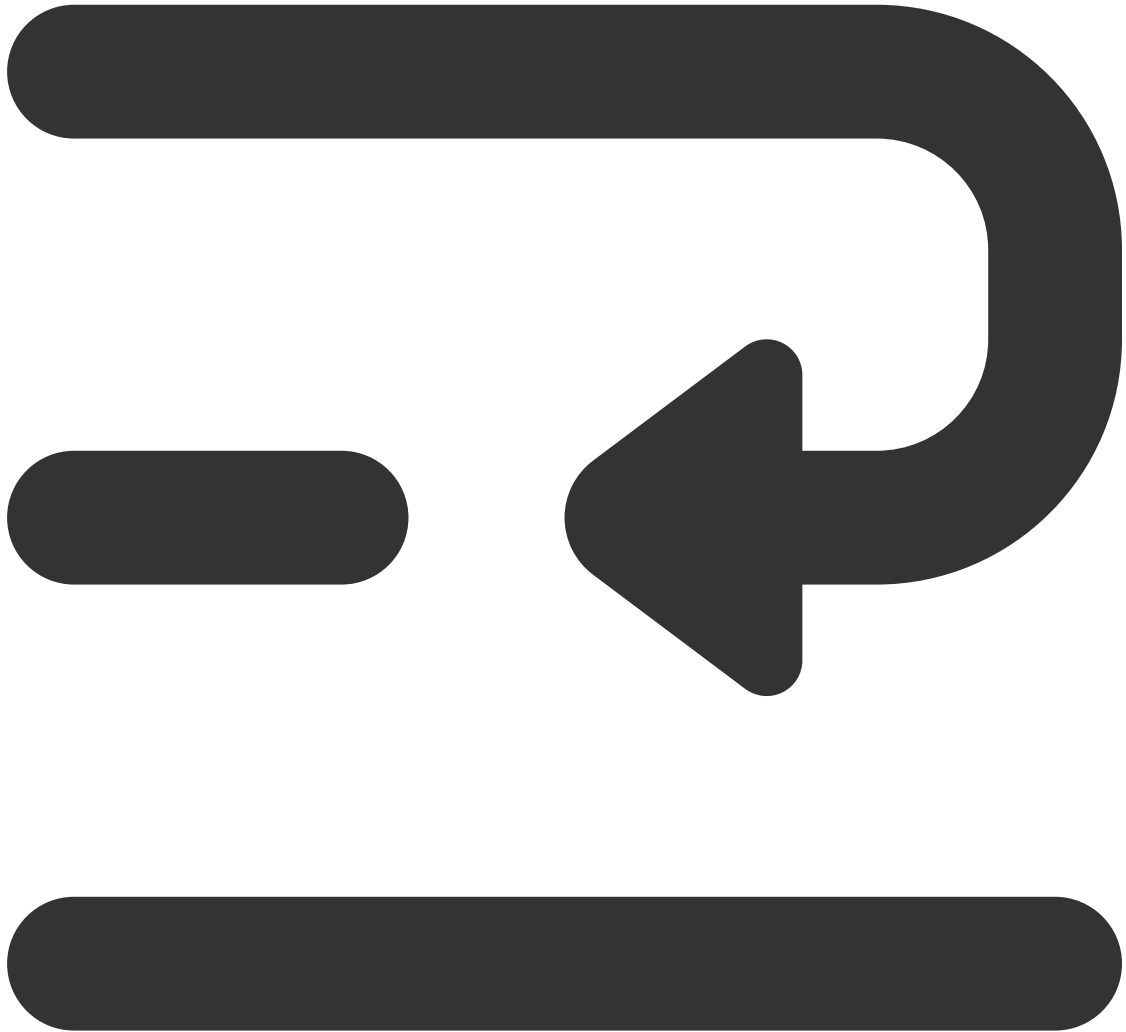
Initialization requires enabling lag monitoring. Lag doesn't need instrumentation, while frame loss rate requires additional instrumentation. It is recommended to add tracking on scrolling lists, such as (ListView, GridView, and RecyclerView).

### Frame Loss Rate Instrumentation

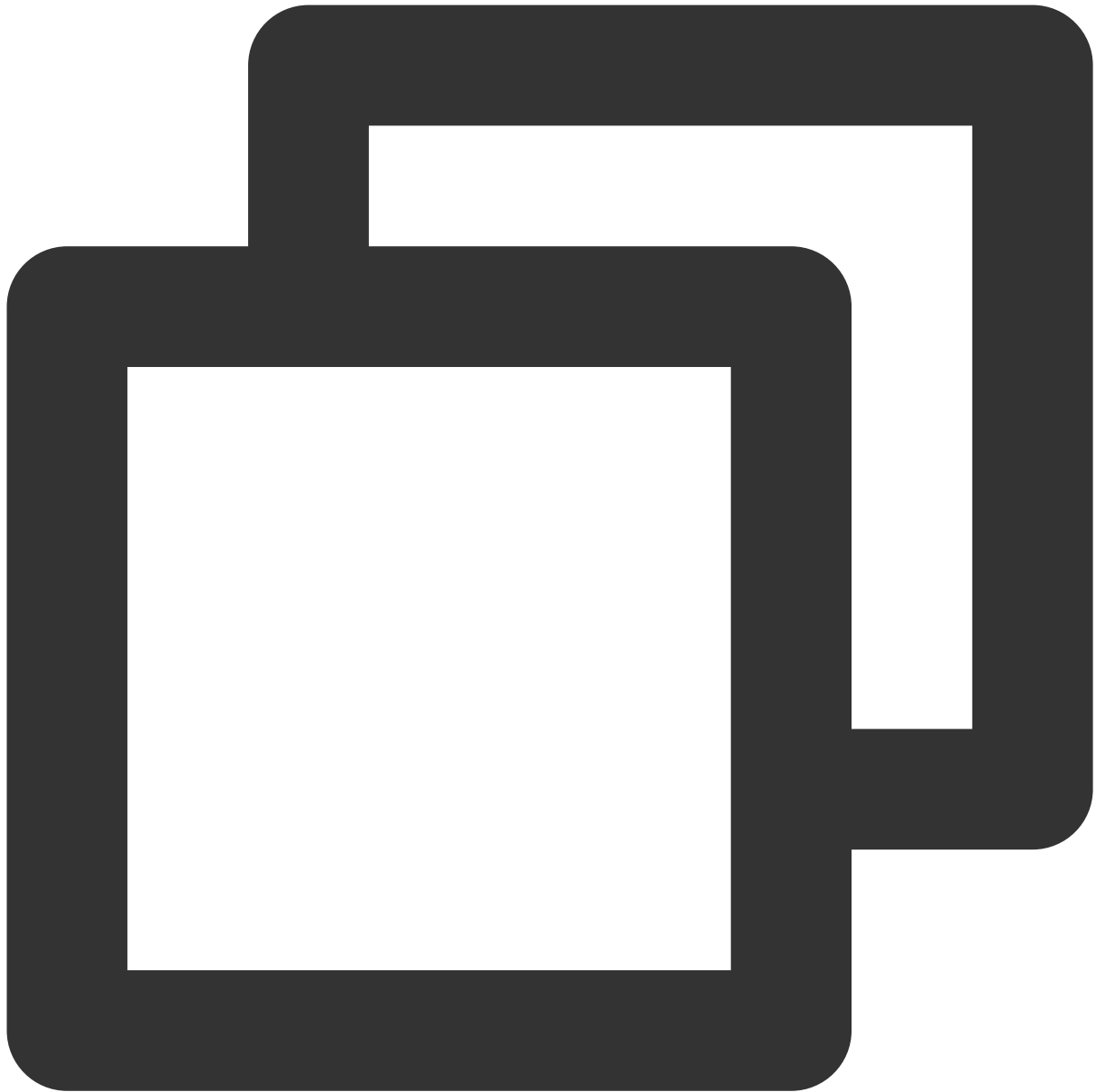
Call `QAPM.beginScene("xxx scrolling", QAPM.ModeDropFrame)` before each scroll.

Call `QAPM.endScene("xxx scrolling", QAPM.ModeDropFrame)` after a scroll ends.

This can generally be achieved by overriding the scrolling component's `onScrollStateChanged` method, as shown below:





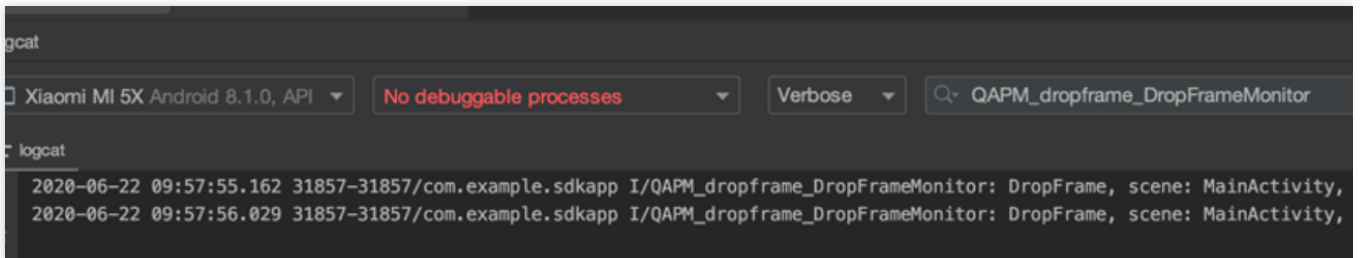


```
@Override
public void onScrollStateChanged(AbsListView view, int scrollState) {
    if (scrollState == AbsListView.OnScrollListener.SCROLL_STATE_IDLE) {
        QAPM.endScene("xxx scrolling", QAPM.ModeDropFrame); //xxx scrolling name ca
    } else {
        QAPM.beginScene("xxx scrolling", QAPM.ModeDropFrame); //xxx scrolling name c
    }
}
```

## Verifying Whether the Feature Is Working Properly

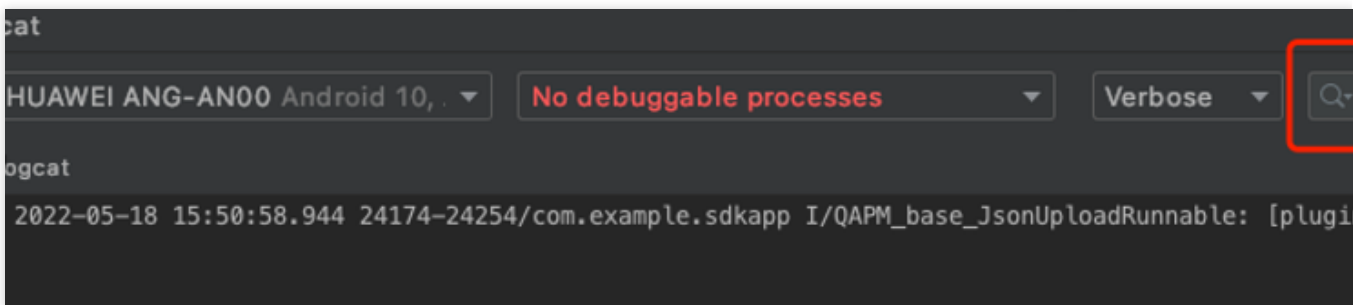
Retrieval tag: QAPM\_dropframe\_DropFrameMonitor

After a scroll ends (endScene calling), the following log message indicates that the frame loss rate data has been stored in the local database:



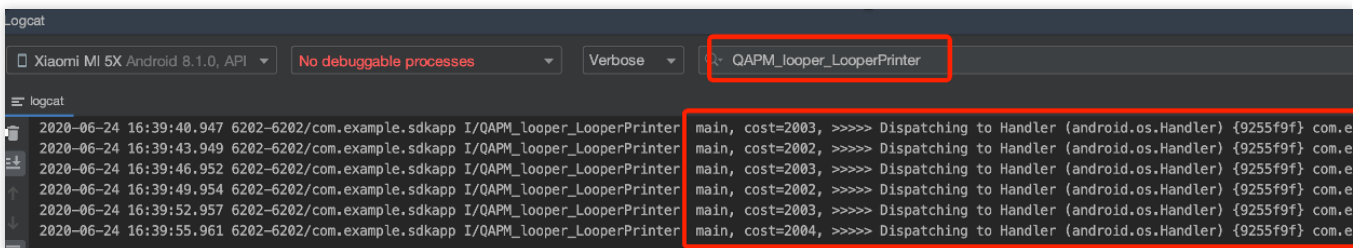
Retrieval tag: [plugin:::101]

The following log message indicates successful reporting of frame loss data that is stored in the app's local database.

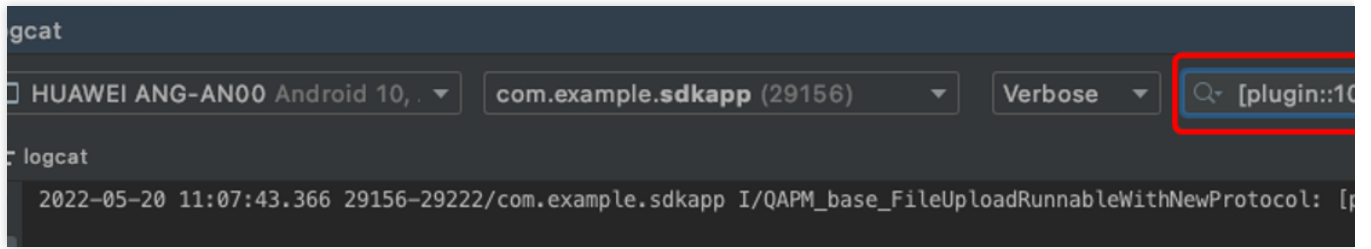


Retrieval tag: QAPM\_looper\_LooperPrinter

The following log message indicates that Lag Monitoring is functioning properly:



The following log message indicates that Lag Reporting is functioning properly:



# Startup Monitoring

Last updated : 2024-05-14 12:36:06

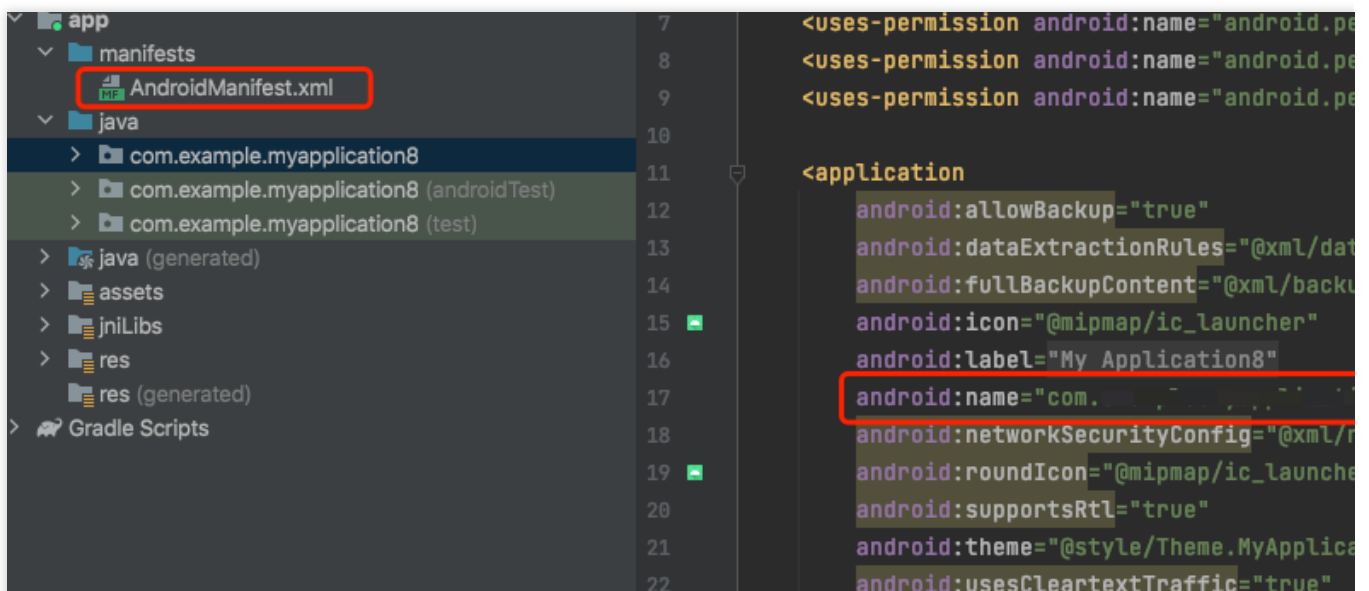
Startup monitoring requires the use of the qapm-plugin plugin for instrumentation during compilation. The default instrumentation points are the various lifecycles of the Application and Activity. In the App SDK, the default startup time is measured from Application's attachBaseContext to the end of onResume of the first Activity.

## Prerequisites

The qapm-plugin has been configured in the app-level build.gradle.

## Configuration Process

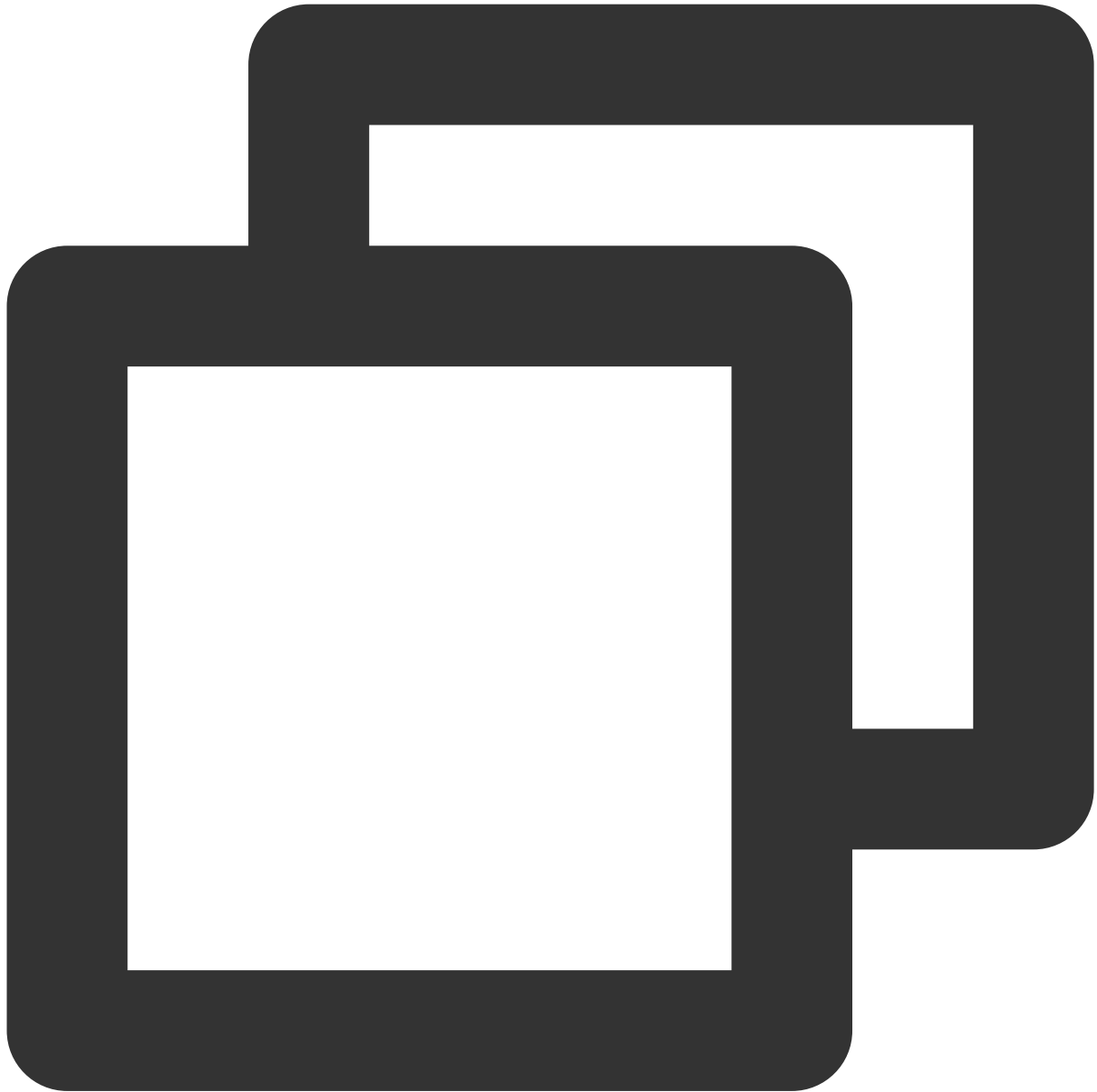
1. Manually add an Application subclass, such as BaseApplication (the name is not restricted, and the subclass does not need to implement any methods or add any attributes).
2. In the AndroidManifest.xml file, add the android:name attribute to the application node, with the value being "package name+Application subclass name".



## Additional Tracking

If you want to measure the execution time of certain methods within the startup interval, additional tracking is required, as shown below:

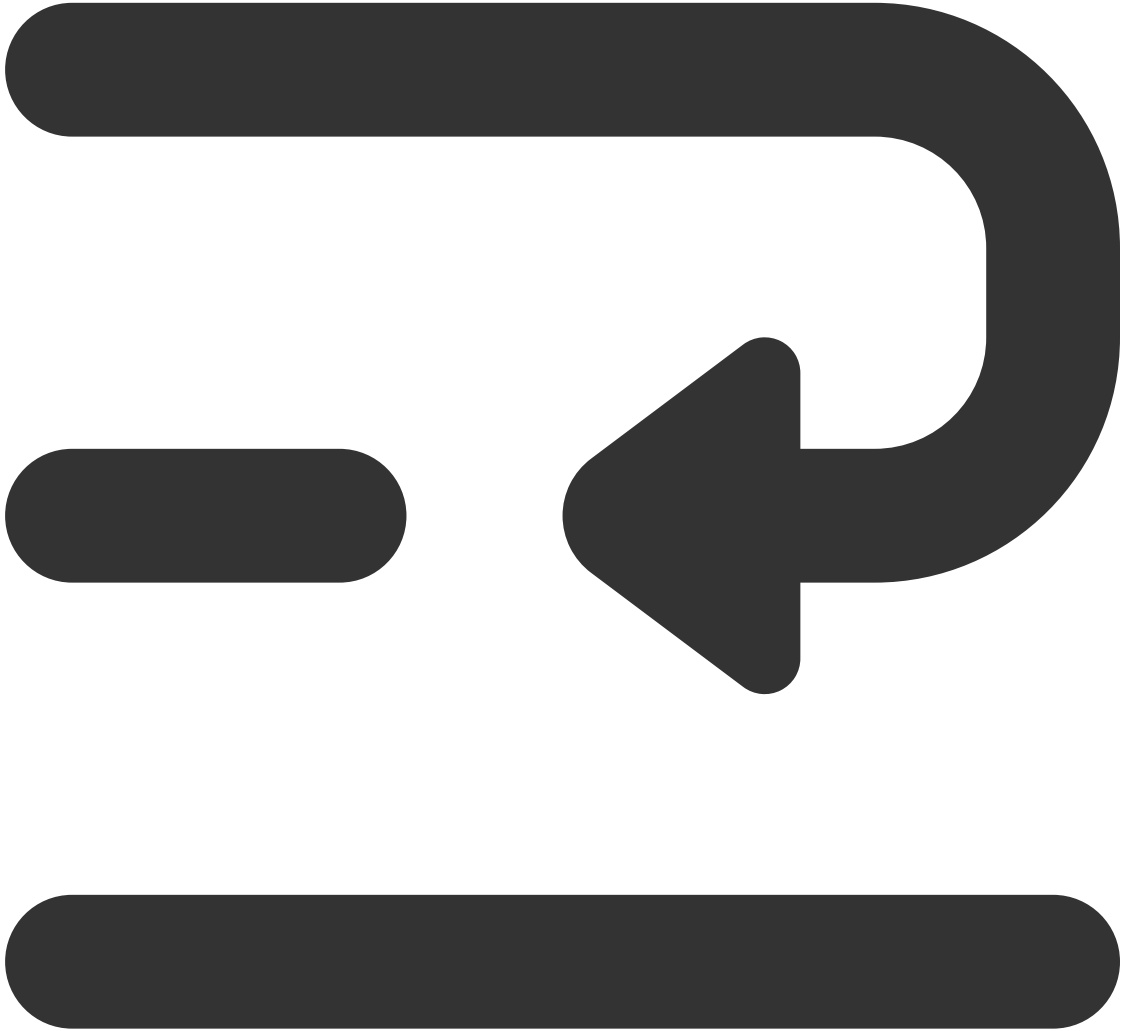
Refer to the code:

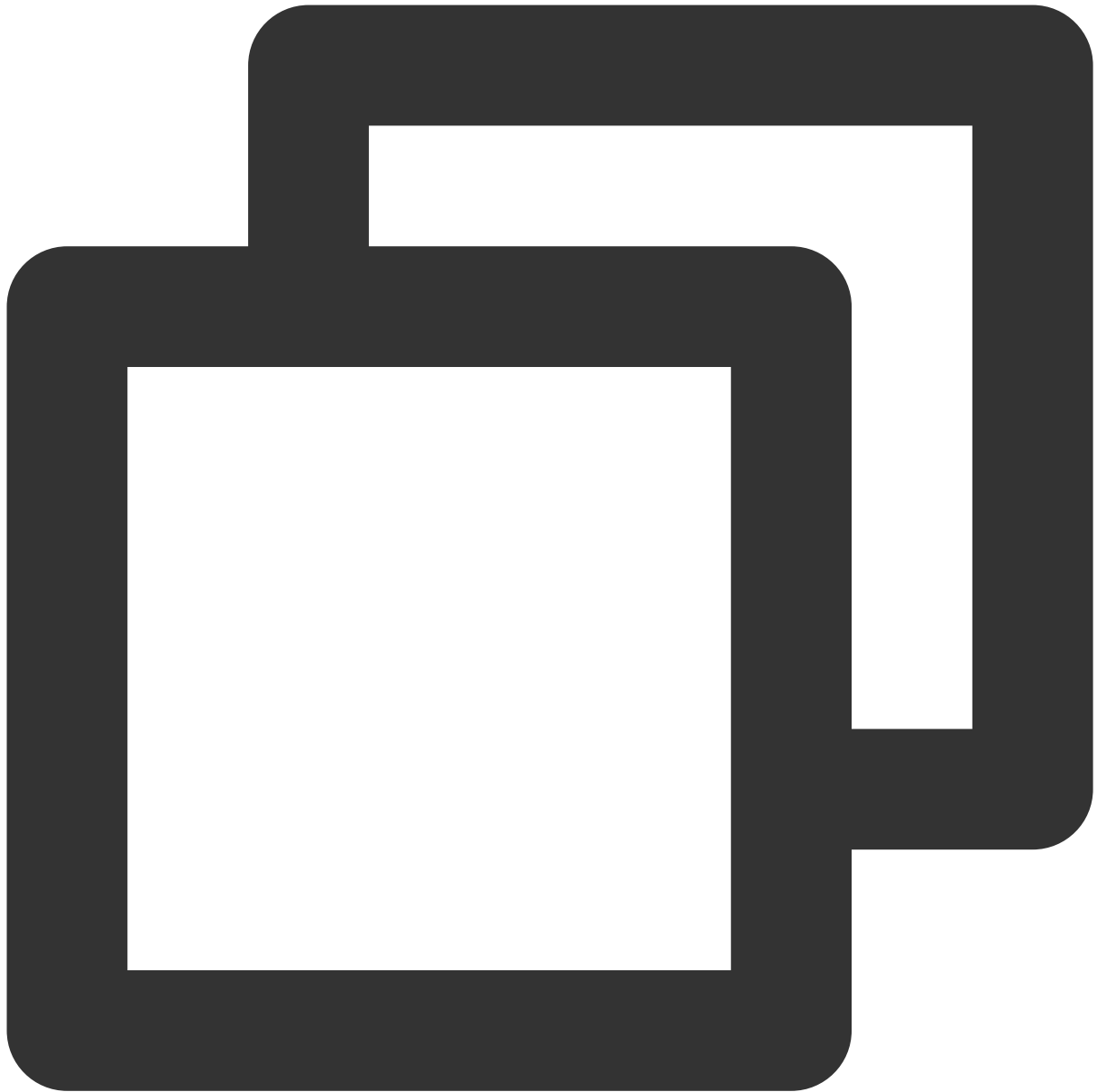


```
QAPM.beginScene(StageConstant.QAPM_APPLAUNCH, "Method Name", QAPM.ModeResource);  
/**Service logic*/  
QAPM.endScene(StageConstant.QAPM_APPLAUNCH, "Method Name", QAPM.ModeResource);
```

If you want to set your own end point for the startup, additional tracking must be done within 20s after the first Activity calls onResume, as shown below:

Refer to the code:



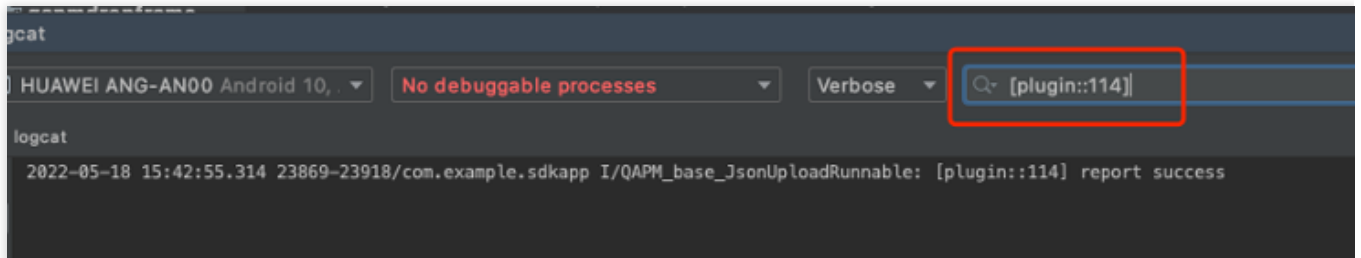


```
/**  
 * Users who need to customize the end point must do so within 20s after onResume.  
 */  
QAPM.endScene(StageConstant.QAPM_APPLAUNCH, QAPM.ModeResource);
```

## Verifying Whether the Feature Is Working Properly

If the following log message appears 20 seconds after each startup or switch to background, it indicates successful reporting of startup metric data.

Retrieval tag: [plugin::114]



### Note:

It requires the qapm-plugin for instrumentation and you must manually add an Application subclass. Otherwise, it will not work.

Individual event data will be reported only if the total startup time exceeds 2.5s.

Launch issue data can be viewed in [Mobile App Performance Monitoring > Startup > Issue List](#).

## Calculation

### Cold Startup:

Occurs after the app starts from the device or after the system terminates the app for the first time.

Android calculation method: `mainActivityOnResume_end - attachBaseContext_start`.

iOS calculation method: Time of the first frame of the first page UI displayed on screen - App process creation time.

### Initial Startup:

Indicates the first launch after the app installation, which is a special case of cold startup.

Android calculation method: `mainActivityOnResume_end - attachBaseContext_start`

iOS calculation method: Time of the first frame of the first page UI displayed on screen - App process creation time.

### Warm Startup:

Under the premise that process and Activity instances still exist (for iOS, the app is in the background and alive). If the app switches to the background for three minutes and then switches back to the foreground, it is defined as a warm startup.

Android calculation method: `activityOnResume_end - activityOnRestart_start`.

iOS calculation method: `ApplicationDidBecomeActive - ApplicationWillEnterForeground`.

### Startup Duration:



Total boot time / Total boot count

# Operation Guide

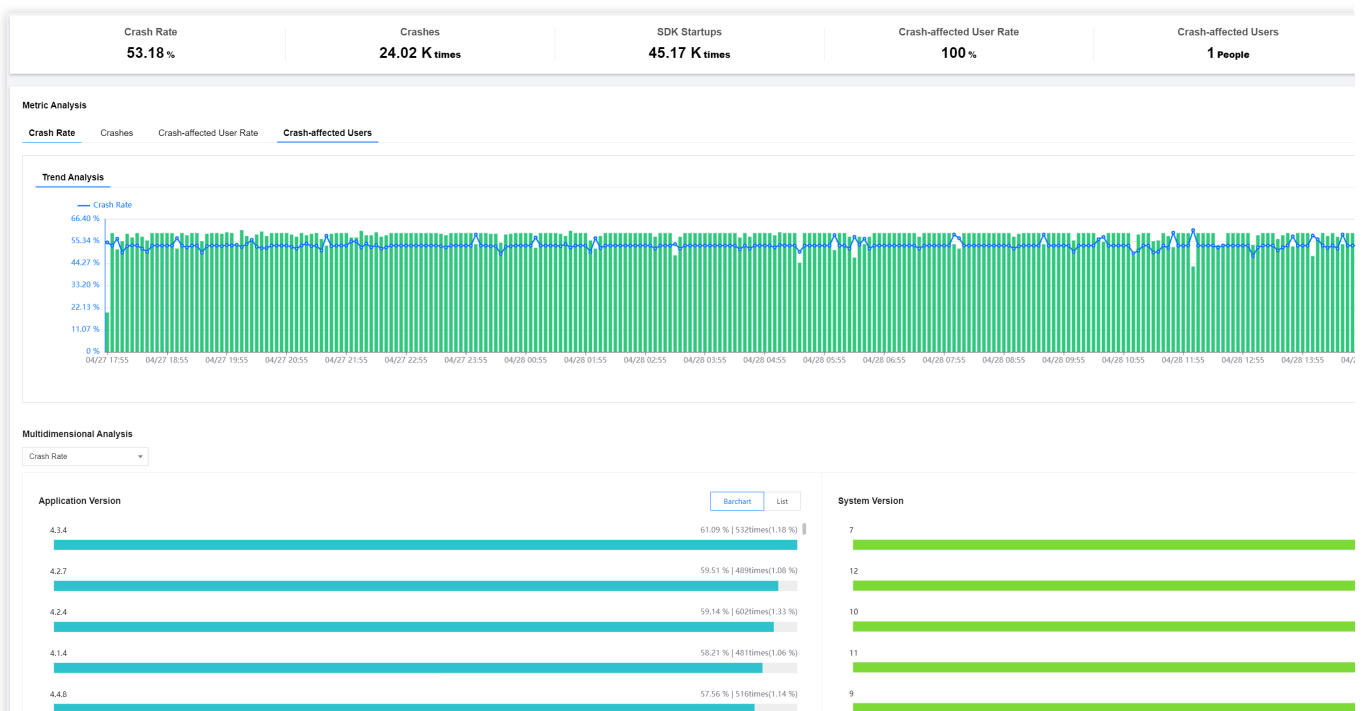
## Crash

Last updated : 2024-05-13 18:03:25

Terminal Performance Monitoring aggregates key characteristics from individual crash incidents, facilitating root cause analysis of App crashes.

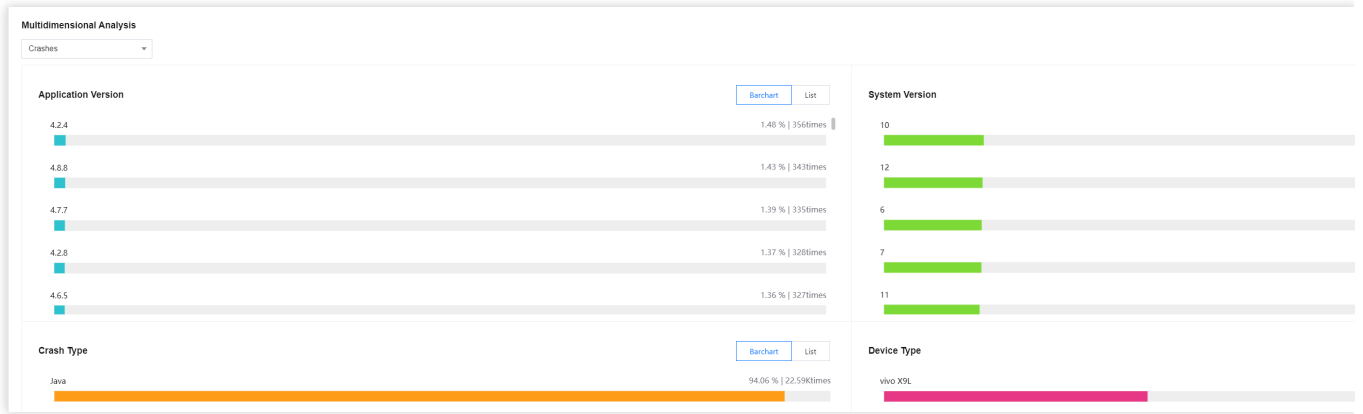
## Feature Entry

1. Log in to [Tencent Cloud Observability Platform](#).
2. In the left navigation bar, select **Mobile App Performance Monitoring > Crash**. Select the business system, app, and time range to analyze crashes.



## Multidimensional Analysis

The Multidimensional Analysis page shows the analysis of key metrics from multiple dimensions such as app version, system version, crash type, device type, and app status. It facilitates targeted root cause analysis of specific crash events.



## Crash Issue List

The crash issue list shows crashes of all devices. You can quickly filter crashes by error type, device ID, function, or filename. You can also click **Issue Description** to view the details of crashes and pinpoint the root causes of crashes.

**Crash Issue List**

All Exception Types | Device ID | Please enter the device ID | Specific Function or Filename | Please enter the function c

Issue Description	Crash-affected Users (Proportion)	Crashes (Prop)
<p><b>ID: 9</b></p> <p>java.lang.UnsatisfiedLinkError java.lang.System.loadLibrary(System.java) com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) android.view.View.performClick(View.java)</p>	1 (20%)	3559 (20.06%)
<p><b>ID: 2</b></p> <p>native_crash app_process32() libc.so() com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile)</p>	1 (20%)	3549 (20.00%)
<p><b>ID: 3C</b></p> <p>java.lang.OutOfMemoryError com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) android.view.View.performClick(View.java)</p>	1 (20%)	3547 (19.99%)
<p><b>ID: 1</b></p> <p>java.lang.RuntimeException com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) android.view.View.performClick(View.java)</p>	1 (20%)	3545 (19.98%)
<p><b>ID: 3</b></p> <p>java.lang.NullPointerException com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) com.tencent.tenkit.demo.performance.activity.MainActivity\$2.onClick(SourceFile) android.view.View.performClick(View.java)</p>	1 (20%)	3542 (19.96%)

## Metrics Description

Related Metrics are as follows:

Metric Name	Metrics Description

Crash Rate	Number of crashes/Number of app launches within a specified time range
Crash-affected User Rate	Number of users affected by crashes/Number of users launching the app within a specified time range
Crashes	Number of crashes within a specified time range
Crash-affected Users	Number of users affected by crashes within a specified time range
Crash Type	Classified into Java crashes and Native crashes based on the location of the crash occurrence
SDK Startups	Number of app launches

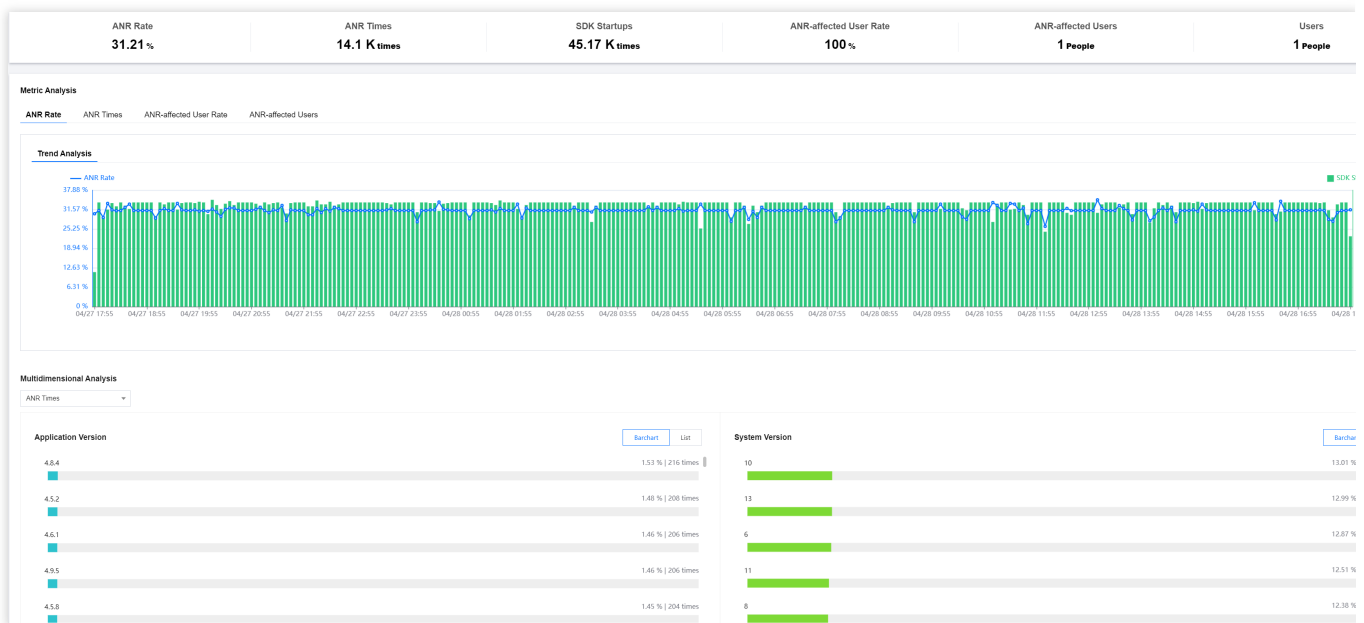
# ANR

Last updated : 2024-05-13 18:03:25

Mobile Performance Monitoring (MPM) aggregates key characteristics of individual Application Not Response (ANR) cases, facilitating root cause analysis of ANR issues for your app.

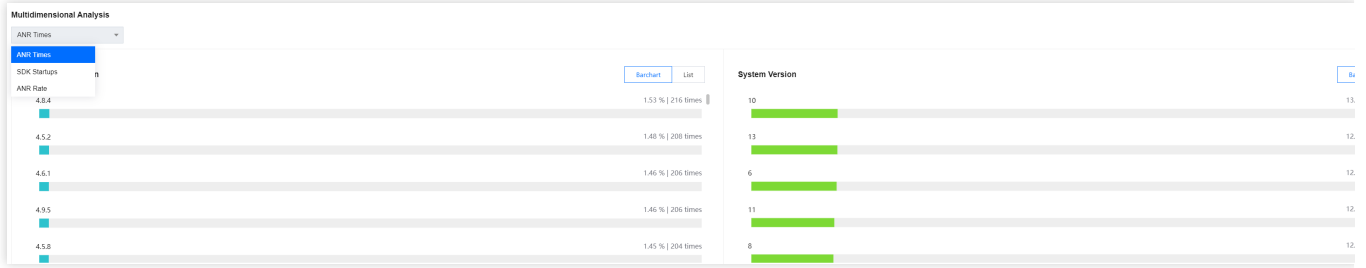
## Feature Entry

1. Log in to [Tencent Cloud Observability Platform](#).
2. In the left sidebar menu, select **Mobile App Performance Monitoring > Anr**.
3. Select the business system, app, and time range to analyze ANR issues.



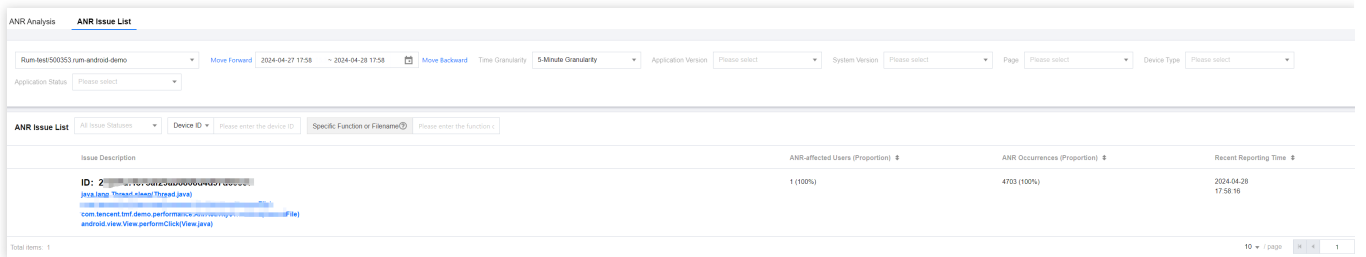
## Multidimensional Analysis

The Multidimensional analysis page shows the analysis of key metrics from multiple dimensions such as ANR count, number of launches, and ANR rate. It facilitates targeted root cause analysis of specific ANR issues.



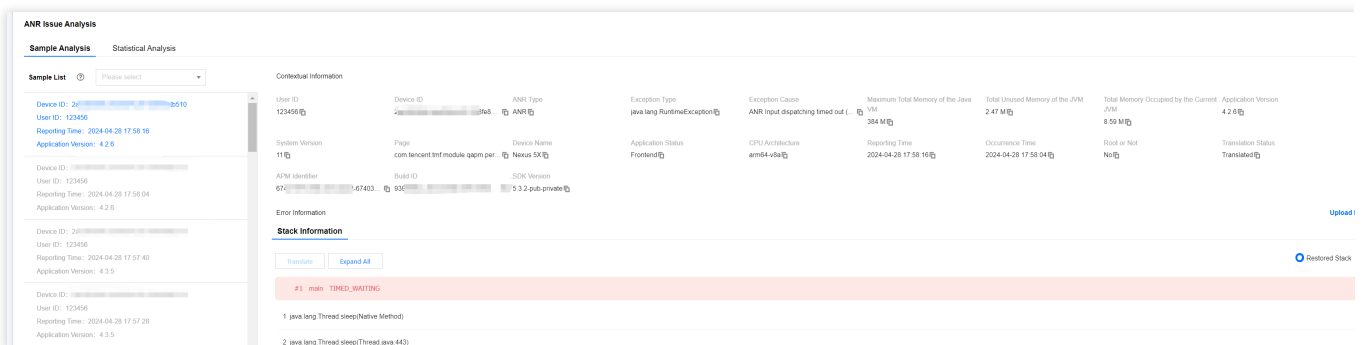
## ANR Issue List

The ANR issue list allows you to view, search, sort, and manage clustering issues. You can enter the issue details page by clicking the View Details button.



## ANR Issue Details

The details page provides multi-dimensional statistics for a category of issues and analysis of individual cases. You can check the details of an issue by clicking the corresponding issue description.



## Metrics Description

ANR-related Metrics are described as follows:

Metric Name	Metrics Description
ANR Rate	Number of devices experiencing ANR/Total number of devices within a specified time range
ANR Times	Number of ANRs occurring in the app within a specified time range
SDK Startups	Number of app launches
ANR-affected User Rate	Number of users affected by ANR/Total number of users launching the app within a specified time range
ANR-affected Users	Number of users affected by ANR within a specified time range
Users	Total number of users launching the app

# Network

Last updated : 2024-05-13 18:03:25

Network issues are analyzed using metrics such as Throughput, Requests, Network Response Time, Slow Request Proportion, HTTP Error Rate, Network Error Rate, and TCP Connection Establishment Time.

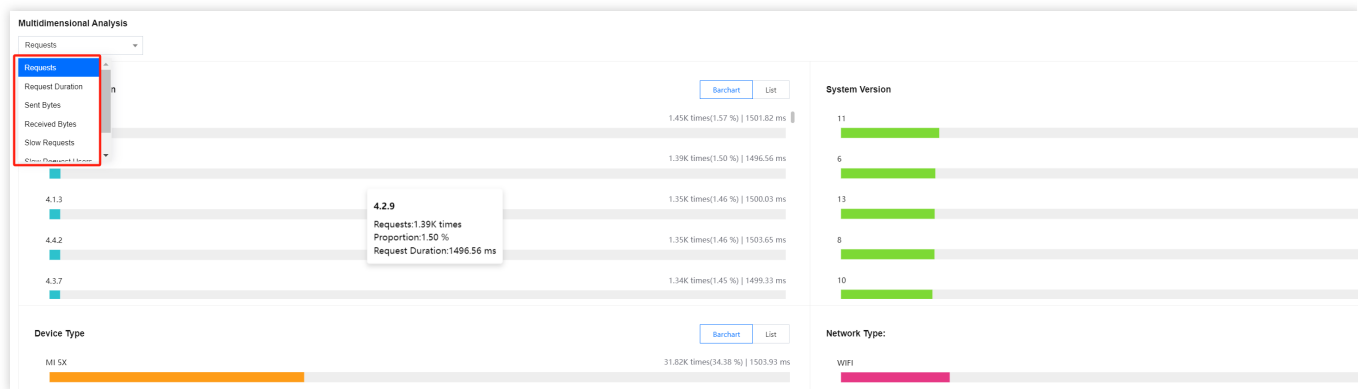
## Feature Entry

1. Log in to [Tencent Cloud Observability Platform](#)..
2. In the left navigation bar, select **Mobile App Performance Monitoring** > **Network**, You can check network issue analysis from multiple dimensions such as business system, app, and time range.

## Multidimensional Analysis

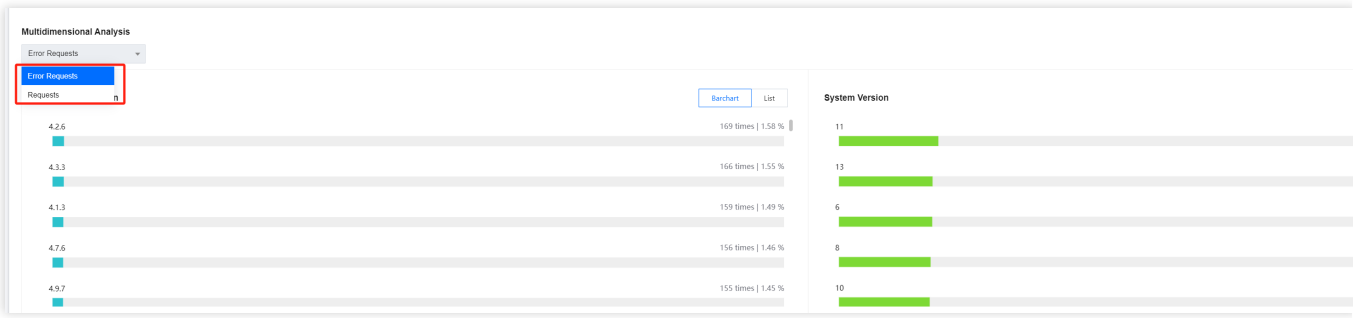
The multidimensional analysis page shows the analysis of key metrics from multiple dimensions such as app version, system version, domain name, URL, device type, network type, region, and internet service provider. It facilitates targeted root cause analysis of specific slow/error requests.

### Slow Requests



### Error Requests





## Slow Request Issue List

The slow request list shows slow requests of all devices. You can quickly filter slow-loading devices by issue type, device ID, specific function, or file name. You can also click the related link under **Issue Description** to view details of slow requests and pinpoint the root cause of slow app requests.

The screenshot shows the 'Slow Request Issue List' interface. It includes a search bar with filters for 'Run-test500353', '2024-04-27 17:58', and '5-Minute Granularity'. Below the filters is a table with the following columns: Issue Description, Slow Request Users (Proportion), Slow Requests (Proportion), and Request Duration (ms).

Issue Description	Slow Request Users (Proportion)	Slow Requests (Proportion)	Request Duration (ms)
ID: [redacted] m.zhipin.com/wapizpgeef/...rchyjoblist.json	1 (25%)	14198 (57.02%)	2151
ID: a-[redacted] www.m-toy.com.tw/...001	1 (25%)	3568 (14.33%)	2109.00
ID: 5-[redacted] www.b...	1 (25%)	3568 (14.33%)	2879
ID: 4-[redacted] tcc.taobao.com/...n/mobile_wl_segment.htm	1 (25%)	3568 (14.33%)	2315.00

Total items: 4

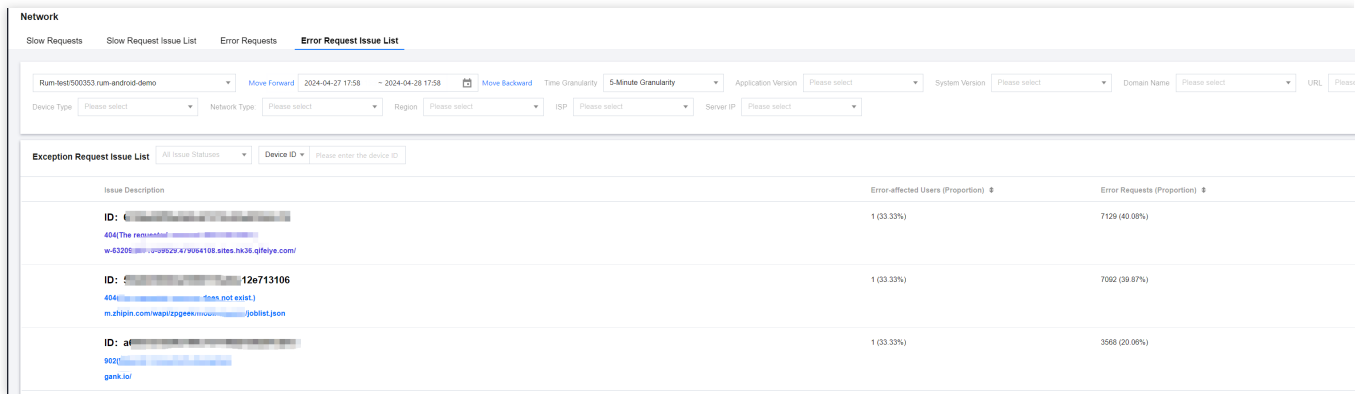
An HTTP request sample is considered a slow request if the transmitted data is over 50 KB and the transfer speed is below 10 KB/s, or if the transmitted data is 50 KB or less and the response time is over 2s. Slow request samples will be shown in the issue list.

The screenshot shows the 'Slow Request Issue Analysis' interface. It includes a 'Sample List' dropdown and a 'Contextual Information' section. The 'Contextual Information' section contains a table with the following columns: User ID, Device ID, URL, Parameter Information, Status Code, Local DNS Server Address, Request Method, Protocol, DNS Query Time, TCP Handshake Time, SSL Duration, TTFB, Response Time, Sent Bytes, Received Bytes, Host IP, System Version, Device Name, Network Type, Client-side source IP, country, ISP, Region, CPU Architecture, Occurrence Time, Root or Not, APM Identifier, Build ID, SDK Version, and Error Information.

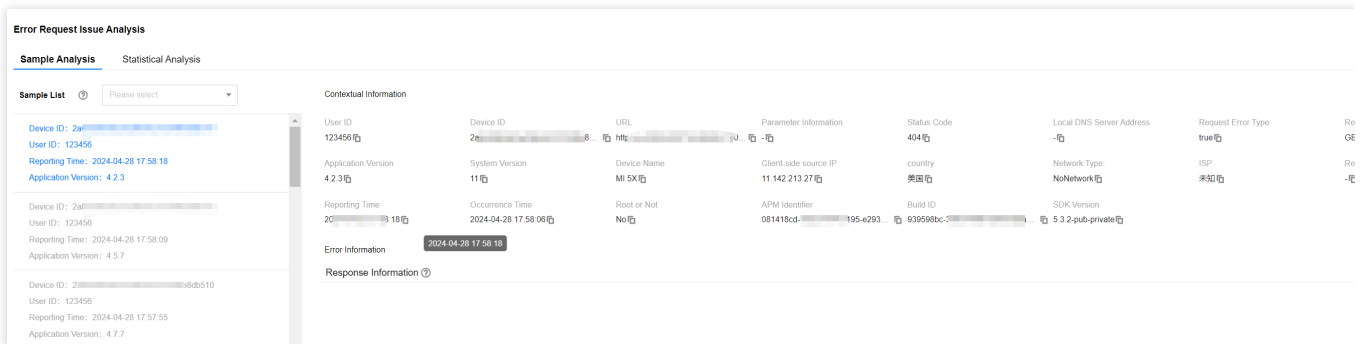
User ID	Device ID	URL	Parameter Information	Status Code	Local DNS Server Address	Request Method	Protocol
123456	[redacted]	https://m.../app/te/...	city=101000000&source=1...	404	-	GET	http
DNS Query Time	TCP Handshake Time	SSL Duration	TTFB	Response Time	Sent Bytes	Received Bytes	Host IP
0ms	0ms	0ms	26ms	187ms	108B	24.50KB	-
System Version	Device Name	Network Type	Client-side source IP	country	ISP	Region	CPU Architecture
6	vivo X9L	WiFi	[redacted]	美国	未知	-	arm64-v8a
Occurrence Time	Root or Not	APM Identifier	Build ID	SDK Version			
2024-04-28 17:58:05	No	53ca35a2-242e-467e-8dd3-cae92...	939566	6398-0a8a... 5.1.0_jmeter			

# Error Request Issue List

Network errors such as HTTP request errors, DNS resolution errors, failure to establish connection, and connection timeout will be displayed in the error request list. You can click the related link under **Issue Description** to view error request details and pinpoint the root cause of error requests.



You can also click **Issue Description** to view error request details and analyze the cause of errors.



# Metrics Description

Related Metrics are as follows:

Metric Name	Metrics Description
Request Duration	App request duration
Slow Request Proportion	The proportion of slow requests to the total number of requests within a selected time range. A request is considered a slow request:

	When the transmitted data is over 50 KB and the transmission speed is below 10 KB/s. When the transmitted data is 50 KB or less and the response time is over 2s.
Slow Requests	The number of slow requests within a selected time range. A request is considered a slow request: When the transmitted data is over 50 KB and the transmission speed is below 10 KB/s. When the transmitted data is 50 KB or less and the response time is over 2s.
Requests	Total application requests
Slow-request Users Proportion	The ratio of the number of users affected by slow requests to the total number of users within a specified time range
Slow Request Users	The number of users affected by slow requests within a specified time range
Request Error Rate	Number of error requests / Total number of requests
Error Requests	Number of network errors in the selected time period Error requests refer to HTTP request errors, DNS resolution errors, inability to establish connection, connection timeout, and other network-related errors
Error-affected User Proportion	Ratio of users affected by error requests to total users within a specified time range
Error-affected Users	Number of users affected by error requests within a specified time range

# Webview

Last updated : 2024-05-13 18:03:25

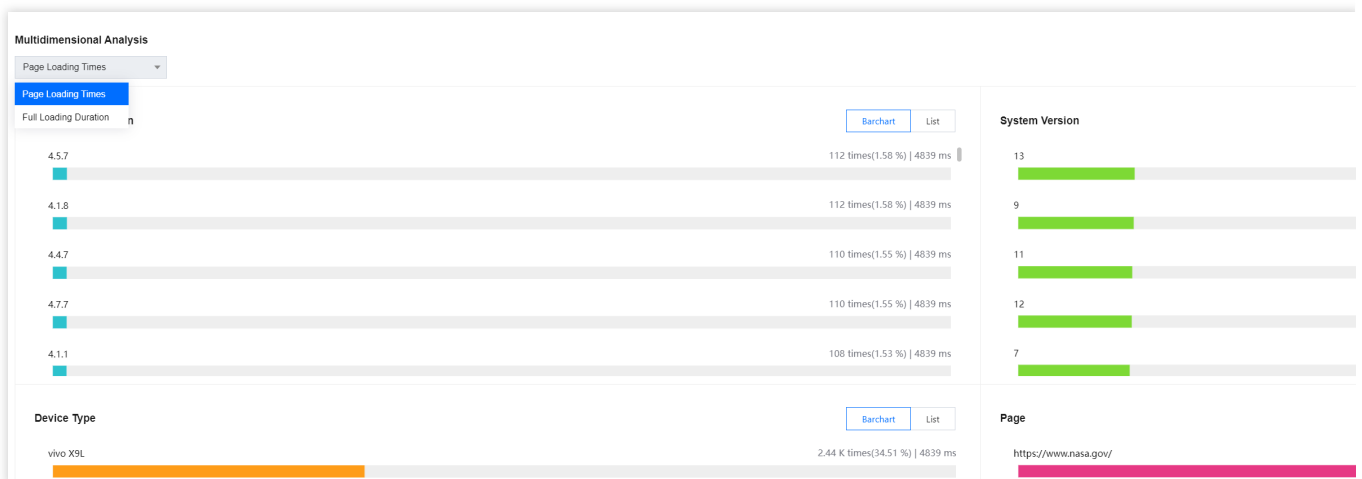
This feature provides WebView metric analysis based on page loading time, slow loading proportion, and JavaScript error rate. It allows you to drill down into WebView and JavaScript errors through the issue list.

## Feature Entry

1. log in to [Mobile App Performance Monitoring Console](#).
2. In the left navigation bar, select **WebView**. Select the business system, app, and time range to analyze WebView issues.

## Slow loading and JavaScript Error Multidimensional Analysis

The multidimensional analysis page shows the analysis of key metrics from multiple dimensions such as app version, system version, device type, page, network type, internet service provider, and region. It facilitates targeted root cause analysis of specific slow loading issues or JavaScript errors.



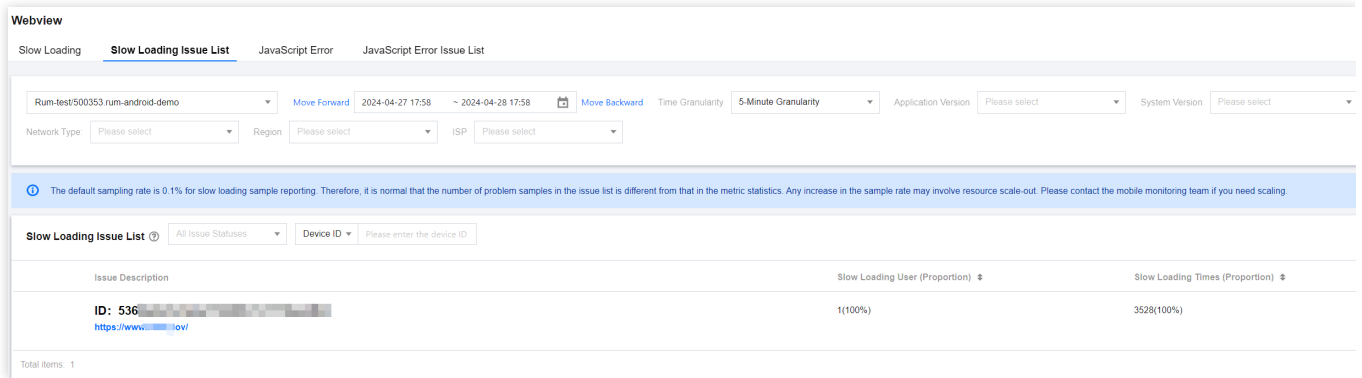
## Slow Loading Issues List

The slow loading issue list shows slow loading issues of all devices. You can quickly filter slow-loading devices by error type and device ID. You can also click **Issue Description** to view the details of slow loading and pinpoint and

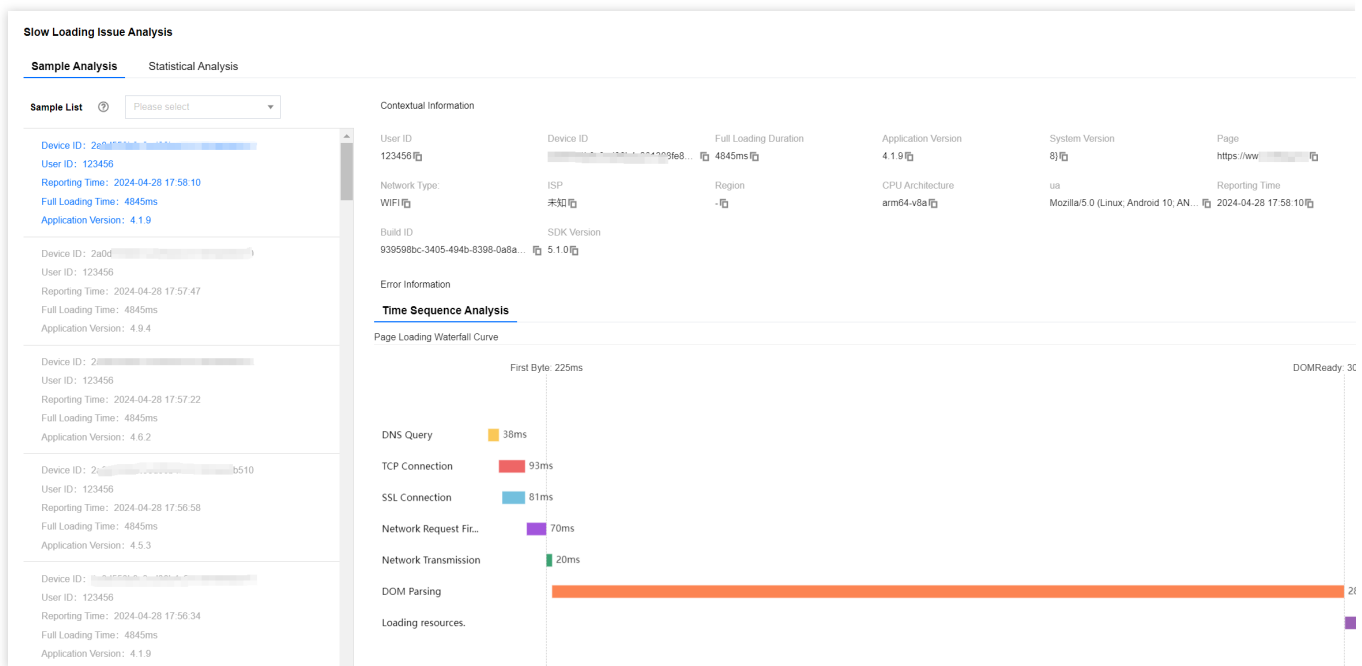
analyze the root causes of slow loading for your app.

**Note:**

The default sampling rate for slow loading sample reporting is 0.1%, so it is normal for the number of issue samples in the issue list to not match the metric statistics.



For each page loading sample, a full loading time greater than 3,500 ms is considered slow loading, and slow loading samples will be displayed in the issue list.

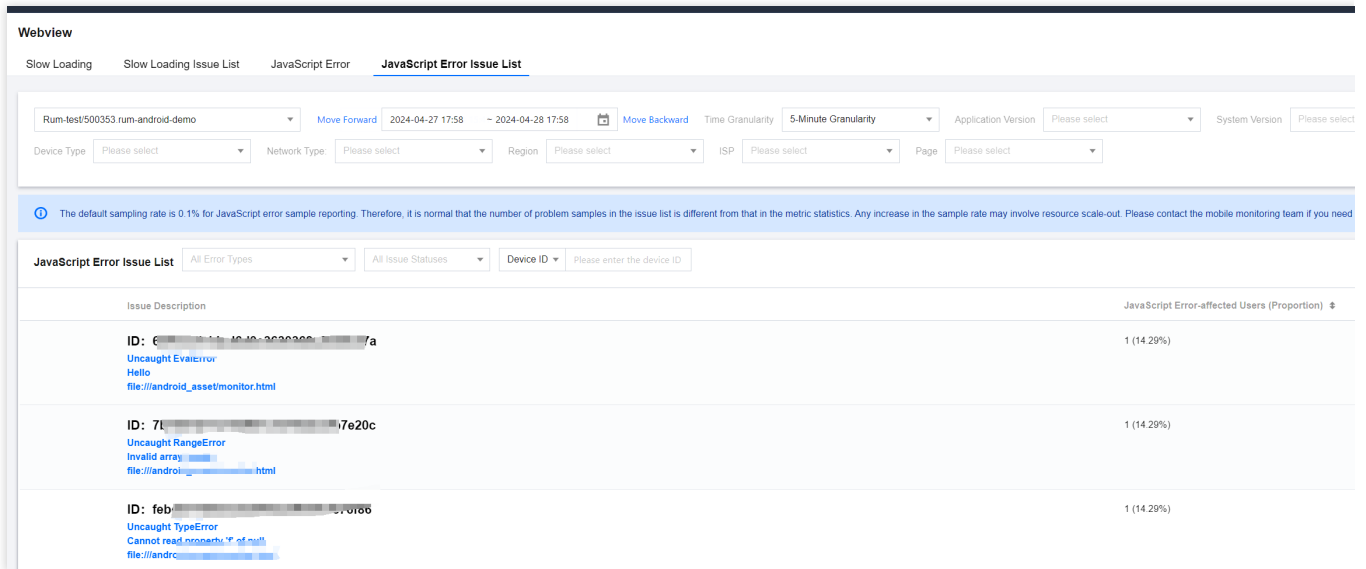


## JavaScript Error Issues List

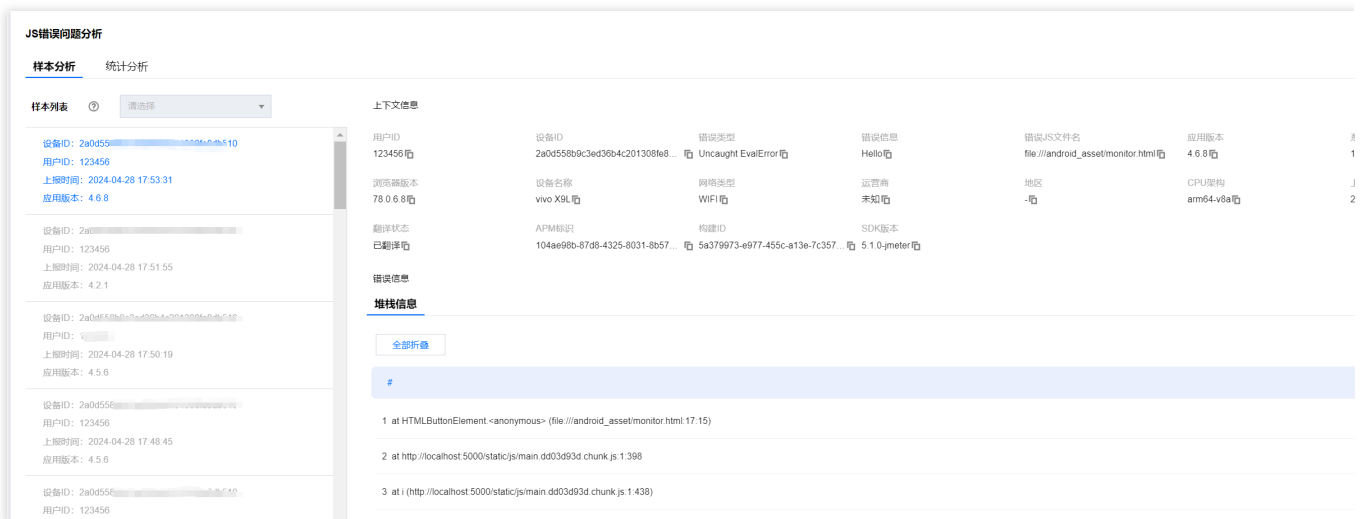
You can view all JavaScript errors in the JavaScript error Issues list.

**Note:**

The default sampling rate for JavaScript error reporting is 0.1%, so it is normal for the number of issue samples in the issue list to not match the metric statistics.



You can also click **Issue Description** to view details of JavaScript errors and pinpoint and analyze the causes of JavaScript errors.



## Metrics Description

Related Metrics are as follows:

--	--

Metric Name	Metrics Description
Page Loading Times	Number of times a page is opened or refreshed
Full Loading Duration	Time taken for the entire web page to be fully loaded
JavaScript Errors	Total number of JavaScript errors within a specified time range
JavaScript Error Rate	Number of users experiencing JavaScript errors/Total number of users accessing the WebView page. Due to computational resource limitations, the numerator and denominator of this metric are not deduplicated.
JavaScript Error-affected User Proportion	Number of users affected by JavaScript errors/Total number of users within a specified time range
JavaScript Error-affected Users	Number of users affected by JavaScript errors within a specified time range

# Application Management

Last updated : 2024-05-13 18:03:25

## Use Cases

On the app management page, you can view your connected end-user apps, new app access, and app IDs, and set an allowlist.

## Directions

### Accessing App

1. Log in to [Tencent Cloud Observability Platform](#).
2. In the left sidebar, select **Mobile App Performance Monitoring > Overview**.
3. click **Application Access**, fill in the app name, set the app type to Android or iOS, set the business system, and click **Next**.

### Application Access

**1** Create Application > **2** Application Access

Application Name (4 to 50 Characters)  ✓

Application Type  ✓

Business System  ✓ [No business system yet? Cli](#)

### Obtaining App ID



On the **Application Management > Application Settings** to enter the app settings page and obtain app IDs from the list.



### Application Management

Business System   **Application Settings**   Allowlist Management

Business System: rum-36uJ0ycDx8qRVY.Rum-test   [Application Access](#)

Application Name	Report ID
rum-android-demo	7015000717156 
rum-ios-demo	3000100000712 

## Allowlist Configuration

Select **Application Management > Allowlist Management** to enter the allowlist configuration page. Click **Add** on the allowlist configuration page. Configure a user ID/device ID allowlist to prevent data reporting from being affected by sampling. That is, users/devices in the allowlist are not affected by the sampling rate and will all be sampled. You can select a user or device type and fill in the relevant ID.

### Add Allowlist Account

Account Type

User ▼

User ID

Please enter the user ID.

Remarks

Please enter remarks for the user allowlist.

Confirm

Cancel