

Table of Contents

Introduction	0
快速入门	1
版本更新	2
Android	2.1
iOS	2.2
使用平台	3
Android	3.1
Gradle 插件安装方法	3.1.1
Eclipse 插件安装方法	3.1.2
Ant 安装方法	3.1.3
Maven 插件安装方法	3.1.4
IntelliJ IDEA eclipse 项目安装方法	3.1.5
2.0.4.*以上版本安装方式	3.1.6
iOS	3.2
OneAPM_iOS_SDK Objective-C 安装文档	3.2.1
OneAPM_iOS_SDK Swift 安装文档	3.2.2
功能说明	4
应用	4.1
交互	4.1.1
ANR	4.1.2
崩溃	4.2
用户信息	4.3
WebView	4.4
网络	4.5
运营	4.6
分析	4.7
设置	4.8

自定义仪表盘	4.9
ANR 数据抓取及展示	4.10
UI性能-卡顿	4.11
用户分析	4.12
Ajax	4.13
告警	4.14
常见问题	5
Android	5.1
SDK 安装后提示错误信息：由于使用 JRE 运行 Eclipse 导致 OneAPM 无法正确加载	5.1.1
Eclipse 安装 SDK 提示后提示找不到 content.xml 文件	5.1.2
追踪用户信息接口使用说明文档	5.1.3
Mac OS下 Android Studio 如何卸载探针？	5.1.4
OneAPM兼容Cordova（PhoneGap）	5.1.5
升级Android Agent 2.0.4后嵌入Token那行代码报错？	5.1.6
Eclipse 安装 SDK 提示“The chosen operation is not currently available”	
集成了 OneAPM 探针后，Crash 功能界面无数据	5.1.7
SDK 安装后提示错误信息：No providers installed	5.1.9
SDK 成功部署后 OneAPM 界面无数据显示	5.1.10
Eclipse 插件安装过程中无法勾选 OneAPM 插件并提示: There are no categorized item	5.1.11
WebView 性能监控使用说明	5.1.12
安装 Android 探针后启动发生异常：finished with non-zero exit value 1解决办法	5.1.13
Eclipse完全卸载说明	5.1.14
iOS	5.2
SDWebImage 造成的 crash 问题	5.2.1
Xcode 版本低于6.3，CPU 占用达到100%	5.2.2
上传dSYM文件步骤	5.2.3
为什么要上传dSYM文件？	5.2.4
崩溃数据统计不到	5.2.5

集成iOS sdk过程中搜索不到libz.dylib和libstdc++.dylib库	5.2.6
SDK成功部署后OneAPM界面无数据显示	5.2.7
若由于网络情况，对数据收集失败，等网络情况恢复后，还能收集到吗？	
不同应用可以使用同一Token值吗？	5.2.8
	5.2.9

Mobile Insight



Mobile Insight 帮助文档. [PDF版](#)

欢迎您使用OneAPM Mobile Insight 「帮助文档」致力于帮助您快速解决困惑，熟悉产品的功能。

新手入门

产品介绍

OneAPM 公司的 Mobile Insight 产品是专注于帮助开发者解决应用上线后性能问题的监控与管理。通过应用内嵌入 Mobile Insight 产品的 SDK，同步真实用户访问体验,及时发现使用过程中的卡顿、崩溃、连接超时、内存泄漏等问题,帮助开发者第一时间终结用户流失。通过实时、多维立体的性能数据展现与自动分析,更能防患于未然,降低 App 上线后维护与迭代成本,直接提升用户留存率。

Mobile Insight 共支持两种平台: **iOS、Android**

版本分类

Mobile Insight分为SaaS免费版、SaaS付费版、企业级。

[相关权限及报价](#)

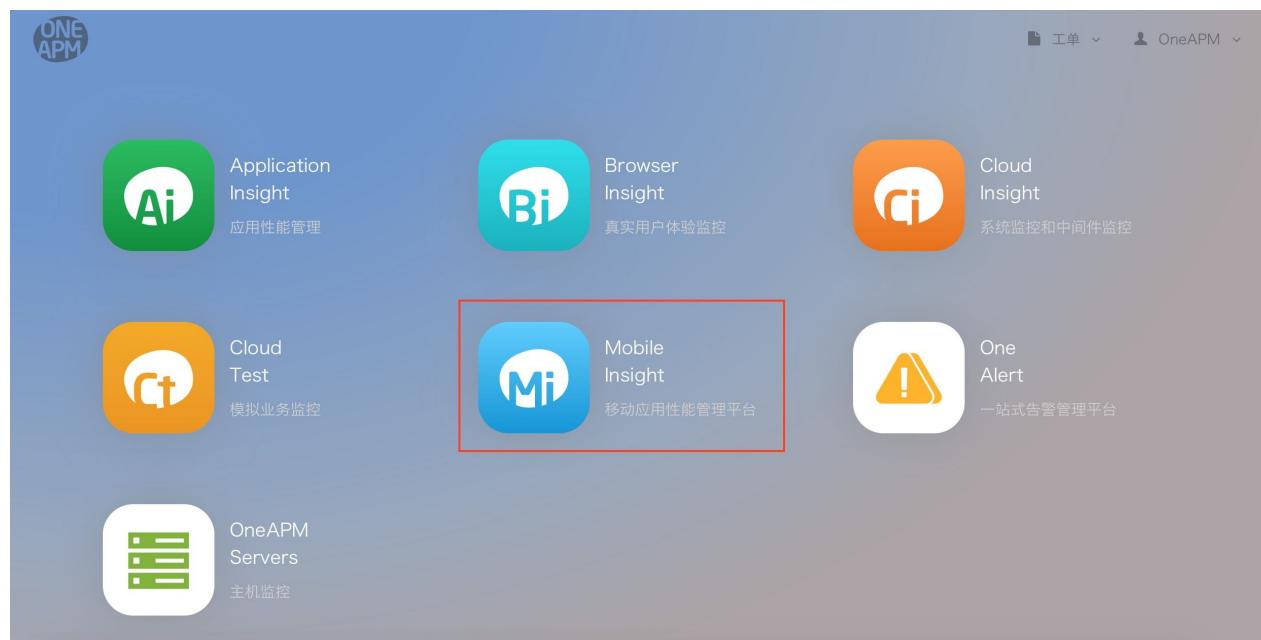
使用流程

1.注册

首先您需要有一个OneAPM的账号: [点击注册](#)

2.选择平台

登陆账号之后选择Mi(Mobile Insight)移动应用性能管理平台，进入应用列表管理界面。



3.添加应用

在应用列表页面，点击<添加应用>按钮，进入下载安装页面。

The screenshot shows the application list page with the following interface elements:

- Header: ONE APM, navigation icons (Ai, Bi, Ci, Ct, Mi, Alert, Servers), and user info.
- Section title: 应用程序 最近一小时信息
- Table: 列出的应用列表（部分信息被遮挡）

应用名称	活跃版本数	包名	崩溃次数	HTTP错误率	网络错误率	活跃设备数
App 1	1	com.app1	0	0.00%	0.00%	0
App 2	1	com.app2	0	0.00%	0.00%	0
App 3	1	com.app3	0	0.00%	0.00%	0
App 4	1	com.app4	0	0.00%	0.00%	1
- Action button: 添 加 应用 (highlighted with a red box)

4.安装集成

点击相应图标，选择您想要集成的Android/iOS SDK，根据步骤将SDK集成到您的项目中。



注意：详尽安装集成步骤，请选择本帮助文档 – 使用平台查看。

5. 查看数据

成功集成之后，运行项目产生数据即可在OneAPM平台上查看应用的各项性能数据。

注意：

- * 给应用取名后获取的Token不能被应用到包名不同的应用程序中
- * iOS崩溃数据必须在Release模式下真机测试并断开与Xcode的连接才能正常搜集到。
- * 安卓WebView数据需要配置之后才能正常使用。
- * 安卓集成中不同的编译器需要选择不同的插件。

版本更新

Android

如何更新最新的 Android SDK？请查看 [Android SDK 更新方法](#)

OneAPM Android SDK 3.0.0

发布日期：2016/05/05

优化点：

- 1.Android agent重构版本
- 2.优化性能
- 3.减小sdk体积
- 4.插件化

OneAPM Android SDK 2.0.4.3 (Beta)

发布日期：2016/04/08

优化功能：

- 1.优化WebView功能；
- 2.优化卡顿功能，探针端增加卡顿开关，默认关闭；
- 3.卡顿开启方法，启动探针处调用withFPSEnable(true)方法； 代码示例：

```
OneApmAgent.init(this).setToken("1270ACC138539652FB6514FE44476CED70").setFP  
SEnable(true).start();
```

OneAPM Android SDK 2.0.8

发布日期：2016/03/17

优化功能：

1. 优化WebView功能。
2. 修改WebView功能集成方法，修改js文件完善WebView功能。
3. 增加Ajax抓取功能。
4. 增加WebView开关功能。

OneAPM Android SDK 2.0.7

发布日期：2016/03/08

优化功能：

1. 优化卡顿功能；
2. 增加探针端卡顿开关功能，默认卡顿功能关闭；
3. 卡顿开启方法，启动探针处调用withFPSEnable(true)方法；

示例代码如下：

```
BlueWare.withApplicationToken("50C964BEB879A561AEDF06951CA2C4E552")
    .withFPSEnable(true)
    .start(this.getApplication());
```

OneAPM Android SDK 2.0.6

发布日期：2016/03/01

优化功能：

1. 优化Android探针性能

OneAPM Android SDK 2.0.5

发布日期：2016/02/26

优化功能：

1. 修复在存在 BaseActivity 情况下，部分上层 Activity 不采集数据的 Bug
2. 修复以 HttpClient 发送网络请求时，当返回错误码 5xx 时，不采集 HttpError 的 bug

3.修复不采集OkHttp中以NewCall方式发送网络请求的bug

4.修复部分网络请求不抓取HttpHeader的bug

5.更正CPU计算时,被除数为0的bug

OneAPM Android SDK 2.0.4.1(Beta)

发布日期：2015/12/28

优化功能：

1. 解决帧率的bug
2. 优化机型的兼容
3. 修复可能的前后台切换黑屏问题
4. 解决渠道号读取bug

OneAPM Android SDK 2.0.3.2

发布日期：2015/12/15

优化功能：

1. 解决ajax在Android5.0上无数据的问题

1. 解决帧率的bug
2. 解决前后台切换的可能的黑屏问题
3. 解决了CPU可能为0的情况
4. 修复jsError点击无法查看详情的情况。

OneAPM Android SDK 2.0.4(Beta)

发布日期：2015/11/26

优化功能：

1. 增加了ANR监控功能
2. 增加了socket监控功能
3. 增加了android的帧率监控功能
4. 优化了运营商获取方式

5. 优化了webview的兼容性

OneAPM Android SDK 2.0.3

发布日期：2015/10/30

优化功能：

1. 增加在操作系统版本升级时重新连接的功能；
2. 优化用户信息查询接口；
3. 增加对 okhttp 2.0.3+ 支持；
4. 优化探针性能。

修复问题：

1. 修复网络时间统计 bug。

OneAPM Android SDK 2.0.2

发布日期：2015/9/21

优化功能：

1. 优化探针对 Cordova 框架，Webview 的支持；
2. 优化探针对网宿 SDK 的支持；
3. 优化设置页面上传 ProGuard 文件功能；
4. 优化日活数据类型，添加 IMEI 信息；
5. 优化用户信息模块，允许用户上传长度为256字节以内的用户信息； 优化网络请求 trace 数据中 HTTP Request/Response Header 信息的采集。默认设置为不采集，付费用户可设置开启该功能。

修复问题：

1. 交互 trace 详情中显示联网信息、地理位置、电量、CPU、Root 信息。

备注：

1. 启用 WebView 功能需要配置对应的 WebView JavaScript 文件，具体操作请参考文档：WebView 性能监控使用说明。

OneAPM Android SDK 2.0.1

发布日期：2015/7/24

增加功能：

1. 移动端网络请求与服务端相关联；
2. WebView 模块支持收集 JavaScript 错误。

修复问题：

1. 修复部分情况下 okhttp 会导致 crash 的问题；
2. 修复特定情况下空指针问题。

OneAPM Android SDK 1.0.8

发布日期：2015/5/26

增加功能：

1. Eclipse 插件更新，支持 Eclipse 4.4 及之后的版本。

修复问题：

1. 本地没有 SSL 的环境下无法发送 crash 信息；
2. 在 crash 没发送成功并且积累比较多的时候启动黑屏的 bug；
3. Reinstall OneAPM 的时候，没有更新到最新版本；
4. Agent 安装过程中，license 信息需要更新；
5. 在手机没有 sim 卡的时候获取手机设备信息会有异常，造成数据收集异常；
6. 在特定情况下无法上传数据；
7. Agent 的信息需要与用户系统保持一致；
8. WebView 自定义 webviewClient 的时候发生 Crash。

OneAPM Android SDK 1.0.7

发布日期：2015/2/14

增加功能：

1. 慢 trace 和 Crash 添加了 running app 功能；
2. Crash 添加了 Crash 操作路径，便于 bug 的复现；
3. 数据传输智能判断是否该使用加密链接；
4. 其他一些细节改进。

OneAPM Android SDK 1.0.6

发布日期：2014/12/22

修复问题：

1. 重构了 Crash 监测功能、添加了记录崩溃前用户的操作轨迹、Crash 修改状态标识、Crash 影响用户等功能；
2. 修复了部分 trace 数据遗漏的问题；
3. 解决了 Eclipse 插件版本号提示不对的问题。

OneAPM Android SDK 1.0.5

发布日期：2014/10/9

修复问题：

1. 解决 trace AsyncTask 的时候 java.lang.ClassCastException 的问题；
2. 可以添加 @SkipTrace 来跳过不需要跟踪的 trace；
3. 解决了没有 versionCode 定义的时候的崩溃问题；
4. 其他小问题的 fix

OneAPM Android SDK 1.0.4

发布日期：2014/8/8

修复问题：

1. 解决 Java u55、Java 8 的崩溃问题；
2. 解决 SDK Instrument Gson 时候的崩溃问题；
3. 下载 OneAPM Android SDK 1.0.4；

OneAPM iOS SDK Release Note

2016-04-06 V2.2.1

特点：

- 与服务器默认通信协议由http改为https

修复问题：

- 与Aspects不兼容的问题；
- 网络监控功能数据收集不全的问题；
- 一些偶发性崩溃问题的修改；

2016-03-16 V2.2.0.9

- 增加JSSError功能，展示错误信息，以及相关堆栈信息；
- 支持监控Ajax请求性能状况；
- 优化交互功能，增加慢交互详情功能；
- 显示慢交互方法调用层级，执行时间，cpu及内存占有率等；
- 优化在个别系统版本下崩溃后数据上报卡顿，优化崩溃轨迹的展示流程；
- 优化启动时间

2016-02-28 V2.1.3

- WebView代码逻辑的优化
- 修复了一些小bug

2016-01-27 V2.1.2

- Bitcode打包改进
- 修正探针导致APP旋转开关为打开状态

2016-01-15 V2.1.1

特点：

- 交互数据（慢交互除外）：
总览中展示View Loading, UIImage, DataBase, Json, NetWork, WebView六种数据的执行时间等信息；
交互列表中展示页面（ViewController）的具体交互数据信息。
- WebView：
请求网络url性能相关信息(暂不支持JS Errors); 慢加载资源耗时信息（目前有link、img、css、script几种类型，暂不支持ajax）。
- 崩溃轨迹：
崩溃发生前的交互行为路径信息，最大支持到19条；

2015-11-17 V2.0.0

特点：完全重构版，重写了90%代码。

- 识别设备类型（iPhone/iPad/iPod、操作系统版本、运营商类型）；
- 统计 http 访问情况、出错情况。
- 修复与百度地图、极光推送等第三方 SDK 不兼容问题；
- 修复当路由被劫持，访问 ip.taobao 连接成功，但返回数据不合法时，数据无法上报的问题；
- 修复首次安装时重试间隔时间错误。
- SDK 自身日志系统改善，统一输出格式，使用[OneAPM printLog:YES] 开启或关闭，默认为关闭状态。

2015-8-27 V1.1.3

- 支持监控通过 NSURLSession 发送的 HTTP 请求；
- 增加用户自定义信息功能；
- 性能优化；
- 解决 Agent 初始化工程中可能发送 crash 的问题。

备注：

新增功能第二条的配置方式如下：

接口信息：

```
+ (void)setCustomInfo:(NSString *)info;  
用法示例：  
[OneAPM setCustomInfo:@"18611421164"];  
[OneAPM startWithApplicationToken:@"225D3C244ACE5E49F1CFA920EF94D8A489"];  
通过该接口设置用户识别信息，该信息会和 crash log 等数据关联。
```

2015-08-04 V1.1.2

- 增加对 SDWebImage 的 HTTP 请求监控；
- 其它优化。
- 解决使用 ASIHttpRequest 时偶发崩溃的问题；
- 解决工程名字为中文时捕获的 crash log 包含乱码的问题；

2015-06-09 V1.1.1

- 修改在一些场景中统计的 http 请求不完善的问题。

2015-06-03 V1.1.0

- iOS SDK Crash 报告；
- Xcode 自动上传符号表错误修正

2015-01-13 V1.0.4

- 增加1.0.3遗漏的 x86-64 支持；
- 修正使用某些社交分享组件时导致崩溃的问题。
- 新增终端移动无线接入网络制式识别；
- 新增 iPhone6/Plus 支持。

2014-08-29 V1.0.3

- 解决了在某些情况下，监测 UI 时产生异常的 bug。
- 增加了控制 SDK log 的接口，log 默认不输出打印。

2014-08-29 V1.0.2

- 解决了某些情况下记录 HTTP 产生异常的 bug。
- 增加对 ARM 64 架构 CPU 的支持；
- 增加对 UI, JSON, CoreData, Image 的测量。

特别说明：

编译时在"Build Settings"中，为"Other Linker Flags"增加"-all_load" 选项。

使用平台

OneAPM for Android

SDK 安装说明

OneAPM Android SDK 会在用户 Java 代码编译期间检测用户用了哪些可能影响用户的 App 性能的接口，并在这些接口执行前记录接口执行的开始时间，在执行后，记录接口的结束时间。

这两个时间的差值就是接口的执行时间，加上其他一些可能的参数一起上报给 Server。Server 根据这些收集到的数据分类的可视化展示，就是你们看见的数据了。

我们的 Agent 启动之后会自己启动一个自己的线程，等于说 SDK 是运行在和用户 App 隔离开的一个沙盒中。可能影响用户 App 性能的安卓系统接口，例如数据库操作中的 insert、update 等；JSON 解析中的 parse 等方法，页面加载中的 onCreate 等方法，图片加载中的 bitmapFactory 中的 decodeFile 方法。希望这个简单的说法能说明白，有任何疑问可以联系 OneAPM 的技术支持。

Android SDK - Gradle 插件安装方法

1. 命名应用程序并获取 App token

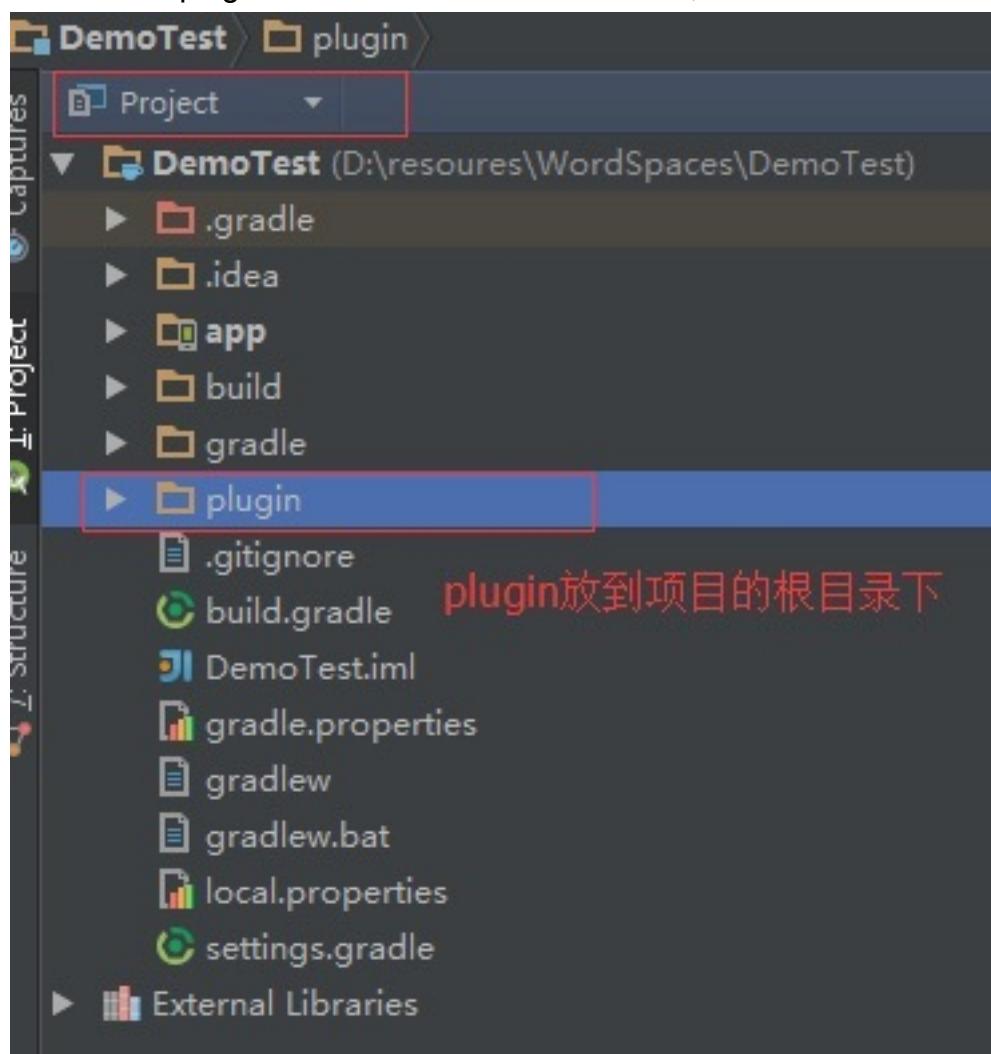
点击进入OneAPM Mobile Insight[集成安装界面](#)，命名应用程序。

2. 下载并解压PneAPM SDK

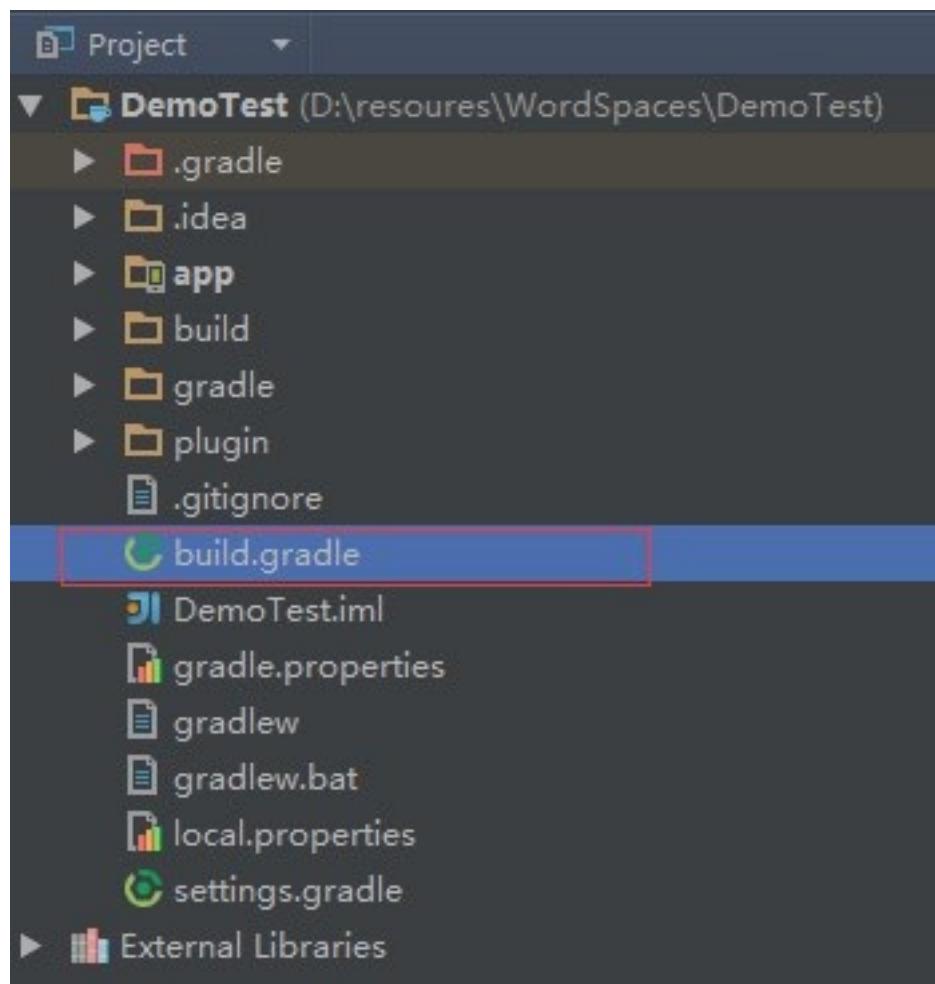
```
** [OneAPM_Android_Agent.zip](https://user.oneapm.com/account/agent/gradle/download.do?versi
```

3. 配置 Gradle

首先我们将 plugin 文件夹整体拷贝到项目根目录, 具体如下图所示



打开工程根目录下的 build.gradle 文件。



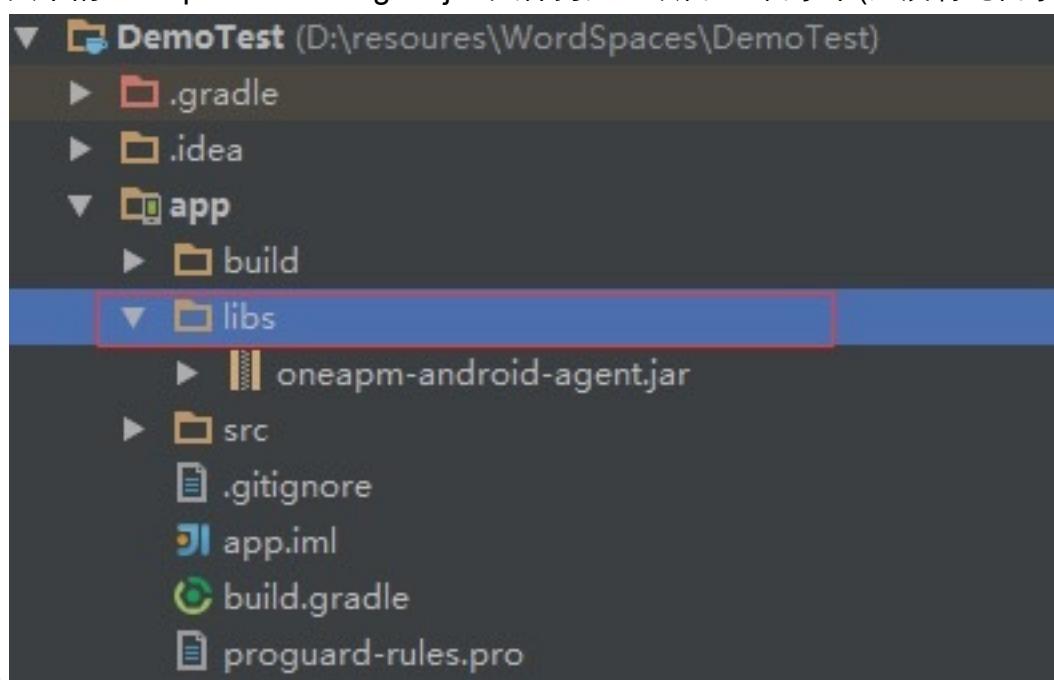
在 **dependencies** 模块中加入代码：

```
classpath fileTree(dir: 'plugin', include: ['*.jar'])
```

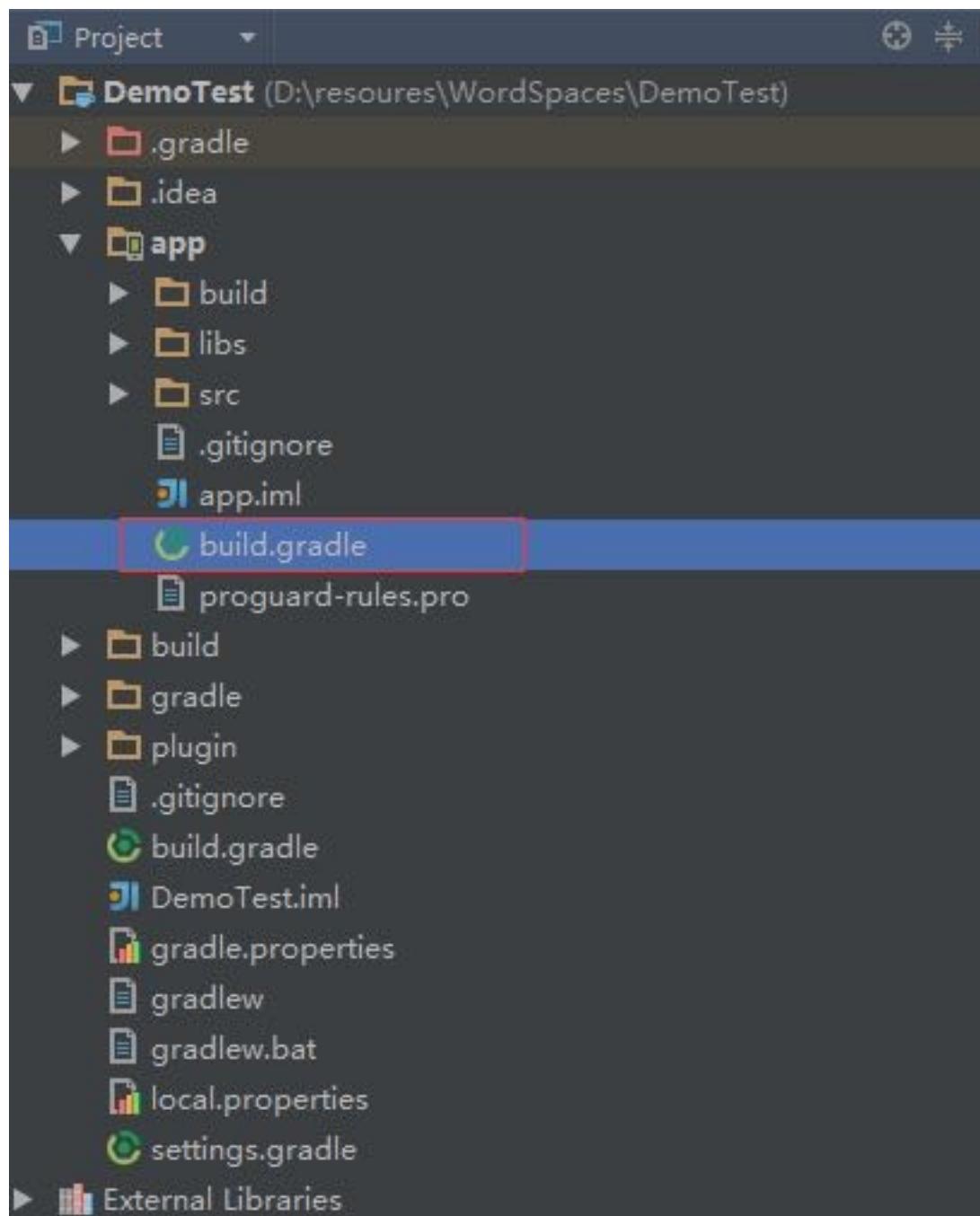
```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.2.2'
        classpath fileTree(dir: 'plugin', include: ['*.jar'])
    }
}
```

引入 OneAPM

将agent文件夹下的 oneapm-android-agent.jar 文件拷贝至项目libs目录下(如没有此目录
请自行创建)



打开主模块目录下的 build.gradle 文件。



在文件头部引入 OneAPM

```
apply plugin: 'oneapm'
```

```

apply plugin: 'com.android.application'
apply plugin: 'oneapm'

android {
    compileSdkVersion 22
    buildToolsVersion "22.0.1"

    defaultConfig {
        applicationId "com.oneapm.demotest"
        minSdkVersion 15
        targetSdkVersion 22
    }
}

```

如此即完成了OneAPM的引入。

建议 rebuild & clean 项目，来确保 OneAPM 配置生效。

如果 *dependencies* 没有如下的配置

```
```compile fileTree(include: ['*.jar'], dir: 'libs')```
```

请在 *dependencies* 中加入如下配置

```
```compile files('libs/oneapm-android-agent.jar')```
```

4. 配置授权信息

确保应用程序的 *AndroidManifest.xml* 配置文件中，引入了以下授权：

```

<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.INTERNET" />
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!--sdk读取设备识别码需要该权限-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--【非必选】若想知道 Crash 的时候，后台有哪些任务运行，请引入该权限-->
<uses-permission android:name="android.permission.GET_TASKS" />

```

注意：如果您的应用使用 *proguard* 混淆，请配置以下：

```
-keep class org.apache.http.impl.client.**
-dontwarn org.apache.commons.**
-keep class com.blueware.** { *; }
-dontwarn com.blueware.**
-keep class com.oneapm.** { *; }
-dontwarn com.oneapm.**
-keepattributes Exceptions, Signature, InnerClasses
```

注意：如果您希望保留行号信息，建议您在 *proguard.cfg* 中添加如下代码：

```
-keepattributes SourceFile, LineNumberTable
```

5. WebView性能监控（可选）

如果你需要开启此功能，请参考 [WebView 性能监控使用说明](#)。

6. 用户信息配置（可选）

顾名思义，就是说和每一个用户相关联的数据信息。例如崩溃的时候可以根据这个配置查询是哪一个用户发生了崩溃。如下：

```
// 附加数据
HashMap<String, String> extraData = new HashMap<String, String>();
String userTel = "15801388723";
extraData.put("tel", userTel);
extraData.put("userId", "888");
extraData.put("email", "88888@qq.com");

ContextConfig config = new ContextConfig();
String searchValue = userTel;
config.setSearchValue(searchValue); // 设置一个搜索值
config.setExtra(extraData);

OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

7. 集成统计分析功能（可选）

在每个 Activity 中导入 OneApmAnalysis 类

```
import com.oneapm.agent.android.module.analysis.AnalysisModule;
```

在每个 Activity 的 onResume() 方法中添加代码:

```
AnalysisModule.onResume();
```

如下示例代码:

```
@Override  
protected void onResume() {  
    super.onResume();  
    AnalysisModule.onResume();  
}
```

在每个 Activity 的 onPause() 方法中添加代码:

```
AnalysisModule.onPause();
```

如下示例代码:

```
@Override  
protected void onPause() {  
    super.onPause();  
    AnalysisModule.onPause();  
}
```

配置渠道信息: 如果您的app需要增加渠道信息请在AndroidManifest.xml的Application标签内添加如下 (请把YOU CHANNEL替换成您自己的发布渠道例如豌豆荚、360等)

```
<meta-data android:name ="BluewareChannel" android:value="YOUR CHANNEL" />
```

注意: 如果两个Activity是继承关系, 只需要在父Activity添加即可, 如果在两个Activity中同时添加, 则会造成重复统计。

8. 功能开关 (可选)

如果您想使用帧率监控功能可以配置如下代码开启帧监控功能

```
PerformanceConfiguration.getInstance().setEnableFps(true);
```

```
AgentHealthConfiguration.getInstance().setHost("www.oneapm.com:9080").setUseSSL(false);
```

```
PerformanceConfiguration.getInstance().setCrashHost("www.oneapm.com:9080").setEnabledCrash(true).setHost("www.oneapm.com:9080").setUseSSL(false);
```

9. 启动Agent

在默认启动的 Activity 中 import OneApmAgent类

```
import com.oneapm.agent.android.OneApmAgent;
```

在App的第一个Activity的 onCreate() 方法中加入如下调用代码来初始化 OneAPM (其中包含了在步骤 2 中根据应用程序名称而生成的授权编号)

```
OneApmAgent.init(this.getApplicationContext()).setToken("---<YOU TOKEN HERE>---").start();
```

或如果设置了用户信息 (第5步) , 初始化代码如下 (config是类似第5步中的用户信息配置)

```
OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

10. 验证是否成功集成探针

在Logcat中过滤oneapm标签, 查看是否有类似如下的日志输出即可(VERSION代表发布版本, 因版本不同而不同)。

```
OneAPM started with version :{VERSION}.
```

11. 静候 5 分钟, 开启 OneAPM 之旅

静候 5 分钟, 等待应用程序向 OneAPM 发送应用程序性能数据, 即可开始使用 OneAPM 应用性能管理。

若应用程序无数据展现，或安装过程中有任何问题：

您可以采取以下方式与我们取得联系：

技术支持热线：400-622-3101

OneAPM 客服邮箱：support@oneapm.com OneAPM MI技术交流3群：471152223

Eclipse 插件安装方法

1. 命名你的应用程序

点击进入OneAPM Mobile Insight[集成安装界面](#)，命名应用程序。

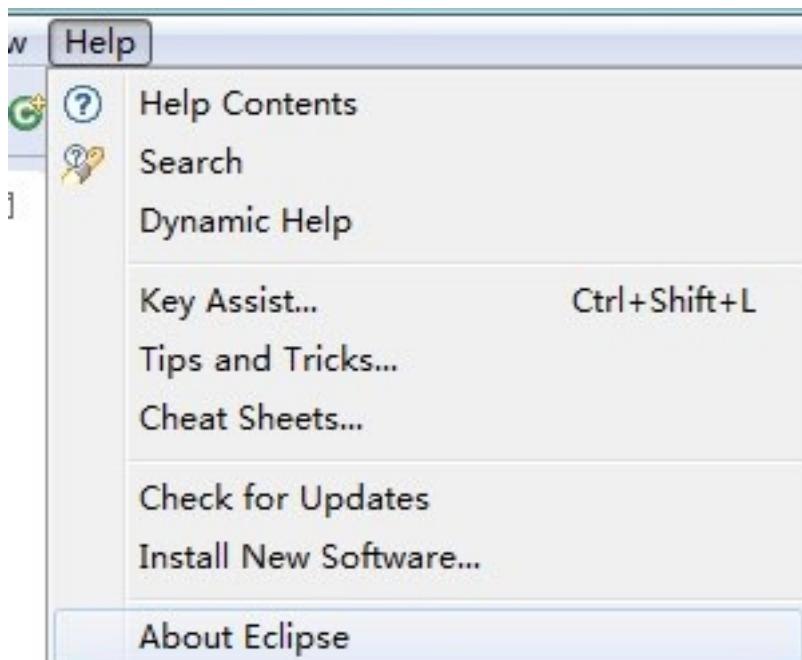
OneAPM Android SDK支持在 Eclipse 集成开发环境中直接使用和部署。

2. 查看 Eclipse 版本号

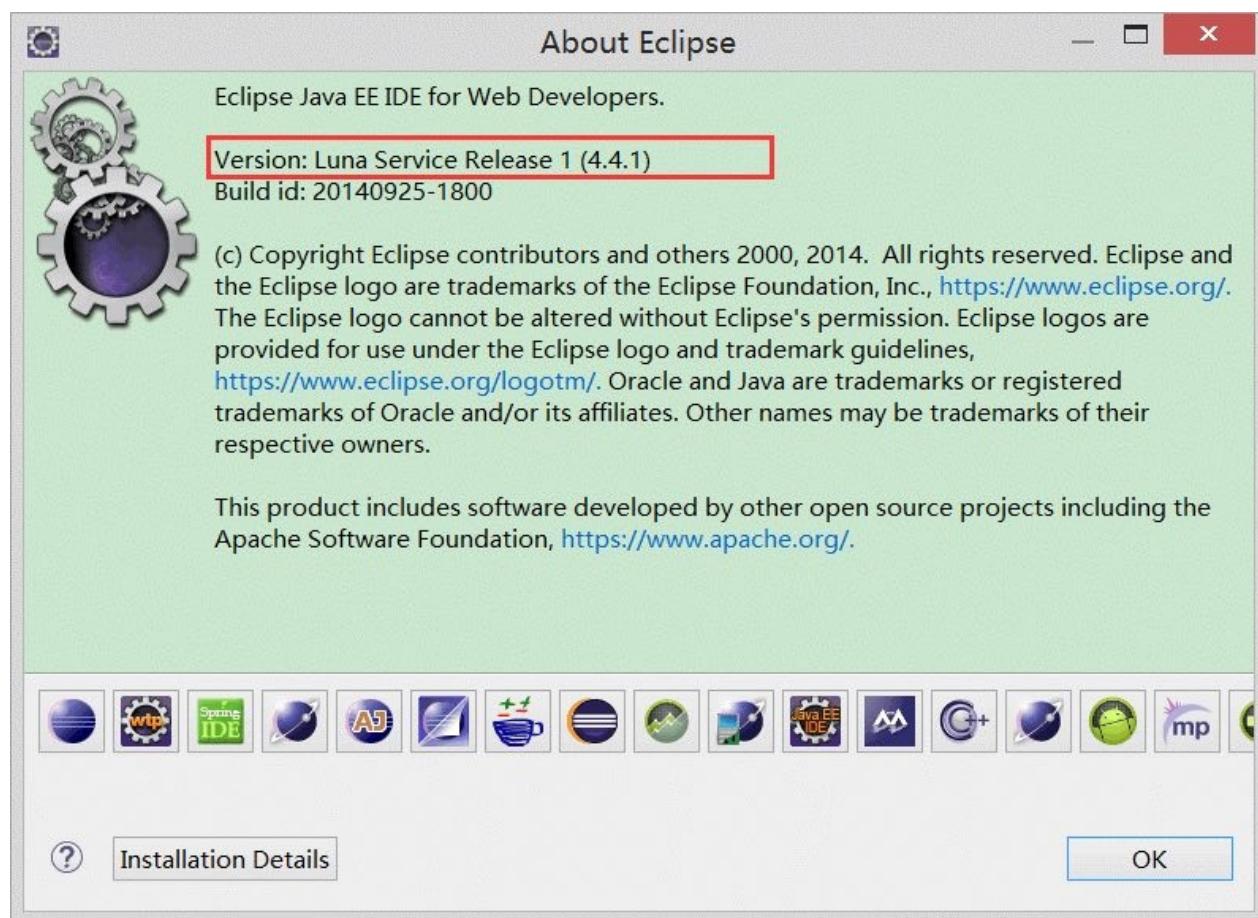
OneAPM Eclipse 插件支持 Eclipse 3.8 及以上版本，请于安装前确认您使用的 Eclipse 版本号：

Mac OS 下 点击进入“关于 Eclipse” Windows 下 点击进入“About Eclipse”或者是“About ADT”

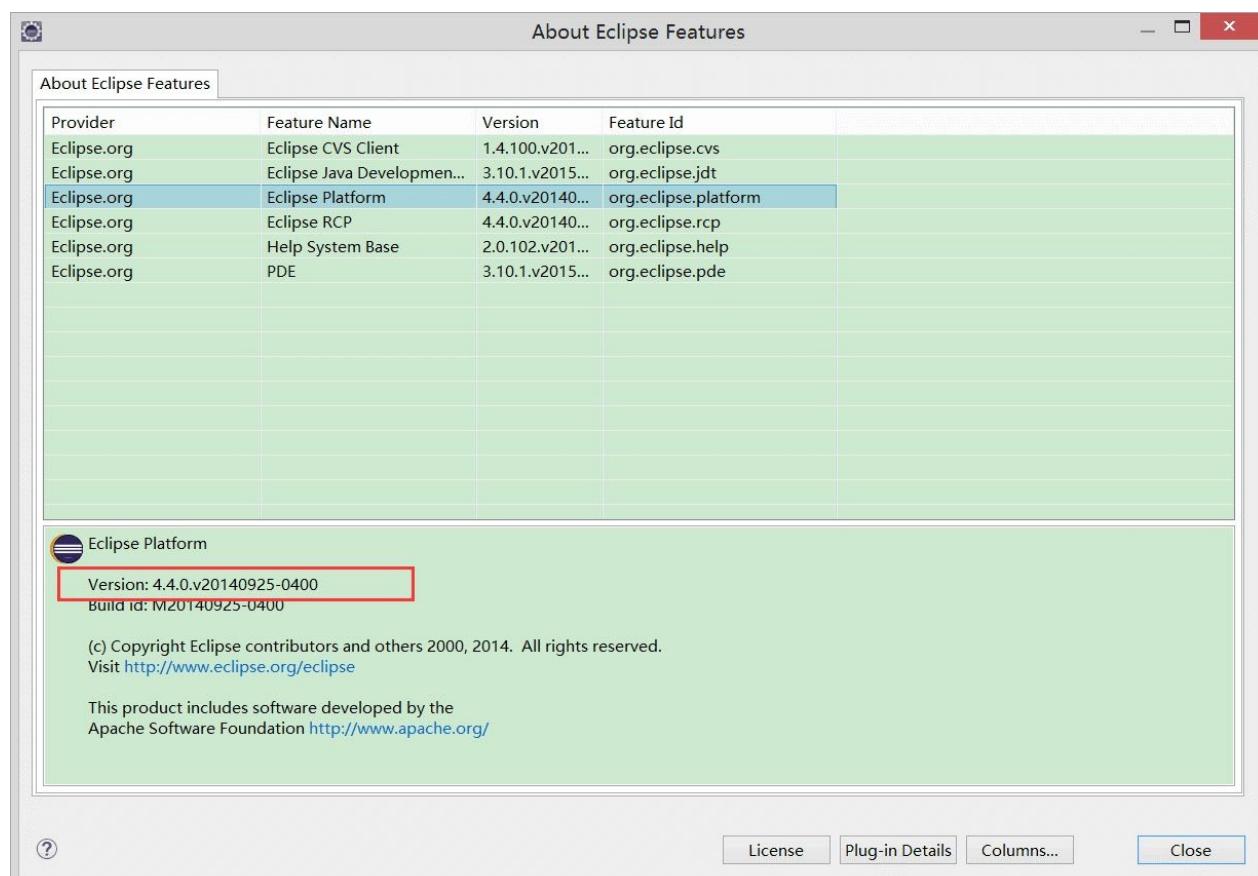
(1)点击Help, 打开“关于 Eclipse”可查看当前版本号



(2)如下图所示, 红色方框内的版本号信息.(如未见详细版本号,可点击进入“Eclipse Plugin”查看,详见步骤(3)。)

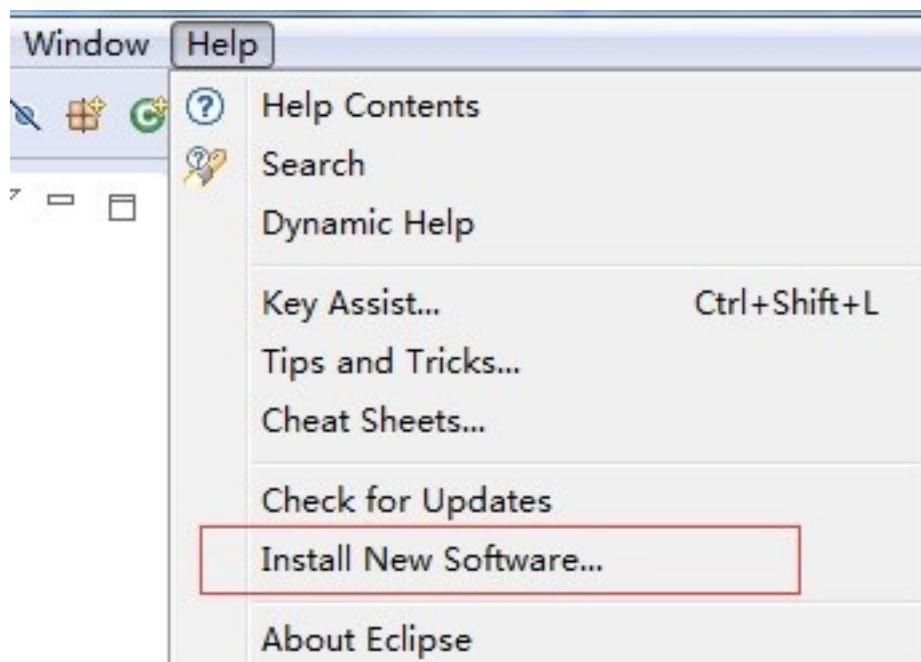


(3)如果您的eclipse版本号在上图中没有出现,可以点击上图中的底部Eclipse图标,点击后弹出对话框,如下图所示:



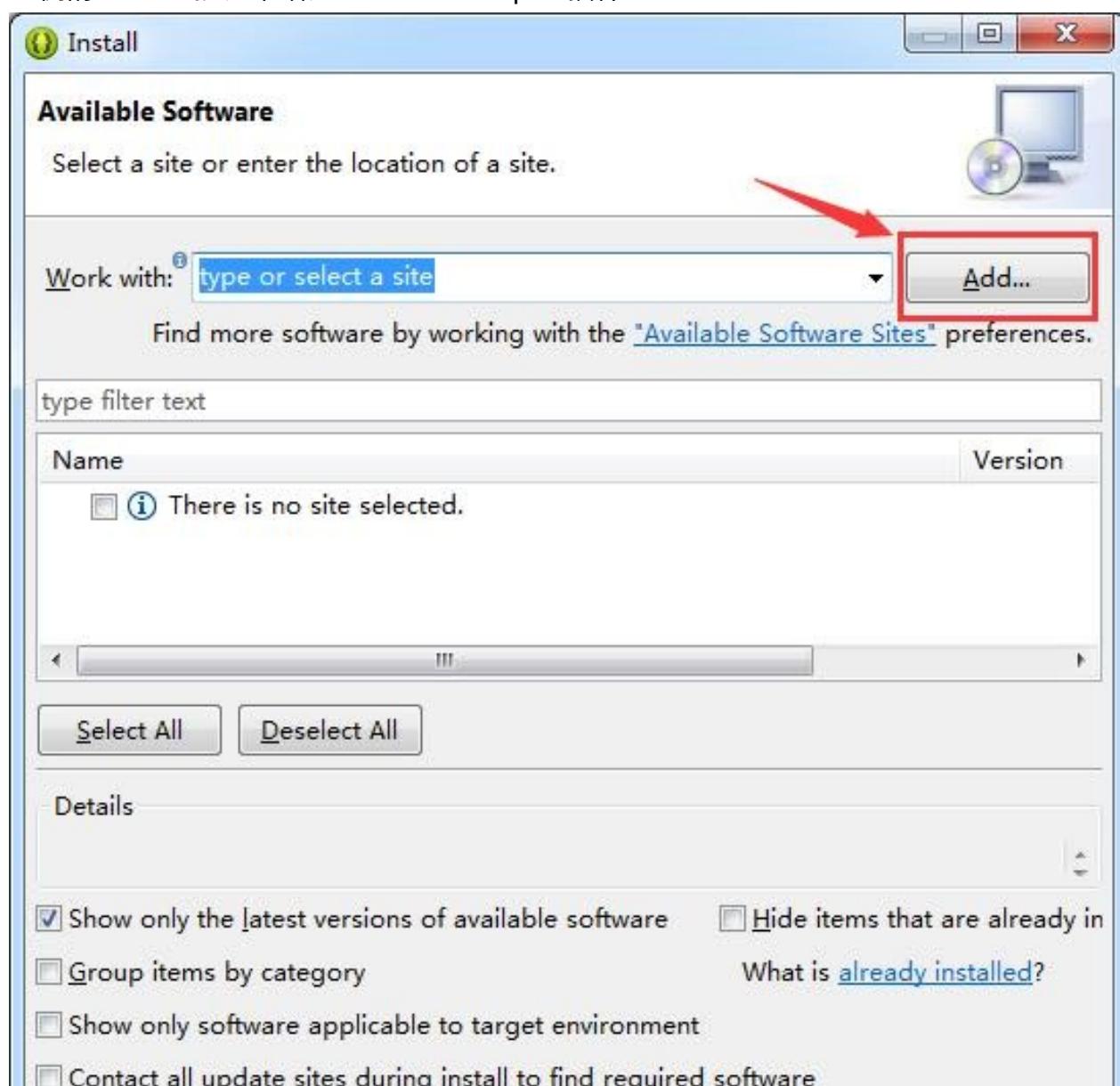
3. 安装 OneAPM Eclipse 插件

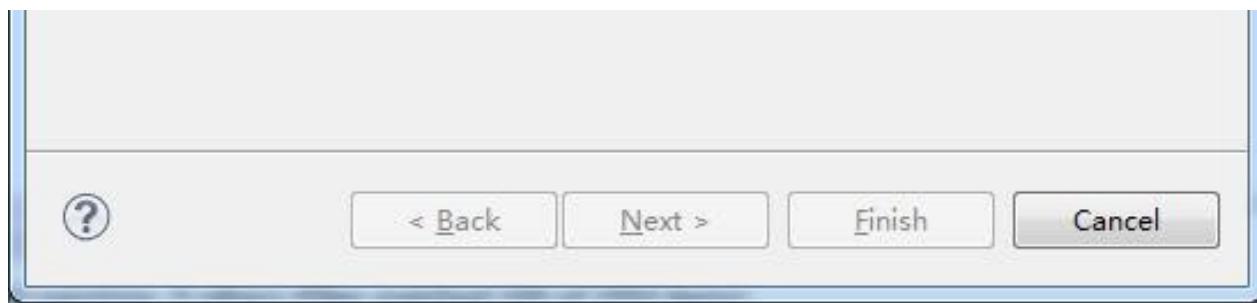
(1) 在 Eclipse 集成开发环境中点击“Help”菜单，选择“Install New Software...”



(2) 点击Work with项

右侧的“Add...”按钮来增加 OneAPM Eclipse 插件.





(3) 设置插件的名称（比如OneAPM）以及URL地址: Eclipse 插件需要 JAVA_HOME 环境变量，目前已支持最新 4.4 版本 Eclipse。Eclipse 4.4 及之后版本请使用以下链接：

```
...  
https://download.oneapm.com/android\_agent/eclipse\_gt\_4.4/  
...
```

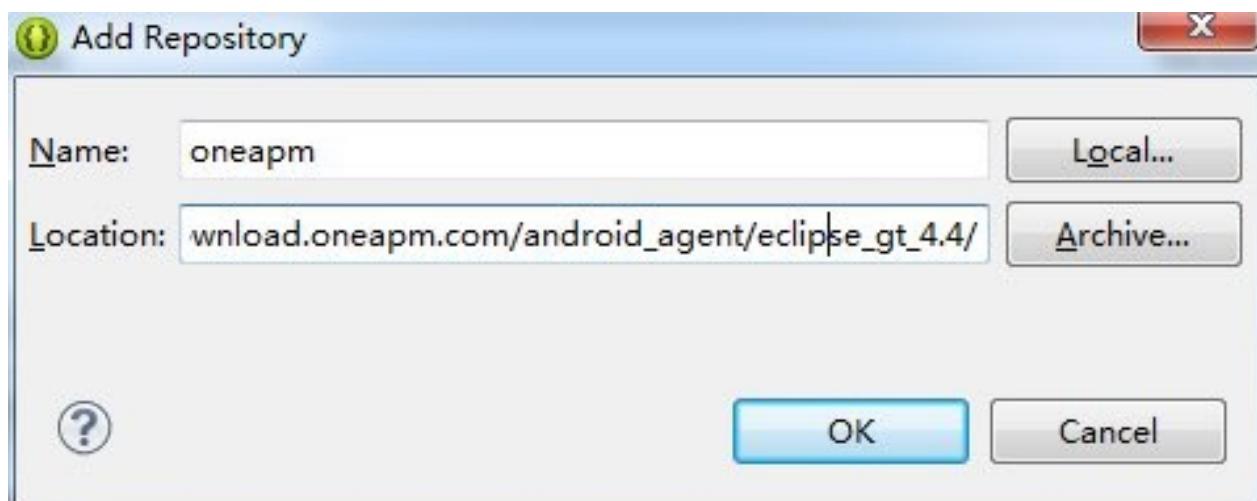
注：OneAPM Eclipse 4.4 插件需要 JDK 1.8。

Eclipse 4.4 之前版本请使用以下链接：

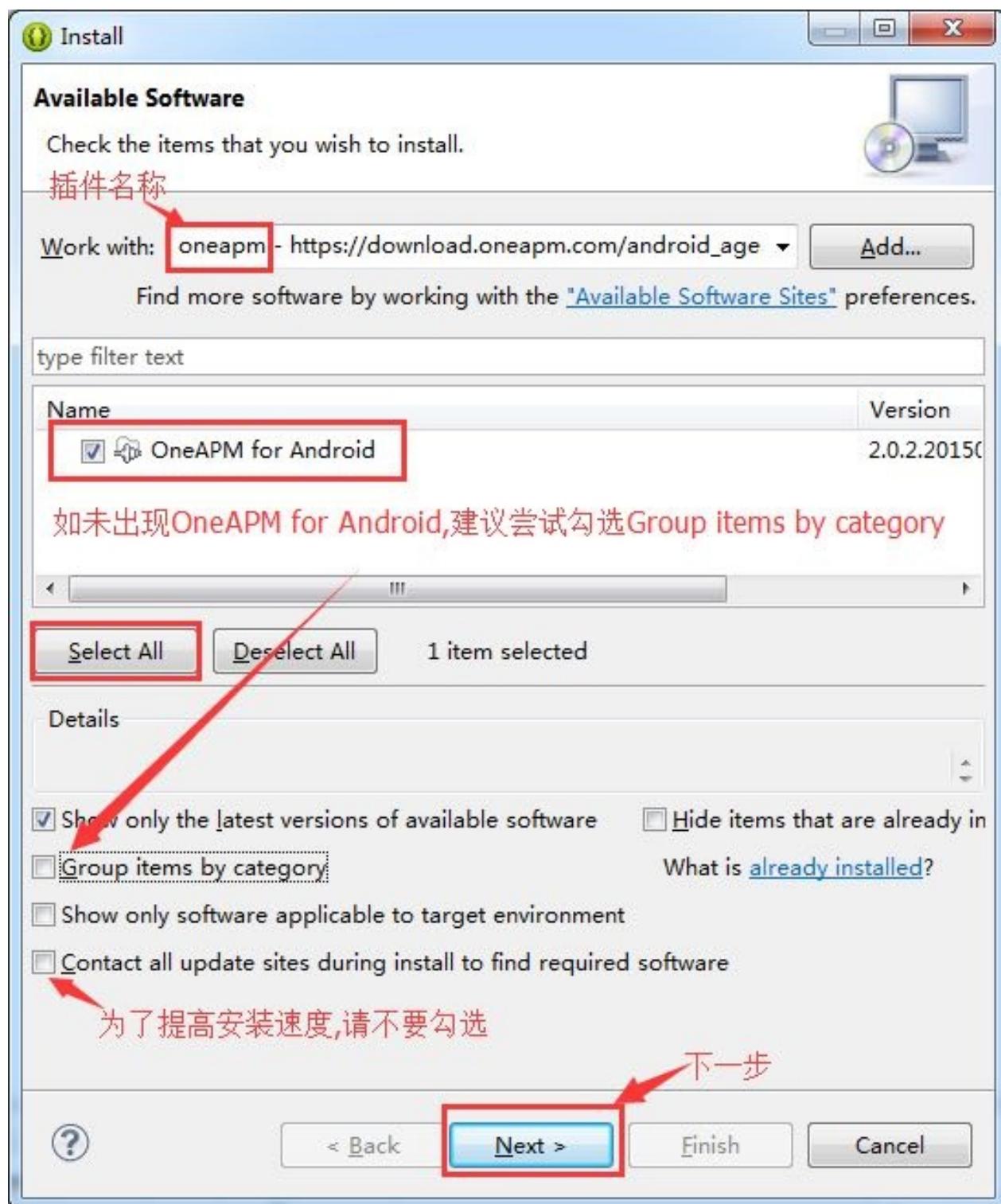
```
...  
https://download.oneapm.com/android\_agent/eclipse\_lt\_4.4/  
...
```

例如, Eclipse 4.4 版本就可使用

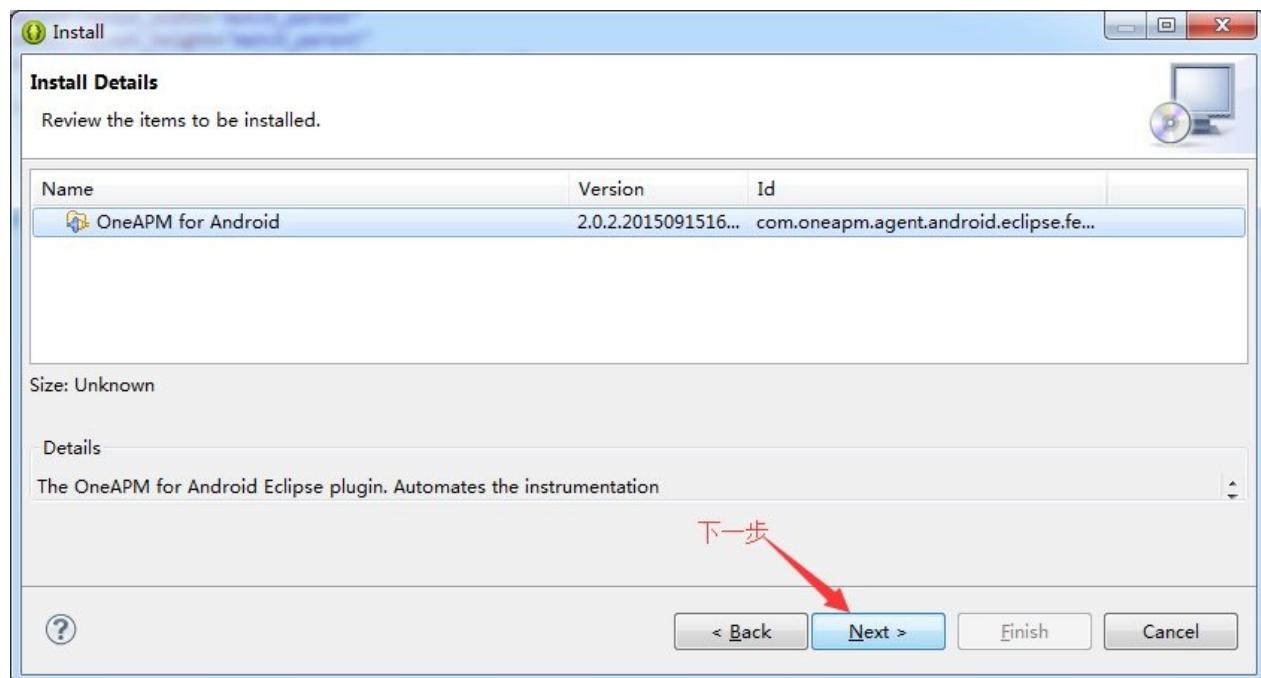
```
...  
https://download.oneapm.com/android\_agent/eclipse\_gt\_4.4/  
...
```



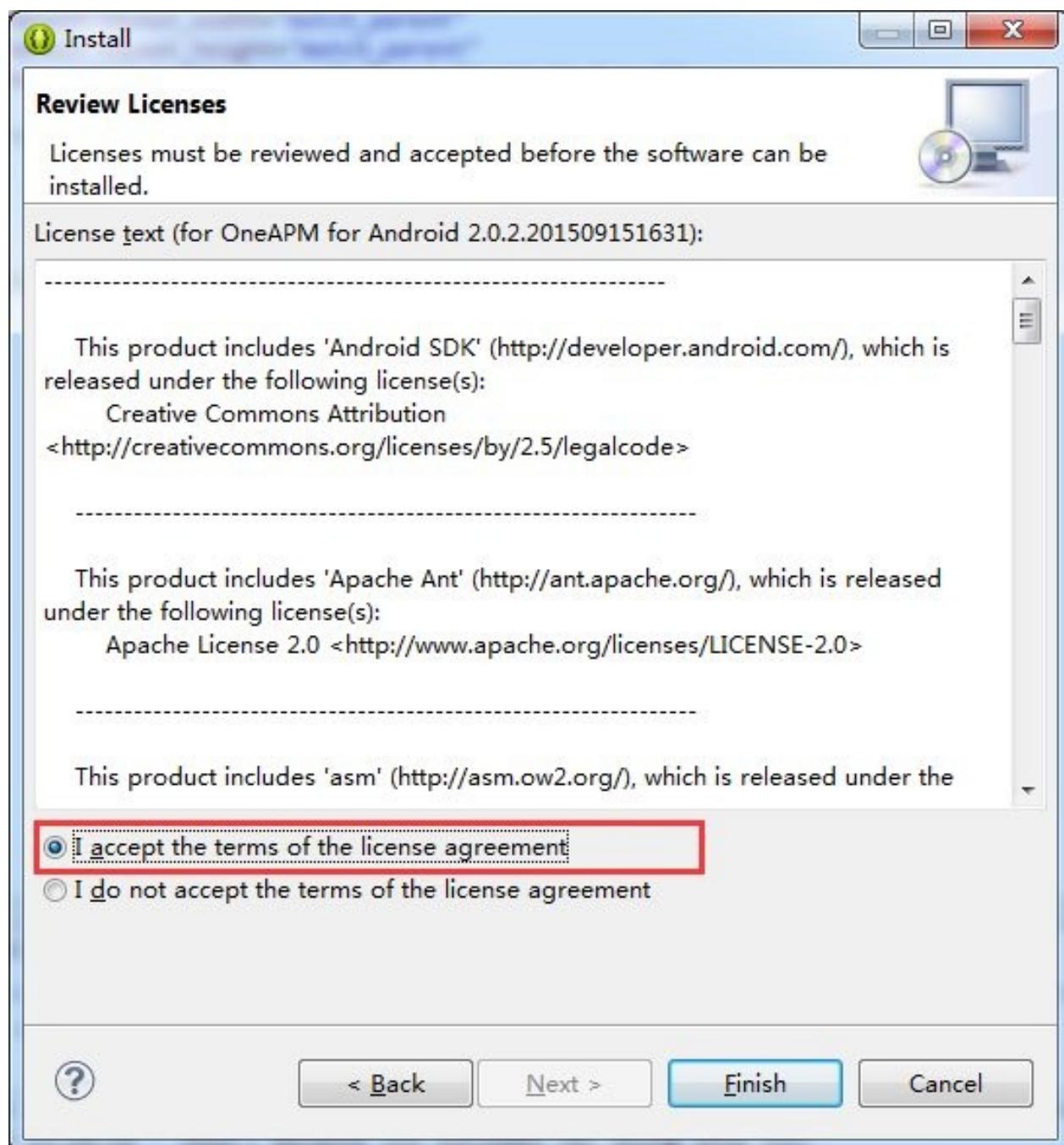
(4) 在 Work with 选项中选中刚才添加的资料库，在下方列表中点击“Select All”选中所有的插件。点击“Next >”到下一步。



(5) 查看插件描述并点击“Next >”进入下一步。



(6) 查看许可协议，选择“*I accept the terms of the license agreement*”，点击“*Finish>*”进入下一步。



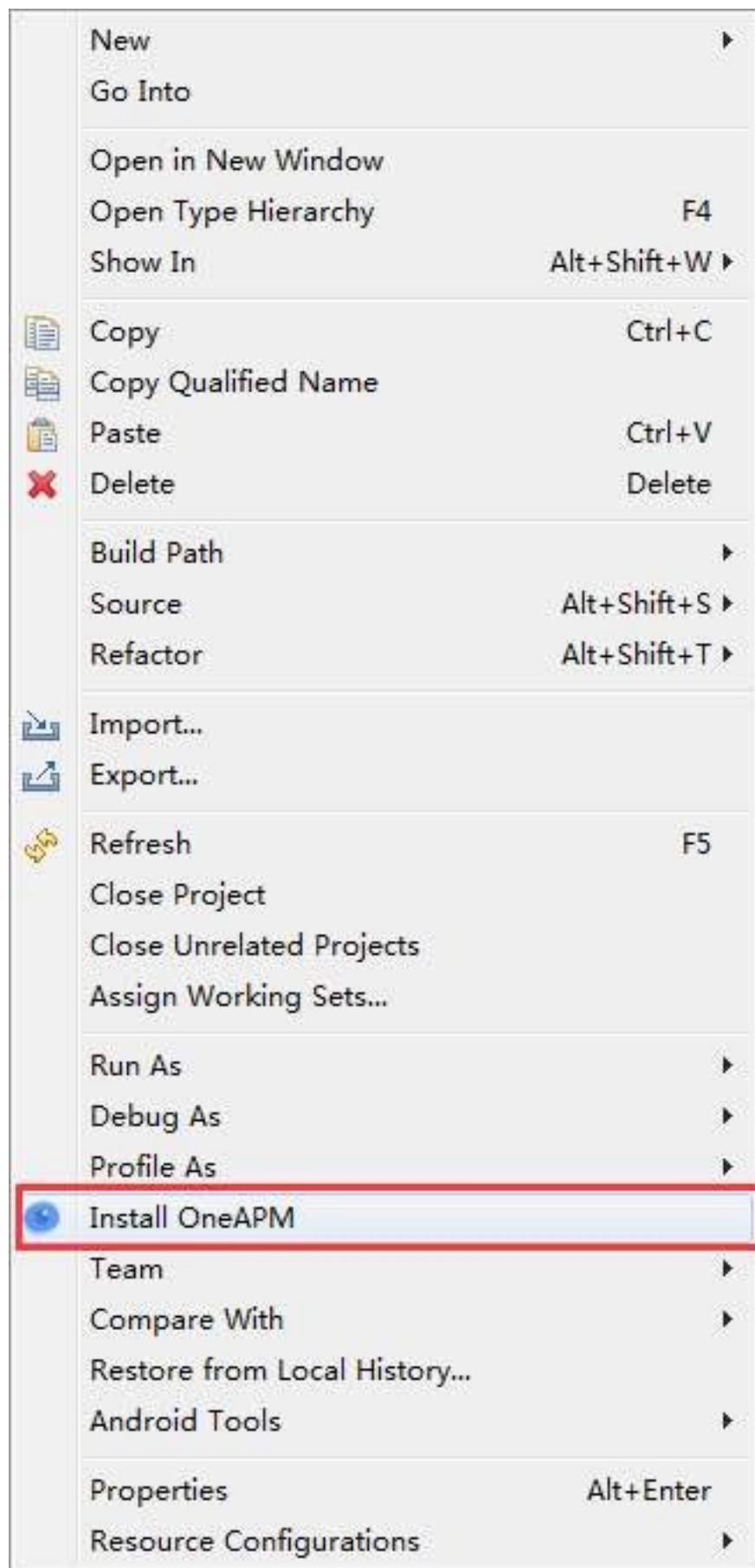
(7)选择信任插件的签名证书，点击“OK”。



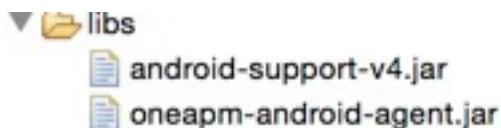
(8)点击“OK”重启 Eclipse 来完成插件的安装。



(9)插件安装完成后，右击需要监控的App，选择“安装OneAPM”



Eclipse 会自动添加“oneapm-android-agent.jar”包到libs目录下，若没有libs目录请新建一个。



注：Eclipse 插件目前只支持JDK（1.5 - 1.8）运行环境（不支持只有JRE的运行环境）
 Window 安装Eclipse插件时，请在没有空格和特殊字符的路径安装JDK Eclipse 插件需要使用JAVA_HOME环境变量，请检查环境变量,如果提示SDK 安装后提示错误信息：由于使用 JRE 运行 Eclipse 导致 OneAPM 无法正确加载，请参考链接：

```
https://oneapm.kf5.com/posts/view/48050/
```

4. 配置授权信息

确保应用程序的 AndroidManifest.xml 配置文件中，引入了以下授权：

```
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.INTERNET" />
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!--sdk读取设备识别码需要该权限-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--【非必选】若想知道 Crash 的时候，后台有哪些任务运行，请引入该权限-->
<uses-permission android:name="android.permission.GET_TASKS" />
```

注意：如果您的应用使用 proguard混淆，请配置以下：

```
-keep class org.apache.http.impl.client.**
-dontwarn org.apache.commons.**
-keep class com.blueware.** { *; }
-dontwarn com.blueware.**
-keep class com.oneapm.** { *; }
-dontwarn com.oneapm.**
-keepattributes Exceptions, Signature, InnerClasses
```

注意：如果您希望保留行号信息，建议您在 proguard.cfg 中添加如下代码：

```
-keepattributes SourceFile, LineNumberTable
```

5. WebView性能监控（可选）

如果你需要开启此功能，请参考 [Webview性能监控说明](#)。

6. 用户信息配置（可选）

顾名思义，就是说和每一个用户相关联的数据信息。例如崩溃的时候可以根据这个配置查询是哪一个用户发生了崩溃。如下：

```
// 附加数据
HashMap<String, String> extraData = new HashMap<String, String>();
String userTel = "15801388723";
extraData.put("tel", userTel);
extraData.put("userId", "888");
extraData.put("email", "88888@qq.com");

ContextConfig config = new ContextConfig();
String searchValue = userTel;
config.setSearchValue(searchValue); // 设置一个搜索值
config.setExtra(extraData);

OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

7. 集成统计分析功能（可选）

在每个 Activity 中导入 OneApmAnalysis 类

```
import com.oneapm.agent.android.module.analysis.AnalysisModule;
```

在每个 Activity 的 onResume() 方法中添加代码：

```
AnalysisModule.onResume();
```

如下示例代码：

```
@Override
protected void onResume() {
    super.onResume();
    AnalysisModule.onResume();
}
```

在每个 Activity 的 onPause() 方法中添加代码:

```
AnalysisModule.onPause();
```

如下示例代码:

```
@Override  
protected void onPause() {  
    super.onPause();  
    AnalysisModule.onPause();  
}
```

配置渠道信息: 如果您的app需要增加渠道信息请在AndroidManifest.xml的Application标签内添加如下 (请把YOU CHANNEL替换成您自己的发布渠道例如豌豆荚、360等)

```
<meta-data android:name ="BluewareChannel" android:value="YOUR CHANNEL" />
```

注意: 如果两个Activity是继承关系, 只需要在父Activity添加即可, 如果在两个Activity中同时添加, 则会造成重复统计。

8. 功能开关 (可选)

如果您想使用帧率监控功能可以配置如下代码开启帧监控功能

```
PerformanceConfiguration.getInstance().setEnableFps(true);
```

9. 启动Agent

在默认启动的 Activity 中 import OneApmAgent类

```
import com.oneapm.agent.android.OneApmAgent;
```

在App的第一个Activity的 onCreate() 方法中加入如下调用代码来初始化 OneAPM (其中包含了在步骤 2 中根据应用程序名称而生成的授权编号)

```
OneApmAgent.init(this.getApplicationContext()).setToken("---<YOU TOKEN HERE>---").start();
```

或如果设置了用户信息（第5步），初始化代码如下（config是类似第5步中的用户信息配置）

```
OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

10. 验证是否成功集成探针

在Logcat中过滤oneapm标签，查看是否有类似如下的日志输出即可(VERSION代表发布版本，因版本不同而不同)。

```
OneAPM started with version :{VERSION}.
```

11. 静候 5 分钟，开启 OneAPM 之旅

静候 5 分钟，等待应用程序向 OneAPM 发送应用程序性能数据，即可开始使用 OneAPM 应用性能管理。

若应用程序无数据展现，或安装过程中有任何问题：

您可以采取以下方式与我们取得联系：

技术支持热线：400-622-3101

OneAPM 客服邮箱：support@oneapm.com OneAPM MI技术交流3群：471152223

Ant安装方式

1. 配置 Ant

- 第一步：命名你的应用程序，获取对应的Token值。
- 第二步：下载最新的 [OneAPM Android SDK](#)
- 第三步：添加 oneapm-android-agent.jar 到工程 libs 目录中，如果项目中没有 libs 目录，请创建一个新的 libs 目录
- 第四步：设置 ANT_OPTS 环境变量

Mac OS / Linux 环境

```
export ANT_OPTS="-javaagent:/path/to/OneAPM_Android_Ant_{VERSION}/class.rewriter.jar"
```

Windows 环境

```
set ANT_OPTS="-javaagent:/path/to/OneAPM_Android_Ant_{VERSION}/class.rewriter.jar"
```

注意：请勿将该环境变量 ANT_OPTS 永久设置到用户或系统环境变量里，否则会影响其他不需要进行嵌码的 Android 项目。建议在单次编译的命令行状态下临时设置该环境变量，或单独在需要嵌码项目的批处理编译脚本中设置该环境变量。

2. 配置授权信息

确保应用程序的 `AndroidManifest.xml` 配置文件中，引入了以下授权：

```
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.INTERNET" />
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!--sdk读取设备识别码需要该权限-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--【非必选】若想知道 Crash 的时候，后台有哪些任务运行，请引入该权限-->
<uses-permission android:name="android.permission.GET_TASKS" />
```

注意：如果您的应用使用 proguard 混淆，请配置以下：

```
-keep class org.apache.http.impl.client.**
-dontwarn org.apache.commons.**
-keep class com.blueware.** { *; }
-dontwarn com.blueware.**
-keep class com.oneapm.** { *; }
-dontwarn com.oneapm.**
-keepattributes Exceptions, Signature, InnerClasses
```

注意：如果您希望保留行号信息，建议您在 proguard.cfg 中添加如下代码：

```
-keepattributes SourceFile, LineNumberTable
```

3. 用户信息配置（可选）

顾名思义，就是说和每一个用户相关联的数据信息。例如崩溃的时候可以根据这个配置查询是哪一个用户发生了崩溃。如下：

```
// 附加数据
HashMap<String, String> extraData = new HashMap<String, String>();
String userTel = "15801388723";
extraData.put("tel", userTel);
extraData.put("userId", "888");
extraData.put("email", "88888@qq.com");

ContextConfig config = new ContextConfig();
String searchValue = userTel;
config.setSearchValue(searchValue); // 设置一个搜索值
config.setExtra(extraData);

OneApAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

4. 集成统计分析功能（可选）

在每个 Activity 中导入 OneApAnalysis 类

```
import com.oneapm.agent.android.module.analysis.AnalysisModule;
```

在每个 Activity 的 onResume() 方法中添加代码：

```
AnalysisModule.onResume();
```

如下示例代码：

```
@Override  
protected void onResume() {  
    super.onResume();  
    AnalysisModule.onResume();  
}
```

在每个 Activity 的 onPause() 方法中添加代码：

```
AnalysisModule.onPause();
```

如下示例代码：

```
@Override  
protected void onPause() {  
    super.onPause();  
    AnalysisModule.onPause();  
}
```

配置渠道信息：如果您的app需要增加渠道信息请在AndroidManifest.xml的Application标签内添加如下（请把YOU CHANNEL替换成您自己的发布渠道例如豌豆荚、360等）

```
<meta-data android:name ="BluewareChannel" android:value="YOUR CHANNEL" />
```

注意：如果两个Activity是继承关系，只需要在父Activity添加即可，如果在两个Activity中同时添加，则会造成重复统计。

5. WebView性能监控（可选）

如果你需要开启此功能，请参考 [Webview性能监控说明](#)。

6. 功能开关（可选）

如果您想使用帧率监控功能可以配置如下代码开启帧监控功能

```
PerformanceConfiguration.getInstance().setEnableFps(true);
```

7. 启动Agent

在默认启动的 Activity 中 import OneApmAgent类

```
import com.oneapm.agent.android.OneApmAgent;
```

在App的第一个Activity的 onCreate() 方法中加入如下调用代码来初始化 OneAPM (其中包含了在步骤 2 中根据应用程序名称而生成的授权编号)

```
OneApmAgent.init(this.getApplicationContext()).setToken("---<YOU TOKEN HERE>---").start();
```

或如果设置了用户信息 (第5步) , 初始化代码如下 (config是类似第5步中的用户信息配置)

```
OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

8. 验证是否成功集成探针

在Logcat中过滤oneapm标签，查看是否有类似如下的日志输出即可(VERSION代表发布版本，因版本不同而不同)。

```
OneAPM started with version :{VERSION}.
```

9. 静候 5 分钟，开启 OneAPM 之旅

静候 5 分钟，等待应用程序向 OneAPM 发送应用程序性能数据，即可开始使用 OneAPM 应用性能管理。

若应用程序无数据展现，或安装过程中有任何问题：

您可以采取以下方式与我们取得联系：

- 技术咨询热线： 400-622-3101

- 销售咨询热线： 400-659-1230
- OneAPM MI技术交流3群： 471152223
-
- OneAPM 客服邮箱： support@oneapm.com

Maven 插件安装方法

1. 命名你的应用程序

点击进入OneAPM Mobile Insight[集成安装界面](#)，命名应用程序。

2. 配置Maven

下面配置使用到的`PLUGIN_VERSION`，需要替换成解压后相应的版本号。例如：下载版本为`OneAPM_Android_Maven_2.0.1`，需将`PLUGIN_VERSION`设置为`2.0.1`。

下载maven插件包解压，假设解压之后的路径是`PATH_TO_ONEAPM_MAVEN_PATH`后续会用到；解压之后会有3个.jar文件一个.pom文件，这些文件下面的操作步骤会用到。

[OneAPM_Android_Maven_Plugin.zip](#)

- 1、注册解压后文件夹中的jar包为本地maven库

注册 `oneapm-android-agent.jar`

```
mvn install:install-file -DgroupId=com.oneapm.agent.android -DartifactId=agent.android -Dversion=PLUGIN_VERSION -Dpackaging=jar -Dfile=PATH_TO_ONEAPM_MAVEN_PATH/oneapm-android-agent.jar
```

注册 `class.rewriter.jar`

```
mvn install:install-file -DgroupId=com.oneapm.agent.android -DartifactId=class.rewriter -Dversion=PLUGIN_VERSION -Dpackaging=jar -Dfile=PATH_TO_ONEAPM_MAVEN_PATH/class.rewriter.jar
```

- 2. 注册 `oneapm-android-maven-plugin.jar`和`plugin.maven.pom` *

```
mvn install:install-file -DgroupId=com.oneapm.agent.android -DartifactId=plugin.maven -Dversion=PLUGIN_VERSION -Dpackaging=jar -Dfile=PATH_TO_ONEAPM_MAVEN_PATH\plugin.maven.jar -DpomFile=PATH_TO_ONEAPM_MAVEN_PATH\plugin.maven.pom
```

例如，配置一个例子：

```
mvn install:install-file -DgroupId=com.oneapm.agent.android -DartifactId=plugin.maven -Dversion=PLUGIN_VERSION -Dpackaging=jar -Dfile=E:\agent\gitlab\android-maven-plugin\target\plugin.maven.jar -DpomFile=E:\agent\gitlab\android-maven-plugin\target\plugin.maven.pom
```

注意：`-Dfile=`这个后面一定不要有空格，否则运行 **Maven** 会报错！！！

- 3、配置本地 pom.xml 文件，添加刚才注册的 jar 包，如下依赖：

```
<dependency>
<groupId>com.oneapm.agent.android</groupId>
<artifactId>agent.android</artifactId>
<version>1.0.8</version>
</dependency>
插件：
<plugin>
<groupId>com.oneapm.agent.android</groupId>
<artifactId>plugin.maven</artifactId>
<version>1.0.8</version>
<executions>
<execution>
<goals>
<goal>instrument</goal>
</goals>
</execution>
</executions>
</plugin>
```

完整如下：

```
<groupId>xxx.yyy.zzzz</groupId>
<artifactId>TestMavenAndroid02</artifactId>
<version>1.0</version>
<packaging>apk</packaging>
<dependencies>
<dependency>
<groupId>com.google.android</groupId>
<artifactId>android</artifactId>
<version>4.1.1.4</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>com.oneapm.agent.android</groupId>
<artifactId>agent.android</artifactId>
<version>1.0.8</version>
</dependency>
</dependencies>
<build>
<finalName>${project.artifactId}</finalName>
<sourceDirectory>src</sourceDirectory>
<plugins>
<plugin>
<groupId>com.jayway.maven.plugins.android.generation2</groupId>
<artifactId>android-maven-plugin</artifactId>
<version>4.0.0-rc.2</version>
<configuration>
<sdk>
<platform>19</platform>
</sdk>
<manifest>
<debuggable>true</debuggable>
</manifest>
</configuration>
<extensions>true</extensions>
</plugin>
<plugin>
<groupId>com.oneapm.agent.android</groupId>
<artifactId>plugin.maven</artifactId>
<version>1.0.8</version>
<executions>
<execution>
<goals>
<goal>insstrument</goal>
</goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

3. 配置授权信息

确保应用程序的 `AndroidManifest.xml` 配置文件中，引入了以下授权：

```
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.INTERNET" />
<!--发送性能数据到服务器需要该权限-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!--sdk读取设备识别码需要该权限-->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!--【非必选】若想知道 Crash 的时候，后台有哪些任务运行，请引入该权限-->
<uses-permission android:name="android.permission.GET_TASKS" />
```

注意：如果您的应用使用 proguard 混淆，请配置以下：

```
-keep class org.apache.http.impl.client.**
-dontwarn org.apache.commons.**
-keep class com.blueware.** { *; }
-dontwarn com.blueware.**
-keep class com.oneapm.** {*;}
-dontwarn com.oneapm.**
-keepattributes Exceptions, Signature, InnerClasses
```

注意：如果您希望保留行号信息，建议您在 `proguard.cfg` 中添加如下代码：

```
-keepattributes SourceFile, LineNumberTable
```

4. WebView 性能监控（可选）

如果你需要开启此功能，请参考 [WebView 性能监控使用说明](#)。

5. 用户信息配置（可选）

顾名思义，就是说和每一个用户相关联的数据信息。例如崩溃的时候可以根据这个配置查询是哪一个用户发生了崩溃。如下：

```
// 附加数据
HashMap<String, String> extraData = new HashMap<String, String>();
String userTel = "15801388723";
extraData.put("tel", userTel);
extraData.put("userId", "888");
extraData.put("email", "88888@qq.com");

ContextConfig config = new ContextConfig();
String searchValue = userTel;
config.setSearchValue(searchValue); // 设置一个搜索值
config.setExtra(extraData);

OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

6. 集成统计分析功能（可选）

在每个 Activity 中导入 OneApmAnalysis 类

```
import com.oneapm.agent.android.module.analysis.AnalysisModule;
```

在每个 Activity 的 onResume() 方法中添加代码:

```
AnalysisModule.onResume();
```

如下示例代码:

```
@Override
protected void onResume() {
    super.onResume();
    AnalysisModule.onResume();
}
```

在每个 Activity 的 onPause() 方法中添加代码:

```
AnalysisModule.onPause();
```

如下示例代码:

```

@Override
protected void onPause() {
    super.onPause();
    AnalysisModule.onPause();
}

```

配置渠道信息: 如果您的app需要增加渠道信息请在AndroidManifest.xml的Application标签内添加如下 (请把YOU CHANNEL替换成您自己的发布渠道例如豌豆荚、360等)

```
<meta-data android:name ="BluewareChannel" android:value="YOUR CHANNEL" />
```

注意: 如果两个Activity是继承关系, 只需要在父Activity添加即可, 如果在两个Activity中同时添加, 则会造成重复统计。

7. 功能开关 (可选)

如果您想使用帧率监控功能可以配置如下代码开启帧监控功能

```
PerformanceConfiguration.getInstance().setEnableFps(true);
```

8. 启动Agent

在默认启动的 Activity 中 import OneApmAgent类

```
import com.oneapm.agent.android.OneApmAgent;
```

在App的第一个Activity的 onCreate() 方法中加入如下调用代码来初始化 OneAPM (其中包含了在步骤 2 中根据应用程序名称而生成的授权编号)

```
OneApmAgent.init(this.getApplicationContext()).setToken("---<YOU TOKEN HERE>---").start();
```

或如果设置了用户信息 (第5步), 初始化代码如下 (config是类似第5步中的用户信息配置)

```
OneApmAgent.init(this.getApplicationContext()).setContextConfig(config).setToken("---<YOU TOK
```

9. 验证是否成功集成探针

在Logcat中过滤oneapm标签，查看是否有类似如下的日志输出即可(VERSION代表发布版本，因版本不同而不同)。

```
OneAPM started with version :{VERSION}.
```

10. 静候 5 分钟，开启 OneAPM 之旅

静候 5 分钟，等待应用程序向 OneAPM 发送应用程序性能数据，即可开始使用 OneAPM 应用性能管理。

若应用程序无数据展现，或安装过程中有任何问题：

您可以采取以下方式与我们取得联系：

技术支持热线：400-622-3101

OneAPM 客服邮箱：support@oneapm.com OneAPM MI技术交流3群：471152223

IntelliJ IDEA eclipse 项目安装方法

本文主要介绍在 IntelliJ IDEA eclipse 环境下安装 OneAPM Android SDK 的步骤：

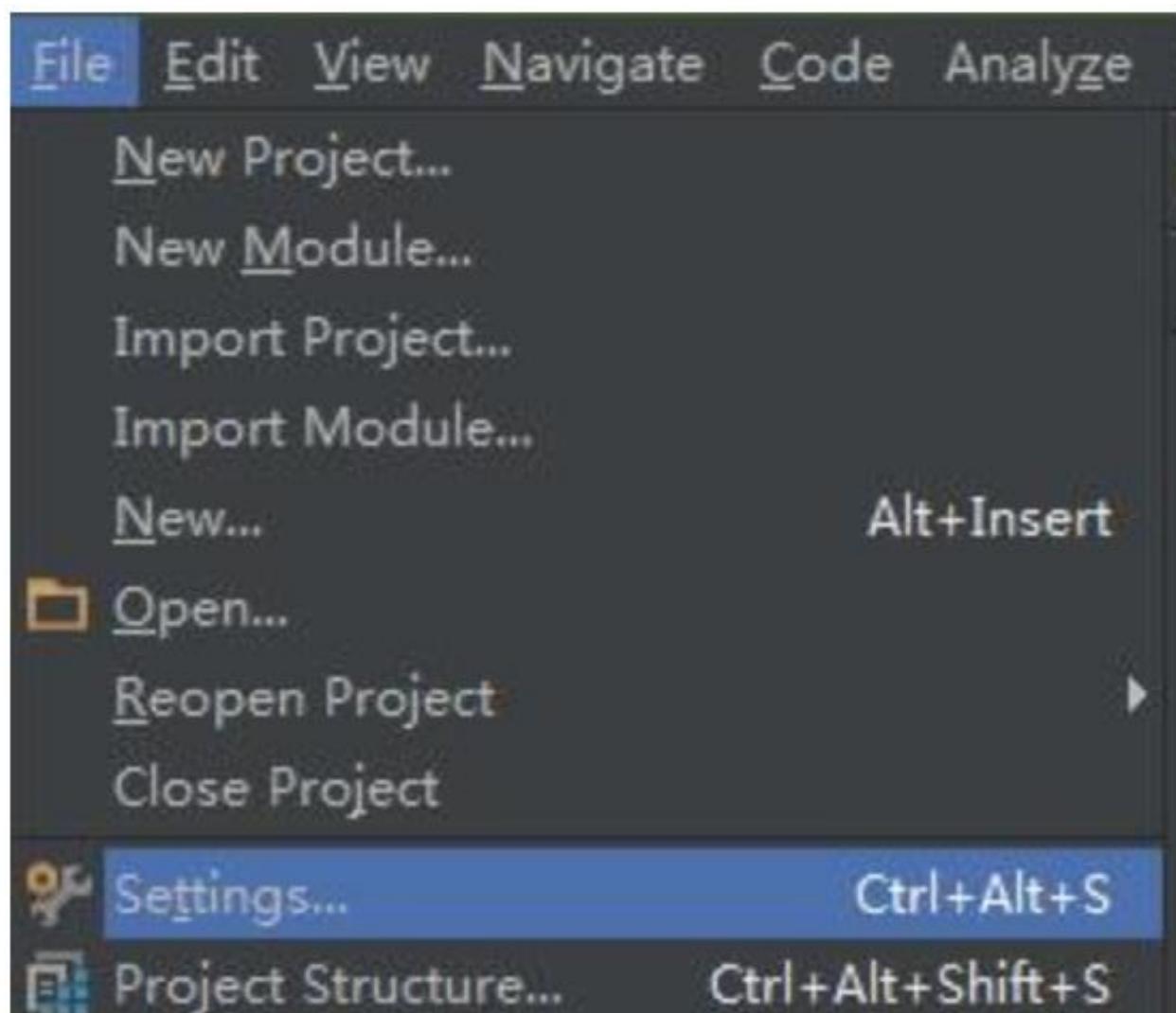
1、下载 Anroid Agent SDK

在移动应用 > 添加应用 > Android 页面，根据 Ant 方法下载 OneAPM Android Agent SDK，解压到某处。如下图：

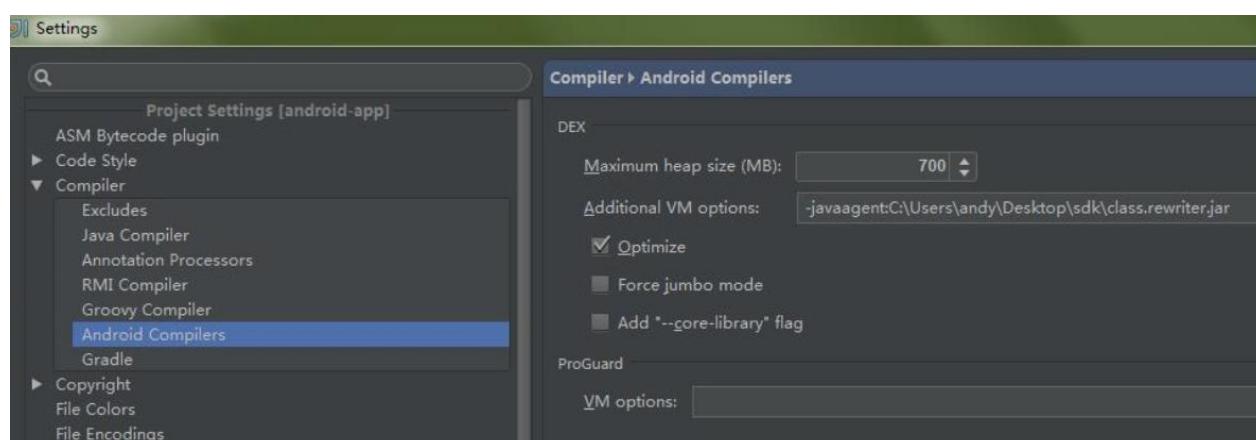
 class.rewriter.jar	9/18/2014 11:5
 oneapm-android-agent.jar	10/8/2014 11:0

2、设置 Android 编译的代理路径

- 选择 file->settings



- 找到 Android Compilers，设置 Additional VM 参数



注意：Additional VM 参数值的设置为： -javaagent: 你的 class.rewriter.jar 文件所在的绝对路径。

此处的值为 -javaagent: C:\Users\andy\Desktop\sdk\class.rewriter.jar。

其中 C:\Users\andy\Desktop\sdk\class.rewriter.jar 是我本地的路径。

3、拷贝 oneapm-android-agent.jar 文件到工程的 libs 目录

添加 libs 目录下的 oneapm-android-agent.jar 到你的 build path 中。

4、添加 OneAPM 启动代码

在你启动页面，一般是 onCreate 的方法中添加 OneAPM 启动代码。

启动代码示例：

```
OneApmAgent.init(this.getApplicationContext()).setToken("----<YOU TOKEN HERE>---").start();;
```

注册 OneAPM 账户，选择移动应用，点击添加应用程序即可获得 App token。

其余流程和 Ant 安装流程一样，请自行对照并查看配置等是否遗漏。

5、重建项目，静候数据

执行 Build-->Rebuild Project 重建项目，如果一切顺利，大约2、3分钟的操作之后即可在后台看见数据了。

2.0.4. 以上版本安装方式

获取探针的方式请根据自身环境并参照2.0.3.* 版本，启动及配置如下；

1、启动Agent

在默认启动的 Activity 中 import OneApmAgent类

```
import com.oneapm.agent.android.OneApmAgent;
```

在 onCreate() 方法中加入如下方法来初始化OneAPM（其中包含了根据应用程序名称而生成的授权编号）

```
OneApmAgent.init(this)
.setToken("<use app token created at step 1>")
.start();
```

配置渠道信息，在AndroidManifest.xml文件中添加meta-data项，要添加在Application标签内部

```
<meta-data android:name = "BluewareChannel" android:value="****" />
```

*代表App的发布渠道，如下图所示：

```
    android:label="app_demo"
    android:theme="@style/AppTheme" >
<meta-data android:name ="BluewareChannel" android:value="wandoujia" />
```

2、加统计分析功能

在每个 Activity 中导入 OneApmAnalysis 类

```
import com.oneapm.agent.android.module.analysis.OneApmAnalysis;
```

在每个 Activity 的 onPause() 方法中添加代码：

```
OneApmAnalysis.onPause();
```

如下图所示：

```
@Override  
protected void onPause() {  
    super.onPause();  
    OneApmAnalysis.onPause();  
}
```

在每个 Activity 的 onResume() 方法中添加代码：

```
OneApmAnalysis.onResume();
```

如下图所示：

```
@Override  
protected void onResume() {  
    super.onResume();  
    OneApmAnalysis.onResume();  
}
```

注意：如果两个Activity是继承关系，只需要在父Activity添加即可，如果在两个Activity中同时添加，则会造成重复统计。

OneAPM_iOS_SDK Objective-C 安装文档

1. 注册 OneAPM

注册 OneAPM，登录账号后点击"Mobile Insight（移动应用性能管理）"，点击苹果图标进入下载安装页面。

您也可以根据本文的安装步骤进行安装。

2. 配置.framework形式开发包

(1) 下载并解压 OneAPM iOS SDK

下载最新版本的 iOS SDK: 在[安装步骤页](#)选择应用类别，再在“命名你的应用程序”输入项目名称，点击提交，记下随后出现的 Token。

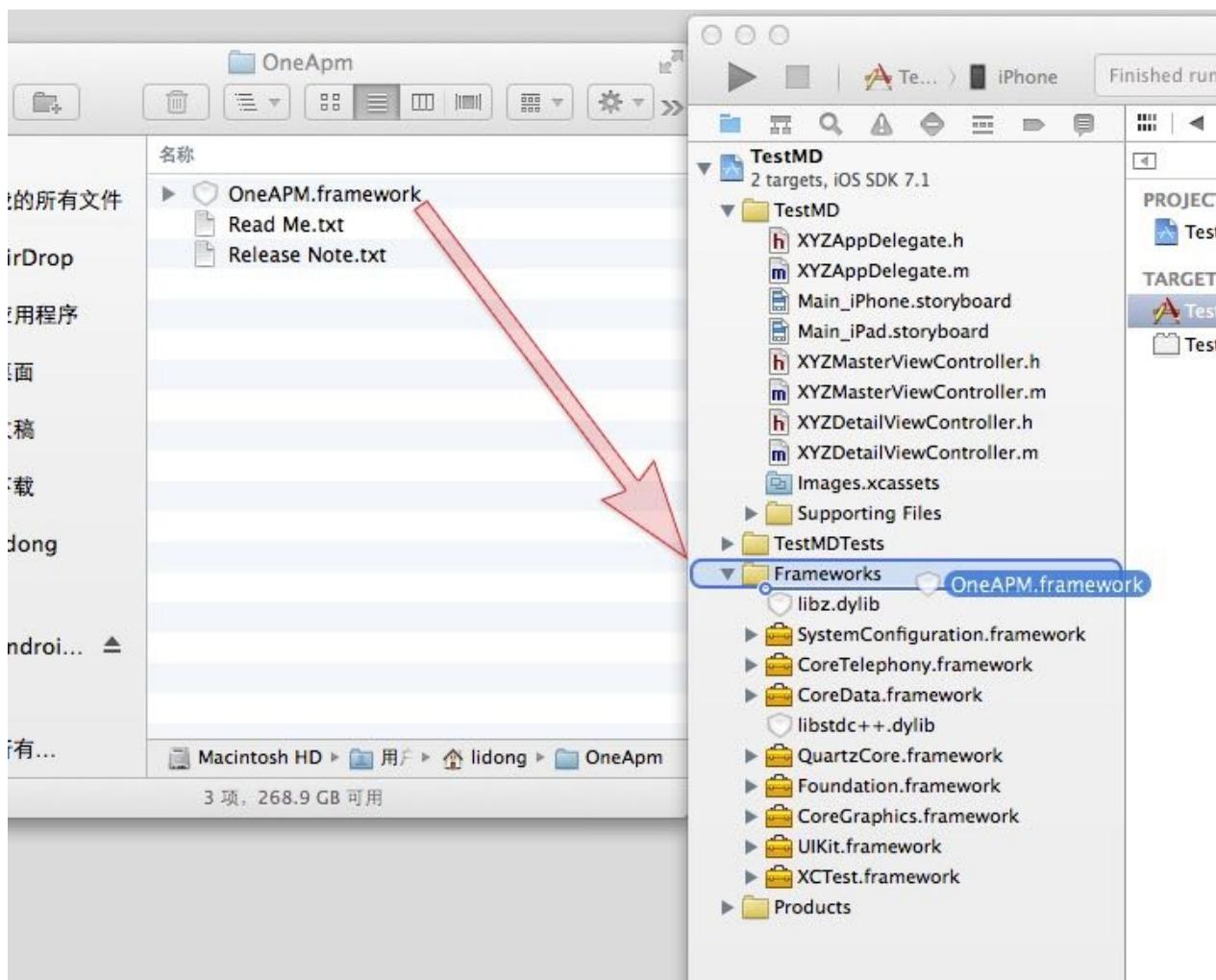
[OneAPM_iOS_SDK \(Xcode6 及以下或者 App Bitcode 无效\)](#)

[OneAPM_iOS_SDK \(Xcode7 及以上并且 App Bitcode 有效\)](#)

(2) 添加 OneAPM Framework 至 Xcode 项目中

解压 SDK，并将「OneAPM.framework」文件夹从 Finder 中拖拽至 Xcode 项目中（悬停至导航窗口的项目中）。

出现提示窗口时，选择「Copy items into destination...」和「Create folder references...」。



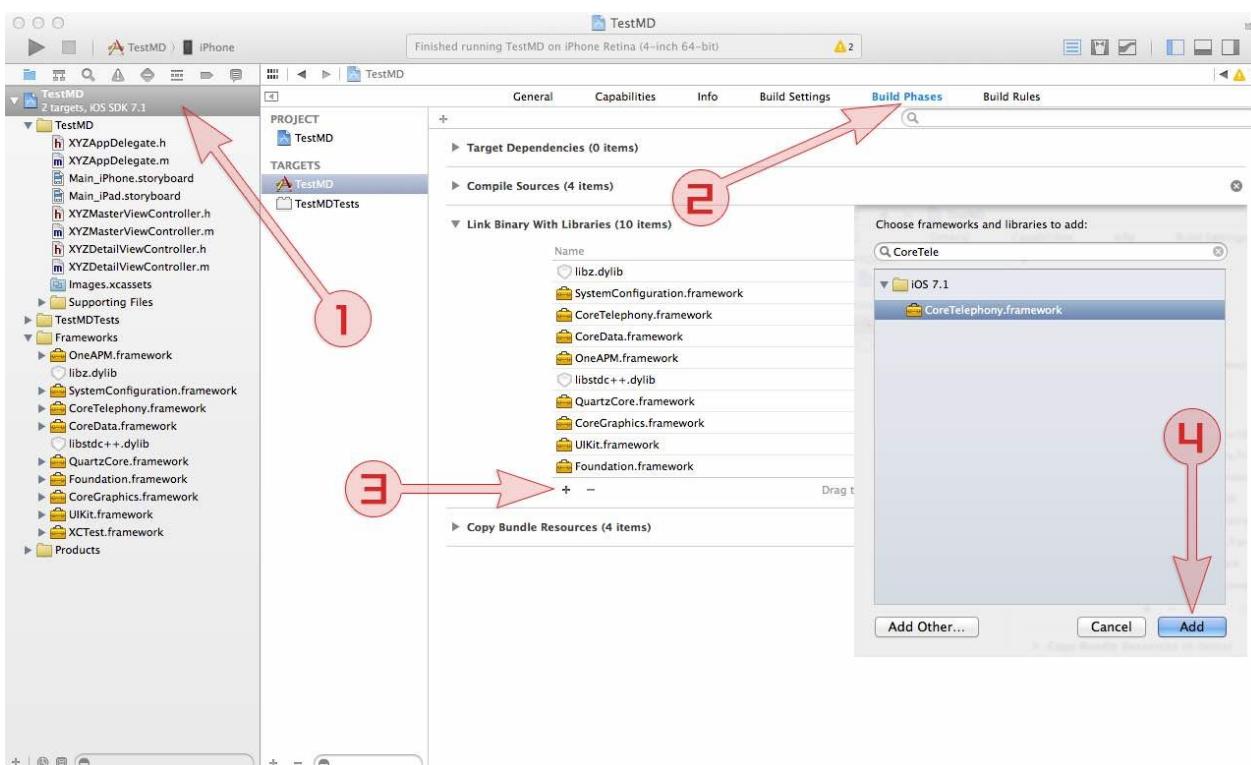
(3) 在 Linker Settings 中添加以下 5 个 Libraries

在项目导航窗口内点击你的 Project，并选中你的 App，然后选择「Build Phases」选项卡。

打开「Link Binary with Libraries」列表。

点击添加：

- SystemConfiguration.framework
- CoreTelephony.framework
- CoreData.framework
- libz.tbd \ libz.dylib (Xcode6及以前版本)
- libstdc++.tbd \ libstdc++.dylib (Xcode6及以前版本)



3. 引入SDK

在项目文件 [app_name]-Prefix.pch (通常在文件夹「Support Files」中) 中，引入 OneAPM 头文件：

```
import <OneAPM/OneAPM.h>
```

在文件 main.m 中添加如下代码，

```
int main(int argc, char * argv[]) {
@autoreleasepool {
[OneAPM startWithApplicationToken: @ ""];
return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
}
}
```

4. 运行应用程序并使用

Clean Project，并重新在模拟器或设备中启动应用程序，开始应用性能管理。

(注：开发时若要使用“崩溃统计”功能，请选Release模式在真机上运行)

请静候 1分钟，等待应用程序向 OneAPM 发送应用程序性能数据，即可开始使用 OneAPM 应用性能管理功能。

5. 符号化表（dSYM文件）上传

当应用列表可以显示出应用“包名”等信息后：

- a.点击“崩溃”进入崩溃信息展示界面；
- b.点击崩溃信息界面右上角“上传dSYM文件”按钮，选择APP版本号，选择要上传的dSYM文件"选取"，“保存”。

OneAPM_iOS_SDK Swift 安装文档

1. 注册 OneAPM

注册 OneAPM，登录账号后点击"Mobile Insight（移动应用性能管理）"，点击苹果图标进入下载安装页面。

您也可以根据本文的安装步骤进行安装。

2. 配置.framework形式开发包

(1) 下载并解压 OneAPM iOS SDK

下载最新版本的 iOS SDK: 在[安装步骤页](#)选择应用类别，在“命名你的应用程序”输入项目名称，点击提交，记下随后出现的 Token。

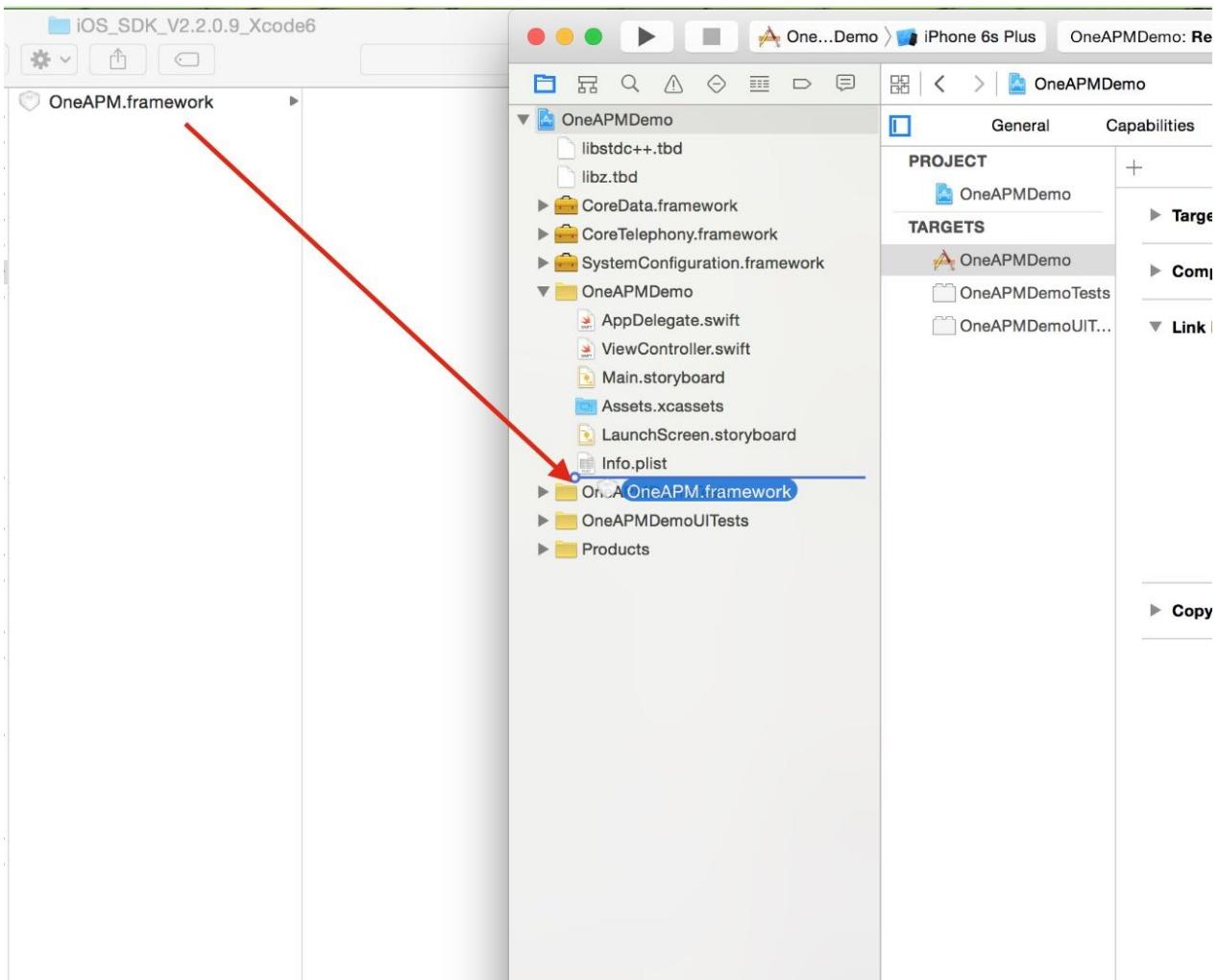
[OneAPM_iOS_SDK \(Xcode6 及以下或者 App Bitcode 无效\)](#)

[OneAPM_iOS_SDK \(Xcode7 及以上并且 App Bitcode 有效\)](#)

(2) 添加 OneAPM Framework 至 Xcode 项目中

解压 SDK，并将「OneAPM.framework」文件夹从 Finder 中拖拽至 Xcode 项目中（悬停至导航窗口的项目中）。

出现提示窗口时，选择「Copy items into destination...」和「Create folder references...」。



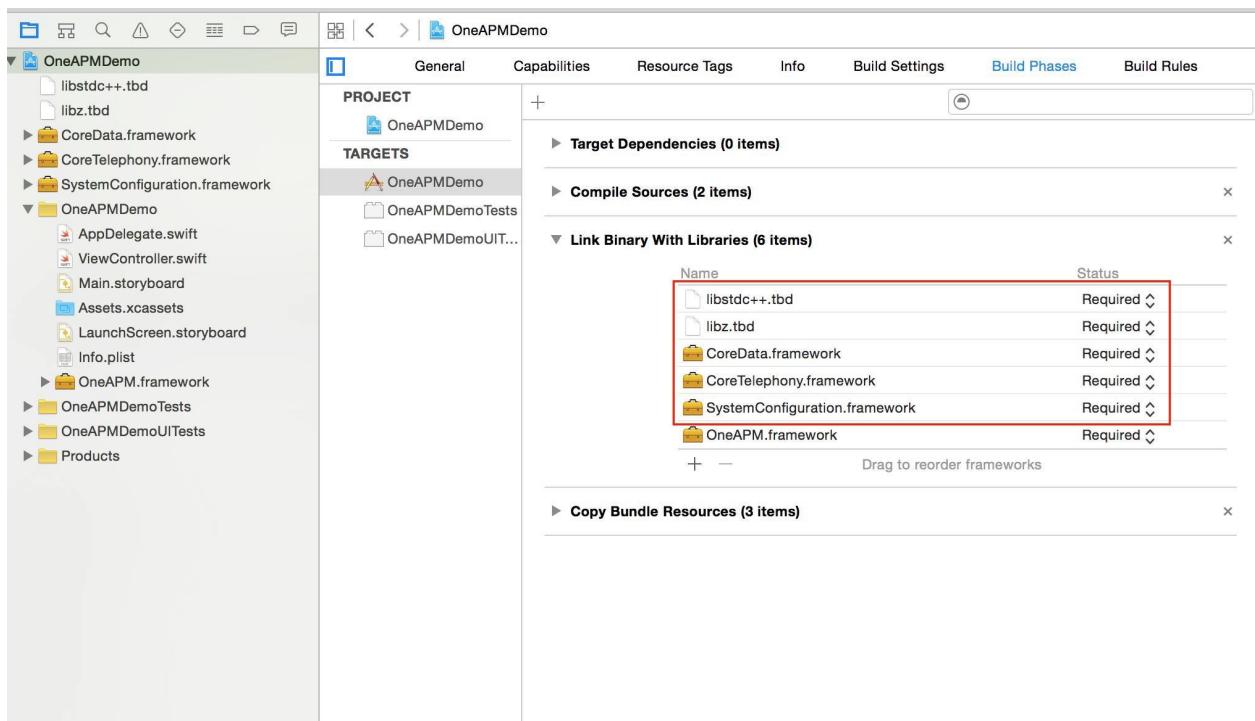
(3) 在 Linker Settings 中添加以下 5 个 Libraries

在项目导航窗口内点击你的 Project，并选中你的 App，然后选择「Build Phases」选项卡。

打开「Link Binary with Libraries」列表。

点击添加：

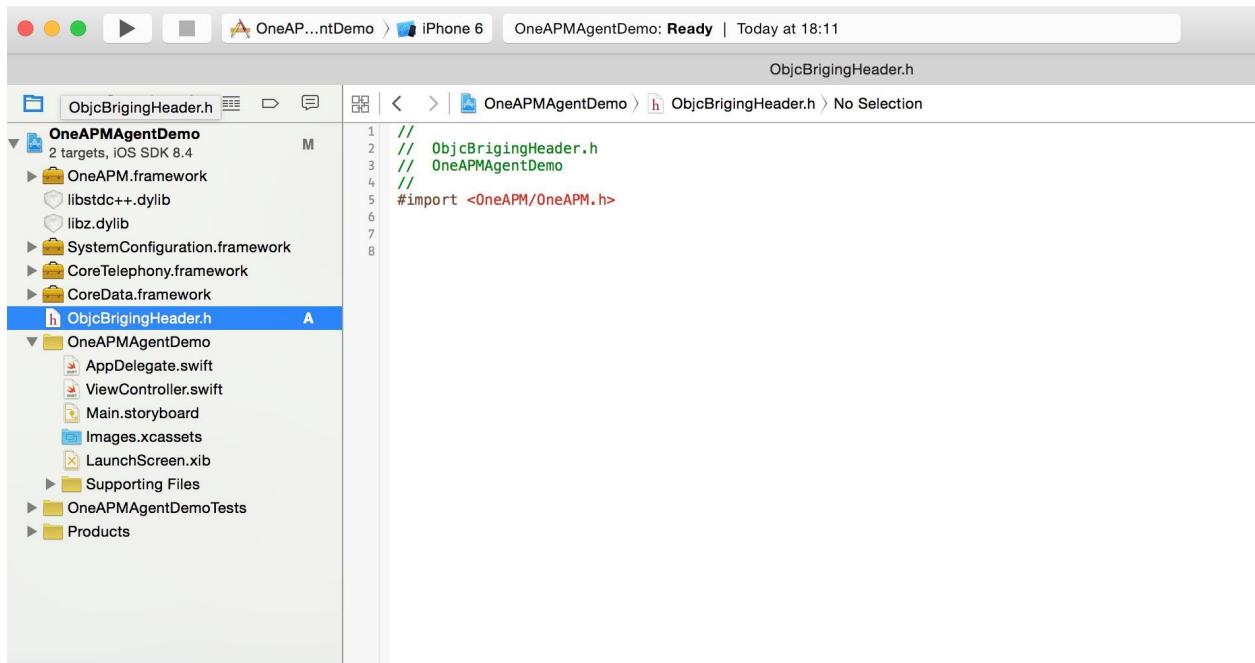
- SystemConfiguration.framework
- CoreTelephony.framework
- CoreData.framework
- libz.tbd \ libz.dylib (Xcode6及以前版本)
- libstdc++.tbd \ libstdc++.dylib (Xcode6及以前版本)



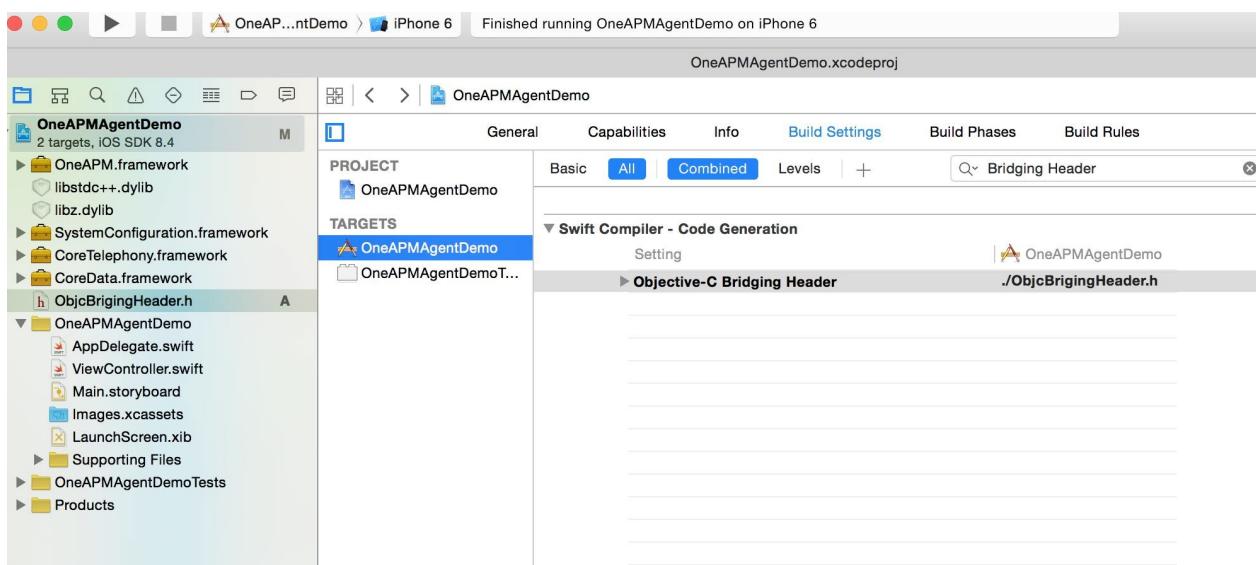
3. 引入SDK

在Swift工程中新建.h文件，例如命名为ObjcBridgingHeader.h，在该文件中引入OneAPM头文件：#import <OneAPM/OneAPM.h>

```
import <OneAPM/OneAPM.h>
```



打开Swift工程的Build Settings设置界面，搜索Bridging Header，找到Objective – C Bridging Header设置项，添加上一步中创建的头文件，到此，在Swift工程中即可调用SDK接口。



在文件 `AppDelegate.swift` 中添加如下代码，并确保它在 `application:didFinishLaunchingWithOptions` 的第一行中。

```
```[OneAPM startWithApplicationToken: @"<use app token created at step 1>"];```
```

## 4. 运行应用程序并使用

Clean Project，并重新在模拟器或设备中启动应用程序，开始应用性能管理。

(注：开发时若要使用“崩溃统计”功能，请选Release模式在真机上运行)

请静候 1分钟，等待应用程序向 OneAPM 发送应用程序性能数据，即可开始使用 OneAPM 应用性能管理功能。

## 5. 符号化表（dSYM文件）上传

当应用列表可以显示出应用“包名”等信息后：

- a. 点击“崩溃”进入崩溃信息展示界面；
- b. 点击崩溃信息界面右上角“上传dSYM文件”按钮，选择APP版本号，选择要上传的dSYM文件“选取”，“保存”。



# 应用

关于移动应用的基本信息，Mobile Insight 主要提供以下信息：

## 总览

该模块分新旧两版，新旧两版可自由切换。

1. 新版界面：

该版共分为5个模块

- 性能评分

使用页面执行时间，崩溃率，网络错误率，HTTP错误率等各项核心性能指标给出APP整体性能评分。



评分指标：

IOS：平均执行时间（访问次数），崩溃次数（影响用户数），HTTP错误率（影响用户数），网络故障率（影响用户数）

Android：平均执行时间（访问次数），崩溃次数（影响用户数），HTTP错误率（影响用户数），网络故障率（影响用户数）

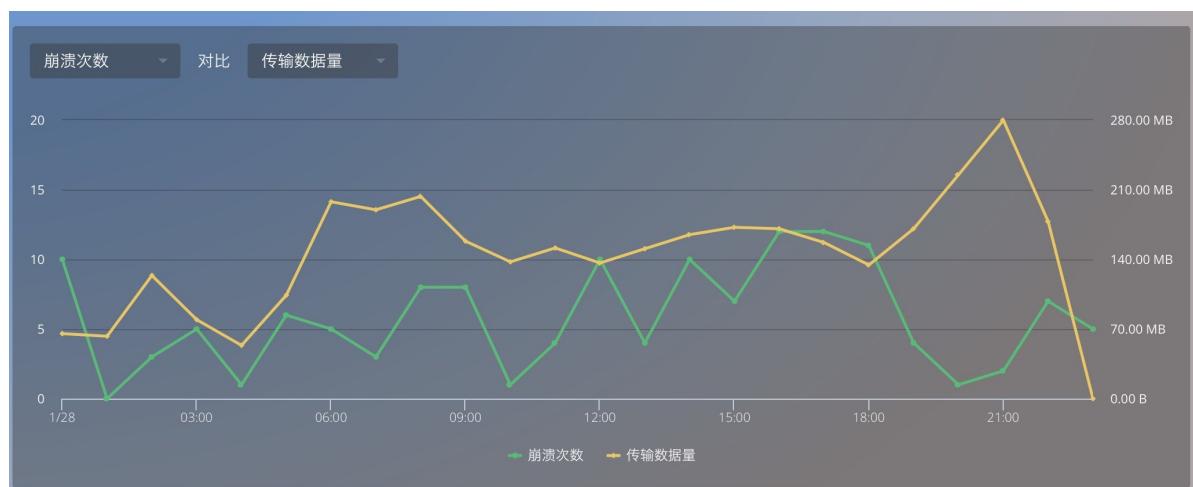
性能评级：

得分 / 次数	性能评级	标示	详情
(80-100) 分	本次检测性能状况良好	绿色	检测各项性能指标，未发现重要性能问题
(60-80) 分	本次检测性能状况一般	黄色	检测各项性能指标，应用存在潜在风险，请持续关注APP性能
< 60 分	本次检测性能状况较差	红色	检测各项性能指标，应用存在一些列问题，会影响APP使用
使用次数为0	无性能状况信息，今日无用户使用应用	—	—

- 性能指标对比分析

选择任意两个指标进行对比。

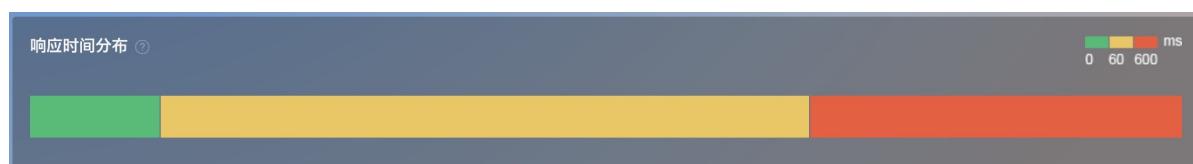
对比指标：平均响应时间，传输数据量，网络错误率，日活，崩溃次数。



- 响应时间分布

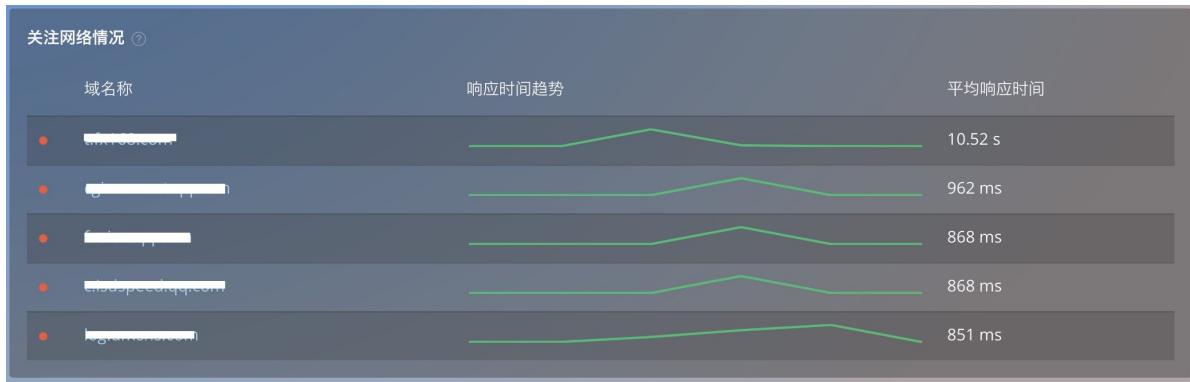
所选时间范围内，所有网络请求耗时进行阶段划分，默认按照订制区间[0-60ms), [60-600ms), [600-Nms)划分网络性能状况。

色块区间：绿[0-60ms)，黄[60-600ms), 红[600-Nms)



- 关注网络情况

展示用户最关心的网络请求的趋势信息，默认展示平均响应时间TOP5的请求。



- 观测指标分布

选择查看各性能指标地域分布情况。

下拉选项： 平均响应时间， 传输数据量， 网络错误率



## 2.旧版界面：

该模块主要展示以下六张图表：

- 执行时间：展示网络请求，数据库，UI 渲染，图片解析，JSON 解析的执行时间趋势图
- 崩溃数：按已解决与未解决划分的崩溃次数时间曲线。
- HTTP 响应时间：展示各类 HTTP 请求响应时间 TOP 5 的时间曲线。HTTP 响应时间是指从发送 HTTP 请求开始，到收到所有响应内容的时间。
- HTTP 错误率 & 网络故障率：以堆叠曲线图展示当前应用的各类 HTTP 错误率和网络故障率历史曲线。HTTP 错误率是指在选定时间段内，HTTP 错误数量与请求数量的比率。网络故障率是指网络错误数量与请求数量的比率。其中发生网络错误的 HTTP 请求数指发生 DNS 解析错误、无法建连、连接超时等网络方面的错误数量。

- 会话数-应用版本：按版本展示会话数 TOP 5 的时间曲线。
- 最慢交互：展示交互耗时 TOP 5 的 Trace 列表。点击 Trace 名会跳转到该交互的详细页面。



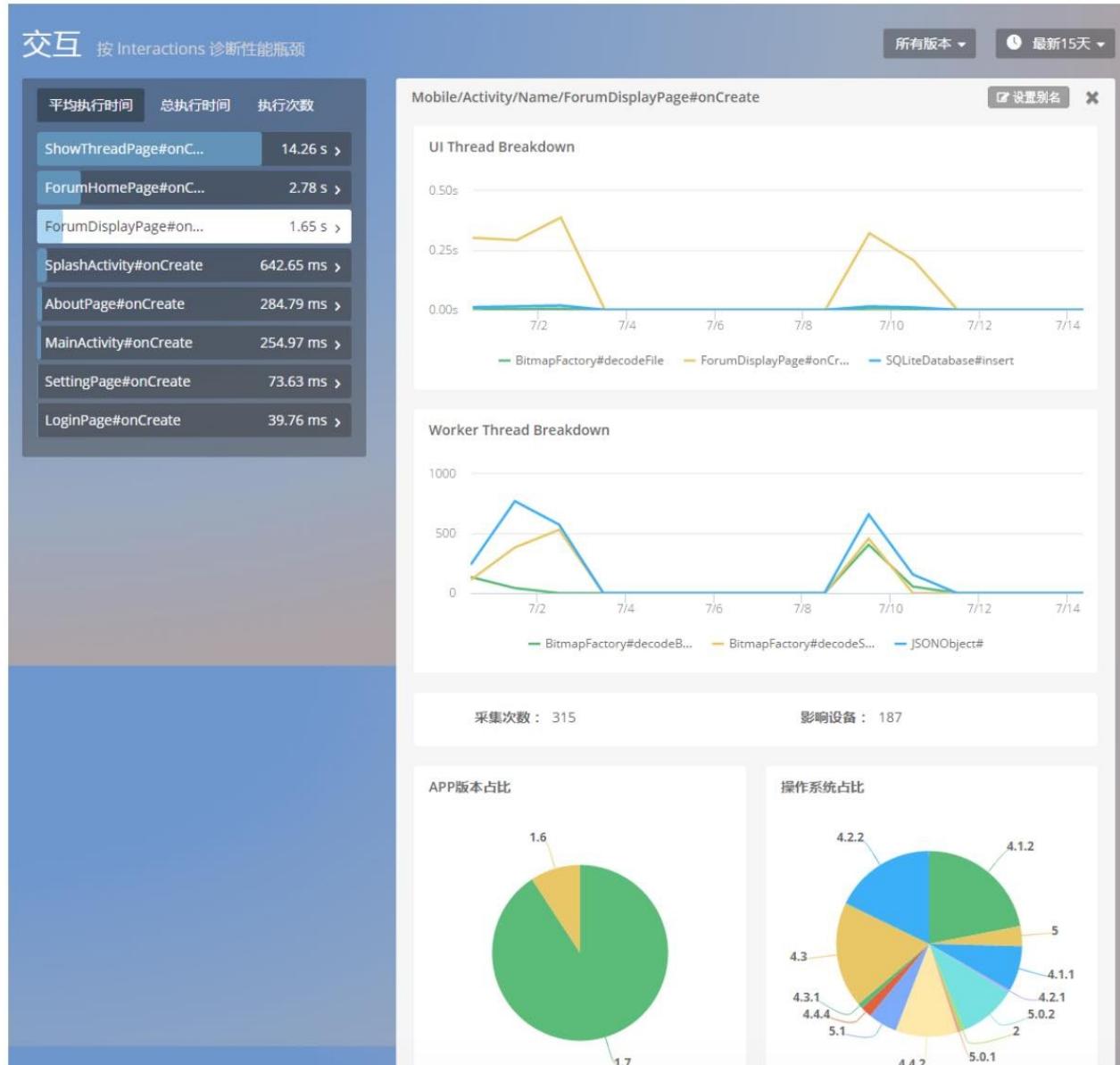
关键词：性能评分 总览 响应时间

# 交互

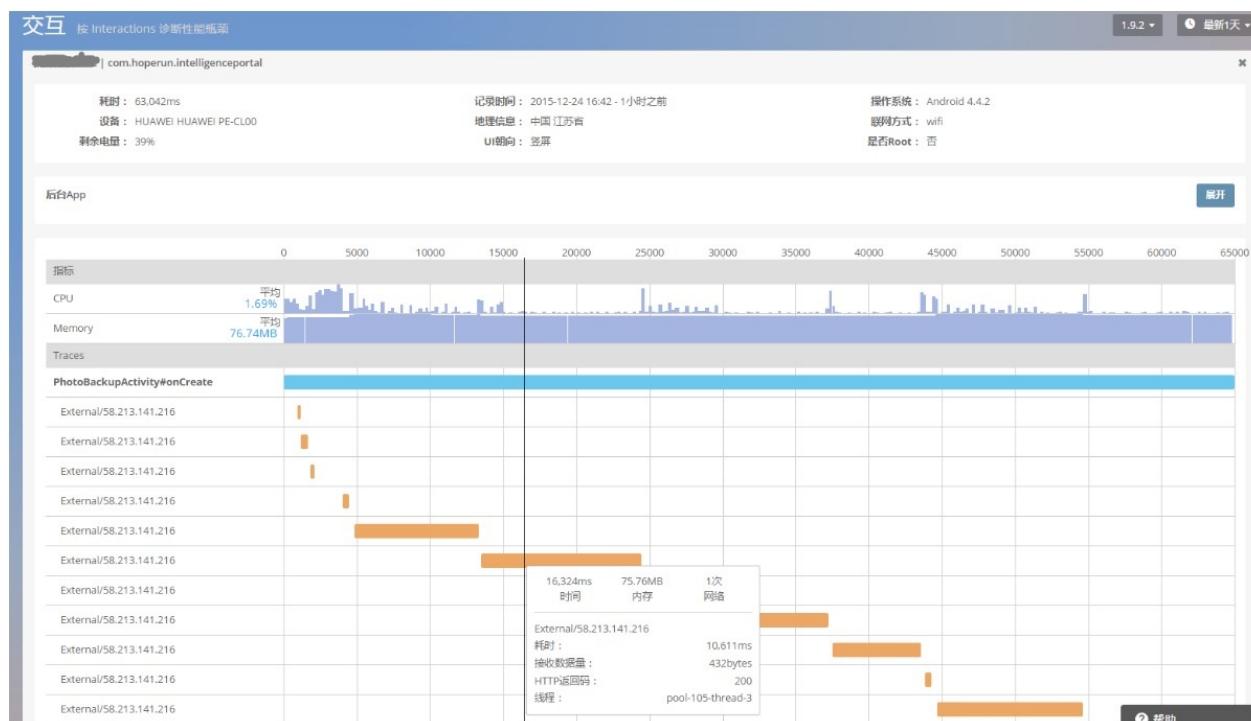
交互功能用于监控终端用户与 App 交互时被调用代码的执行情况，用于分析发现耗时的用户交互。如图：



选择某个 Activity，查看该 Activity UI 主线程和 Worker 线程耗时状况。查看峰值，对执行次数最多，执行时间最长的线程进行优化。然后结合业务逻辑优化代码，尽量将耗时操作放在 worker 线程上，避免阻塞主线程进而影响用户体验。



点击慢交互列表中的任意交互名，可查看其详情：



- 在该页面中显示发生缓慢的某次交互中 CPU/memory 消耗状况，网络请求，异步线程的分布情况。
- 页面上方展示了，用户在操作该页面（Activity/View）时所使用的应用版本，设备操作系统，设备厂商，操作时间，所在地区，联网方式，剩余电量，UI朝向。
- 指标信息图则展示的是下方Trace时序图的堆叠形式，从内存堆叠图可以看出，该应用执行过程中内存释放情况。网络堆叠图可以看出同一时刻发生了多少异步网络请求，是否与最初设计的业务逻辑相符合。从而判断出造成某一时刻慢的原因。
- Trace 时序图是按照页面内线程加载顺序展示各个线程耗时情况，内存消耗，传输数据量等信息。
- 线程信息包括：Image, Database, Json, ViewLoading, Network, WebView 共6类。
- 主线程使用粗体标注，时序图中色块长度代表线程耗时时长。也可以帮助非开发该功能的程序员看清楚其中的逻辑关系，快速定位造成 Activity 慢的原因。

关键词：交互 线程 CPU 主线程 UI线程

# ANR

ANR，全称 Application Not Responding 或 The Application of Non Response，意思是“应用无响应”。本文主要介绍 ANR 产生原因以及 OneAPM Mobile Insight 提供的 ANR 数据抓取及展示功能。

引起 ANR 问题的根本原因，可以大致归为两类：

- 应用进程自身引起

例如：主线程阻塞、挂起、死循环。应用进程的其他线程 CPU 占用率过高，导致主线程无法抢占 CPU 时间片等。

- 其他进程间接引起

例如：当前应用进程与其他进程开展通信请求，其他进程长时间没有反馈。或者其他进程 CPU 占用率过高，导致当前应用进程无法抢占 CPU 时间片。

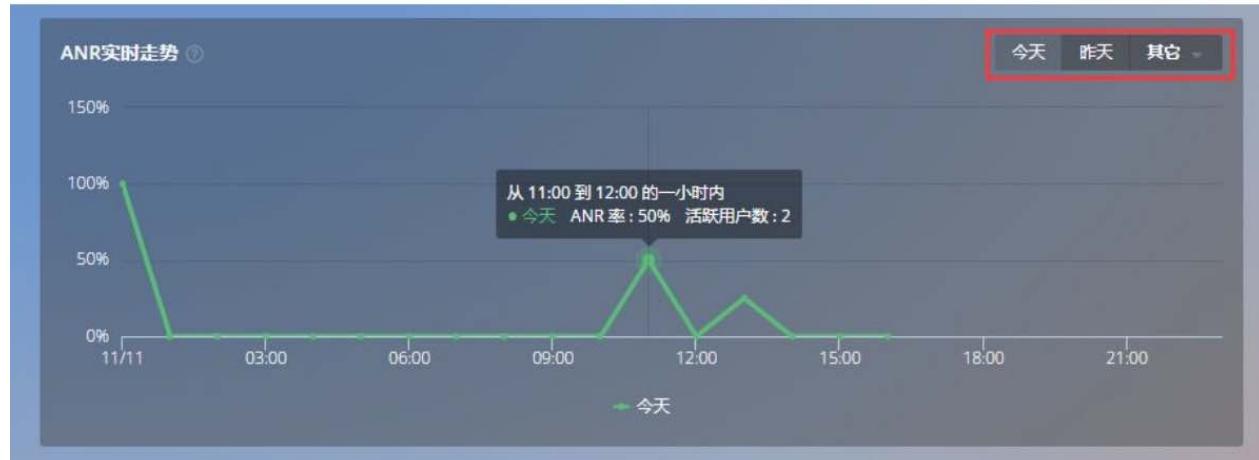
ANR 对于应用的影响并不亚于 Crash。那我们如何监控 ANR 呢？

OneAPM Mobile Insight 提供了 ANR 监控功能：



ANR 统计包括：ANR 趋势信息以及 ANR 率。默认情况下展示 ANR 当天趋势信息，若鼠标悬停，则展示此时 ANR 率。

**ANR 率 = ANR 影响用户/活跃用户数。**



ANR 类型列表：按 APP 版本与 ANR 类型进行分类展示 ANR 问题，展示某类 ANR 所影响的用户数，以此表征问题严重程度。

问题描述	APP 版本	发生次数	影响用户数	首次发生时间	最后发生时间	状态
ANR keyDispatchingTimedOut	1.4.1	3	1	2015/11/11 00:16:08	2015/11/11 13:18:43	已解决

点击列表中问题描述，查看 ANR 信息详情：

查看某版本上该类 ANR 问题分布设备与操作系统版本的比例，以及 ANR 发生趋势，帮助您从宏观上控制 ANR 影响范围。





ANR 线程等信息记录，帮助用户定位造成 ANR 的原因。

关键词： **ANR** 应用无响应 线程阻塞

# 崩溃

崩溃界面按照崩溃解决与否、应用版本号实时展示崩溃数量变化图。崩溃列表则按照崩溃类型、代码信息、代码位置、首次发生时间、最后发生时间、应用版本号、发生次数、影响设备数以及崩溃状态展示崩溃的详细信息。

崩溃次数统计图展示了 app 发生的崩溃数。它分为两部分：已解决和未解决。当 OneAPM mobile 发现了崩溃现象，将把该崩溃归类到未解决类别。当你解决一个 bug 时，你可以标记该 crash 为已解决状态。在图表中，所有出现该崩溃都会重新分配到已解决的类别中，重新规划崩溃次数图。随着你崩溃问题的修复，查看已解决崩溃次数图将降为 0。

注意：如果已解决崩溃数依旧在增长，那么就表明这个所谓的已解决崩溃并没有完全修复，或者用户没有更新 APP。

根据版本的崩溃次数统计图展示了随时间推移 app 发生的崩溃数的趋势。它是按照版本进行分类的。用于检验新发布版本较老版本的性能是否有显著提升。

**备注：**OneAPM 采用 Version Code 为标识，监控应用版本信息，也即上文的应用版本号。



可筛选查看未解决崩溃和被设置为优先级较高的一类崩溃信息。此外，点击代码位置，可跳转至该崩溃的详情页面。该页面以图形化的方式，展示了该崩溃的发生次数、影响的设备类型、操作系统类型，还包含了崩溃发生的特定环境、崩溃类型、崩溃轨迹，以及导致崩溃发生的线程，深入到代码行。

## 崩溃详情

**崩溃** 崩溃统计和详情

所有版本 ▾ 最新15天 ▾

java.lang.IllegalStateException

应用版本：1.7 同类崩溃：1 影响设备：1  
最后发生时间：2015-07-09 22:05 - 5天之前 状态：未解决

崩溃次数

影响设备

影响操作系统版本

崩溃详情 影响用户

第 1/1 个同类崩溃  
发生时间：2015-07-09 22:05

代码定位：ListView.java line 1566 in android.widget.ListView.layoutChildren()  
崩溃类型：java.lang.IllegalStateException  
设备信息：samsung samsung SM-N9009  
内存占用：18MB  
可用磁盘：74.72 MB  
屏幕朝向：竖屏  
APP版本：1.7  
操作系统：Android 4.4.2  
联网方式：wifi  
Agent版本号：1.0.8

错误信息

The content of the adapter has changed but ListView did not receive a notification. Make sure the content of your adapter is not modified from a background thread, but only from the UI thread. Make sure your adapter calls notifyDataSetChanged() when its content changes. [in ListView(2131296284, class com.pediy.bbs.kanxue.widget.XListView) with Adapter(class android.widget.HeaderViewListAdapter)]

后台 App 展开

崩溃轨迹

Session Start > SplashActivity#onCreate 9秒之前 > MainActivity#onCreate 1秒之前 >

ForumDisplayPage#onCreate  
2015-07-09 22:04

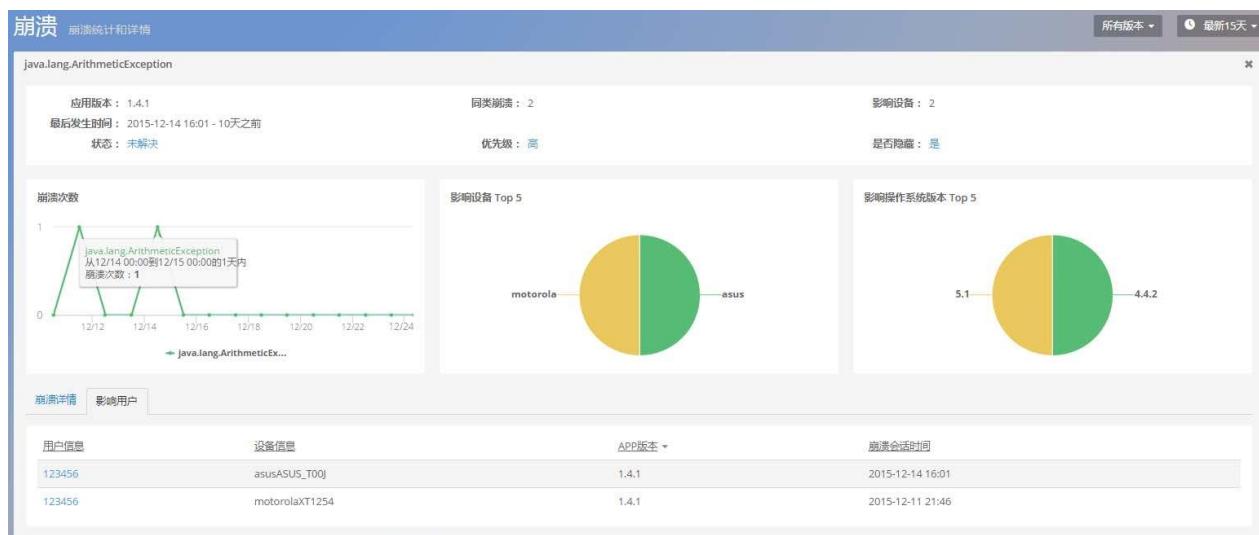
Thread - main

包名	详细信息
android.widget	ListView.java line 1566 in android.widget.ListView.layoutChildren()
android.widget	AbsListView.java line 2578 in android.widget.AbsListView.onLayout()
android.view	View.java line 15706 in android.view.View.layout()

- 统计某类崩溃在各类设备、操作系统和各 app 版本中出现的次数,被该崩溃影响的用户数
- 交互追踪部分展示了所追踪的app,从用户会话开始到发生这一崩溃这段时间内的交互时间轴。这一部分可以帮助您重现用户操作,还原本次崩溃异常现场。

- 堆栈信息部分记录崩溃时的崩溃 log 和线程堆栈详情信息

## 影响用户



展示该类崩溃影响的具体用户的详细信息，点击用户信息，查看该用户使用APP的详细信息。



关键词：崩溃 追踪 影响用户 代码行

# 用户信息

- 通过搜索关联用户信息，得到该用户使用APP时设备的崩溃次数，以及崩溃详情。
- 查看某类崩溃影响的用户信息列表。

注意：用户信息模块崩溃查询功能以及崩溃模块影响用户列表只在付费后可使用或查看。

该模块提供查询单个用户发生的崩溃信息的入口。



此外，通过设置用户识别信息，您可以自定义收集用户信息。我们提供的接口如下：

- IOS 配置用户信息

```
+ (void)setCustomInfo:(NSString *)info;
```

备注：`setCustomInfo` 内容长度为 200 字节，即 100 个汉字长度，超过部分会被截取。

用法示例：

```
[OneAPM startWithApplicationToken:@"225D3C244ACE5E49F1CFA920EF94D8A489"];
[OneAPM setCustomInfo:@"18611421164"];
```

- Android 配置用户信息

```
ContextConfig contextConfig = new ContextConfig();
contextConfig.setSearchField("user_imsi_custom");
```

参数配置：

```
HashMap extraData = new HashMap();
extraData.put("手机号码","");
extraData.put("用户ID","");
extraData.put("邮箱","88888@qq.com");
```

```
contextConfig.setExtra(extraData);
```

## 按照上传信息查询用户崩溃情况

如图，在输入框中键入用户信息，可以查询单个用户发生崩溃的信息。

The screenshot shows a search interface with a search bar containing '123456'. Below the search bar is a table with four columns: User Info, Device Info, APP Version, and Crash Occurrence Time. The table contains three rows of data:

User Info	Device Info	APP Version	Crash Occurrence Time
123456	asusASUS_T00j	1.4.1	2015-12-14 16:01
123456	motorolaXT1254	1.4.1	2015-12-11 21:46
123456	unknownunknown	1.4.1	2015-12-04 18:59

查询结果包括：用户信息，用户使用的设备信息，APP版本以及崩溃开始时间。

## 查看崩溃详情

点击单条信息，可查看发生崩溃时用户所处的环境：

This screenshot shows the detailed view for user 123456. It includes basic info like device (asus ASUS\_T00j), OS (Android 4.4.2), and location (China). The crash details table shows:

崩溃类型	代码信息	代码位置	首次发生时间	最后发生时间	应用版本	发生次数	状态	优先级
java.lang.Arithme...Exception	UserinformationActivity.java com.blueware.android.app.client.UserinformationActivity\$1.onClick()	Line 29	2015-12-14 16:01	2015-12-14 16:01	1.4.1	1	未解决	高

Additional information section shows:

序号	类型	描述
1	手机号码	888888888888
2	用户ID	勤劳的小蜜蜂
3	邮箱	888888@qq.com

点击崩溃列表中的崩溃名，可跳转至崩溃详情页：

This screenshot shows the 'Crash Details' page for the exception 'java.lang.Arithme...Exception'. It includes summary stats (version 1.4.1, last occurrence 2015-12-14 16:01, status unresolved), a timeline chart showing a single spike on Dec 14, and pie charts for top devices (motorola, asus) and operating systems (5.1, 4.4.2).

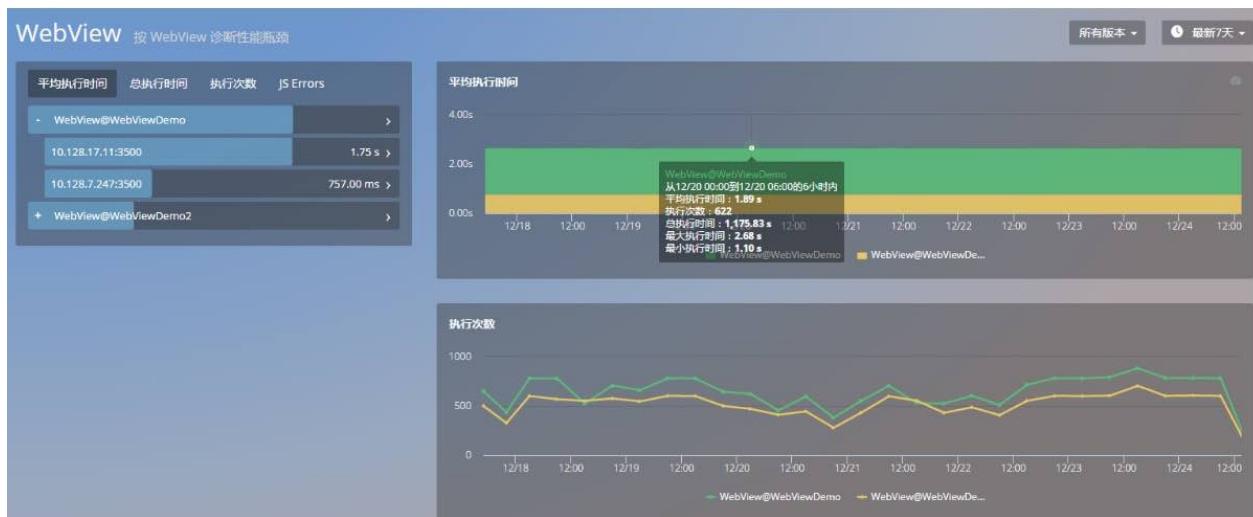
Below the summary are tables for User Info and Device Info.

User Info	设备信息	APP版本	崩溃发生时间
123456	asusASUS_T00j	1.4.1	2015-12-14 16:01
123456	motorolaXT1254	1.4.1	2015-12-11 21:46

关键词：崩溃 影响用户 用户信息

# WebView

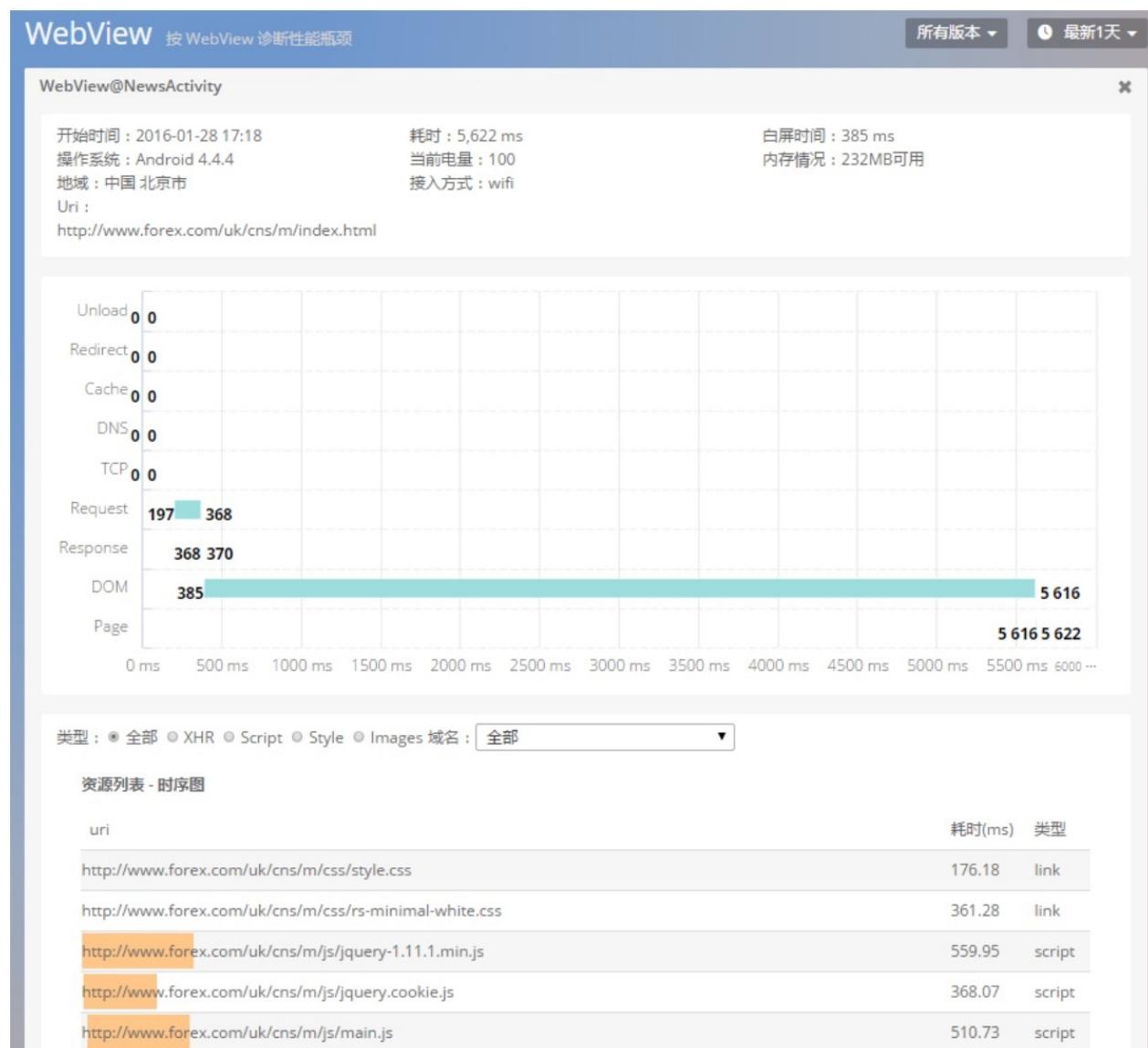
WebView 界面用于监控移动应用中 WebView 页及 WebView 中的请求的性能状况，包含其加载时间、执行时间、执行次数、JS 错误等。



如图，左侧列表按照 WebView 所在的 Activity 进行归类，从平均执行时间、总执行时间、执行次数三个角度进行展示。右侧则展示 WebView 平均执行时间、执行次数的实时动态图。在下部，有 WebView 慢加载追踪列表，展示了慢 WebView 的白屏时间、总耗时、发生时间、操作系统及版本。其中，白屏时间采取了精确的算法，旨在衡量用户真实体验到的加载状况。

慢加载追踪列表				
名称	白屏时间(ms)	耗时(ms)	发生时间	操作系统及版本
WebView@TestWebView	99	111	2015-07-28 11:54	Android 4.4.4
WebView@TestWebView	127	139	2015-07-28 11:51	Android 4.4.4
WebView@TestWebView	179	195	2015-07-15 15:23	Android 4.4.4
WebView@TestWebView	116	135	2015-07-15 15:23	Android 4.4.4

点击任一慢加载 WebView 名，进入其详情页。该页面以时序图的形式展示了单条 WebView 的详细信息。时序图基于 HTML5 页面加载解析过程创建，您可以清晰地了解页面渲染时各个流程的耗时情况，从而有针对性地优化代码。提供资源加载时序图，有效判别造成页面加载慢的耗时资源。



关键词: *WebView JS 资源加载 Ajax JS错误 H5*

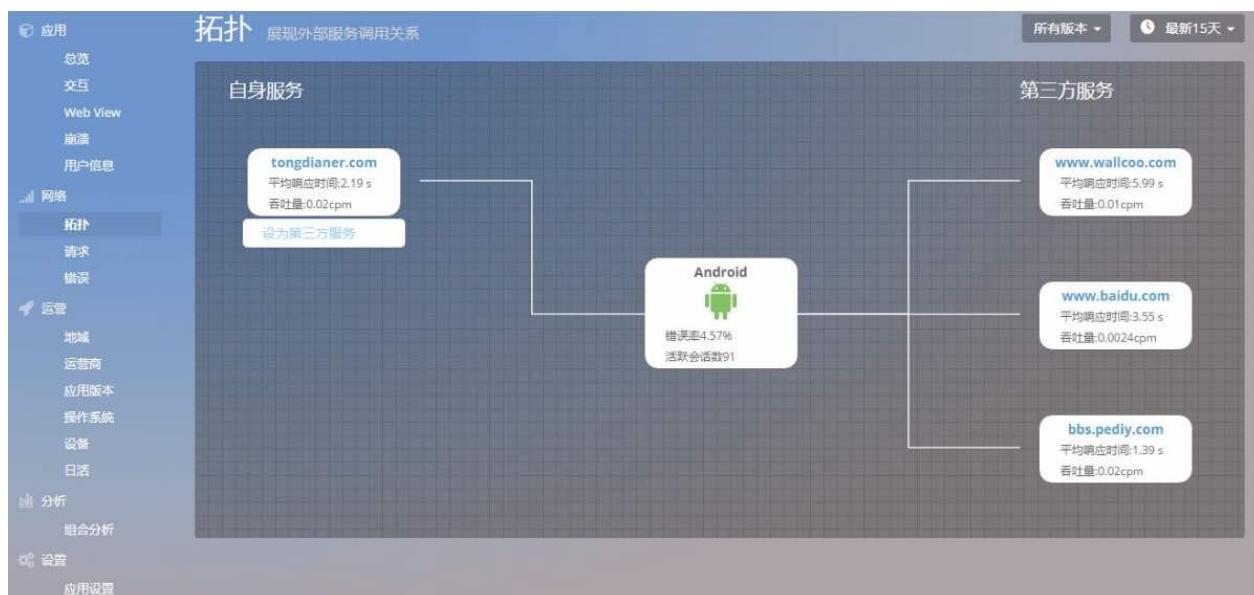
# 网络

针对移动 App 的网络监控，Mobile Insight 主要有以下功能：

- 拓扑
- 请求
- Socket
- 错误

## 1. 拓扑

「拓扑图」展示该应用的网络请求状况。显示App正在使用的服务，以及对应的平均响应时间、吞吐量等。默认情况下，左侧为自身服务、右侧为第三方服务。若您需要自主设定服务所属类别，将鼠标悬停在其图标上即可进行设置。点击服务名称，可跳转到网络请求详情页面。



## 2. 请求

「请求」功能监测分析网络请求的性能数据，帮助运营和开发人员提升用户体验。该界面包含以下内容：

- 网络请求列表：按照平均响应时间、总响应时间、传输数据量、响应次数进行列表。
- 响应时间：展示网络请求平均响应时间TOP5的时间曲线；该时间是指从发送 HTTP

请求开始，到收到所有响应内容的时间。

- 吞吐量：展示吞吐量 TOP 5 的时间曲线；吞吐量是指平均每分钟的 HTTP 请求数。



此外，您可以利用请求列表中的搜索功能，快速检索您关心的网络请求。



点击「请求」列表的URL，可查看对应的请求详细信息：响应时间、吞吐量、网络故障率、HTTP 错误率。



### 3. Socket

「Socket请求」可以监控socket在发送过程中的各种问题。在java中socket有两个类可以实现socket的通讯功能,其一是socket,另一个就是socketChannel,目前可以支持java中两种socket的监控。

具体操作：把socket代码进行一些修改就可以记录数据，这样就可以直观了解socket在传输过程中的问题。

## Socket的监控

首先将socket功能提供一个包装类,然后可以实现数据收集。

具体实现如下：

```
Socket socket = new Socket();
SocketInstrumentation socketInstrumentation = new SocketInstrumentation();

socketInstrumentation.connect(new InetSocketAddress("127.0.0.1",12345));
//读取服务器端数据
DataInputStream input = new DataInputStream(socketInstrumentation.getInputStream());
//向服务器端发送数据
DataOutputStream out = new DataOutputStream(socketInstrumentation.getOutputStream());

...
socketInstrumentation.close();
```

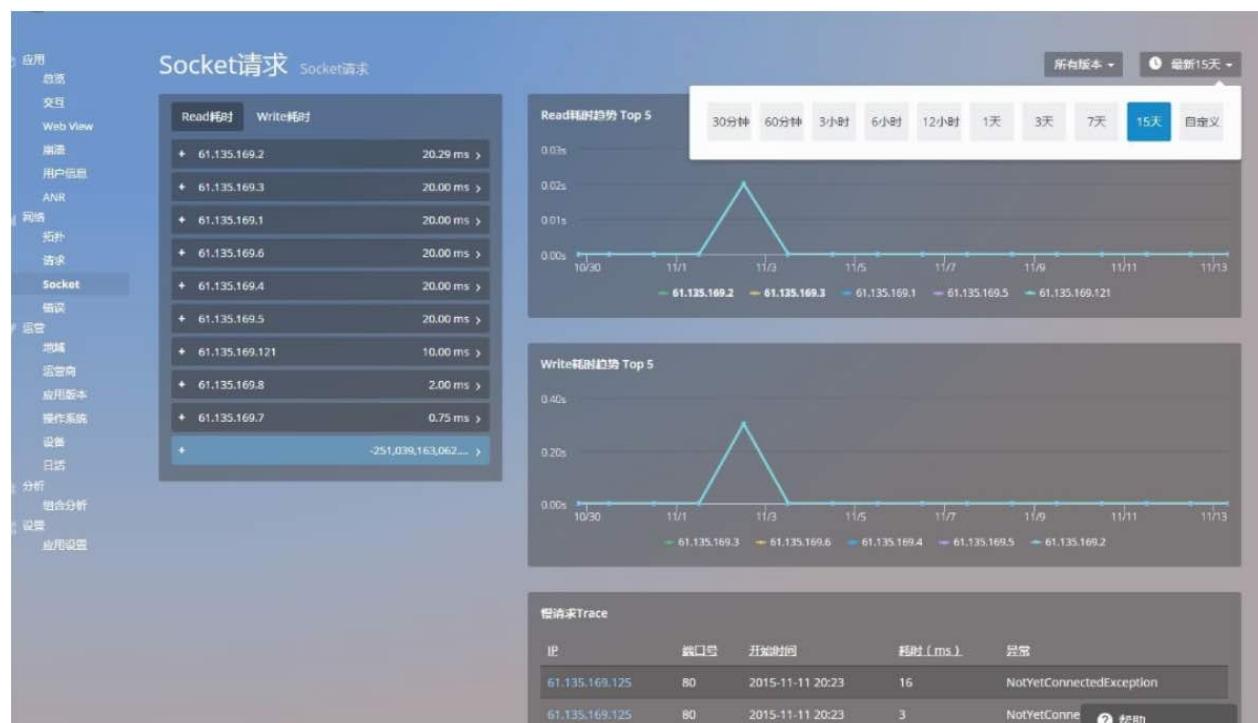
## SocketChannel

具体实现如下：

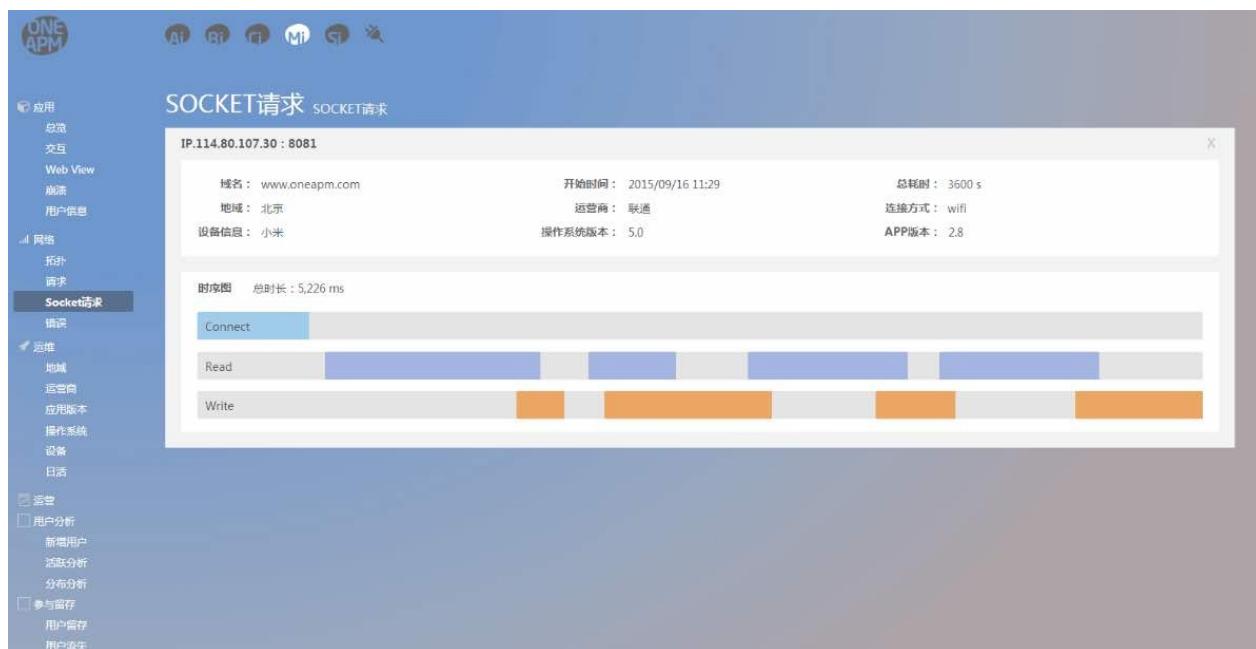
```
SocketChannel socket = SocketChannel.open();
SocketChannelInstrumentation socketChannelInstrumentation1 =
 new SocketChannelInstrumentation(socket);
socketChannelInstrumentation1.connect(
 new InetSocketAddress("127.0.0.1", 12345));
socketChannelInstrumentation1.read(readBuffer);
socketChannelInstrumentation1.write(pendingData.get(0));

...
socketChannelInstrumentation1.finishConnect();
```

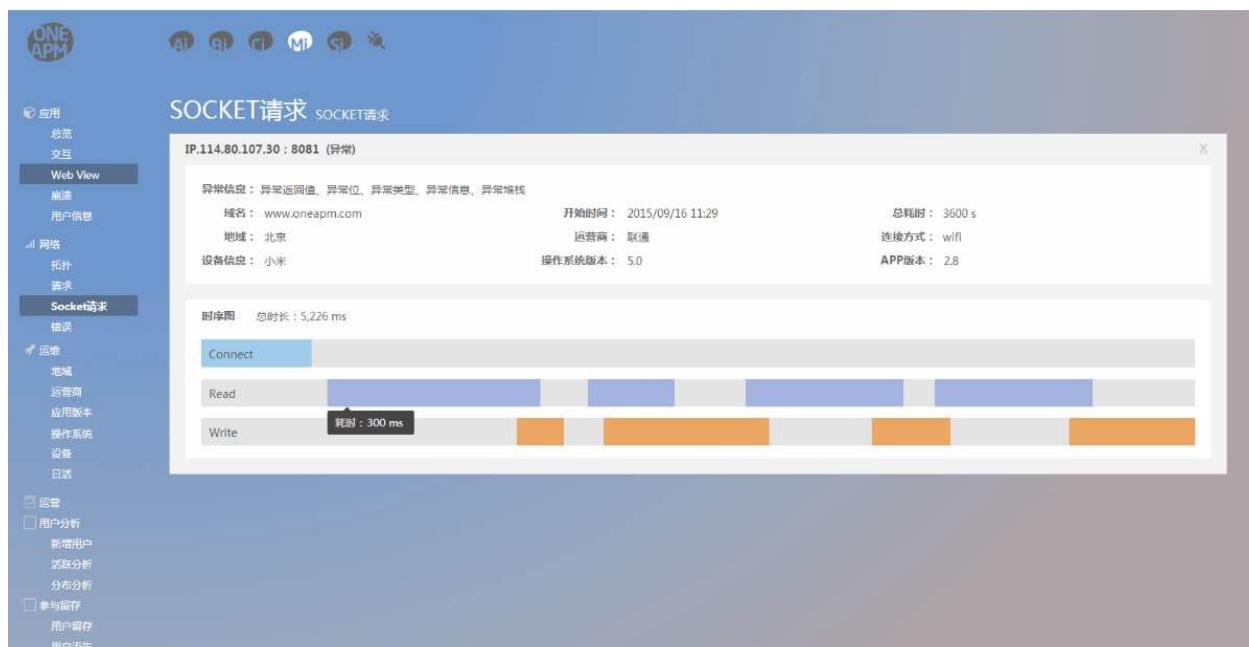
这样就可以进行操作了，socket监控程序能够正确的运行并收集您程序中存在的问题，可以直观展示：Read耗时趋势Top5，Write耗时趋势Top5，右上方鼠标悬停可以添加至仪表盘，在耗时曲线上鼠标悬停可以看到某一天的耗时时间。



点击列表中任意无异常数据，查看详情：



点击异常信息，查看详情：



## 4. 错误

「错误」功能展示的是在App上发生的HTTP错误和网络故障，该界面包含以下内容：

- 错误统计列表：可以按照域名、状态码、故障类型三种方式对错误进行筛选排序。
- HTTP 错误率：HTTP 错误率最高的五个域名。
- 网络故障率：网络故障率最高的五个域名。
- 错误详情：通过 URL名称和HTTP状态码识别为一个错误，并展示错误首次发生时间、最后发生时间、应用版本号以及发生次数，点击错误查看详情。



## 错误详情

详情页面展示了该网络错误次数的时间曲线、错误影响区域的占比、错误影响运营商的占比和错误影响接入方式的占比。

- 错误次数的时间曲线能告诉用户网络错误的趋势，在哪个时段最容易出现错误。
- 地域、运营商和接入方式均是与网络密切相关的指标。
- 地域占比能告诉用户应该特别注意哪个地区的网络性能。
- 运营商占比反映了哪家运营商的服务对自己的应用最友好。

如图所示，这是用户连接 WIFI 使用 App 时，发生了 HTTP 错误。此外，HTTP 错误归类滚动追踪轴，可以查看单次 HTTP 错误的详情。当 App 上线后，短时间内的 HTTP 错误可能多达上百条，不易查看。将同类信息合并，可以快速发现关键问题。而需要查看细节时，也可以滚动追踪轴，查看每条记录。



关键词：拓扑 请求 Socket 错误



# 运营

App 上线后，通过OneAPM丰富的运营数据，用户可以定制更加智能的应用优化策略，该模块包括以下功能：

- 地域
- 运营商
- 应用版本
- 操作系统
- 设备
- 日活

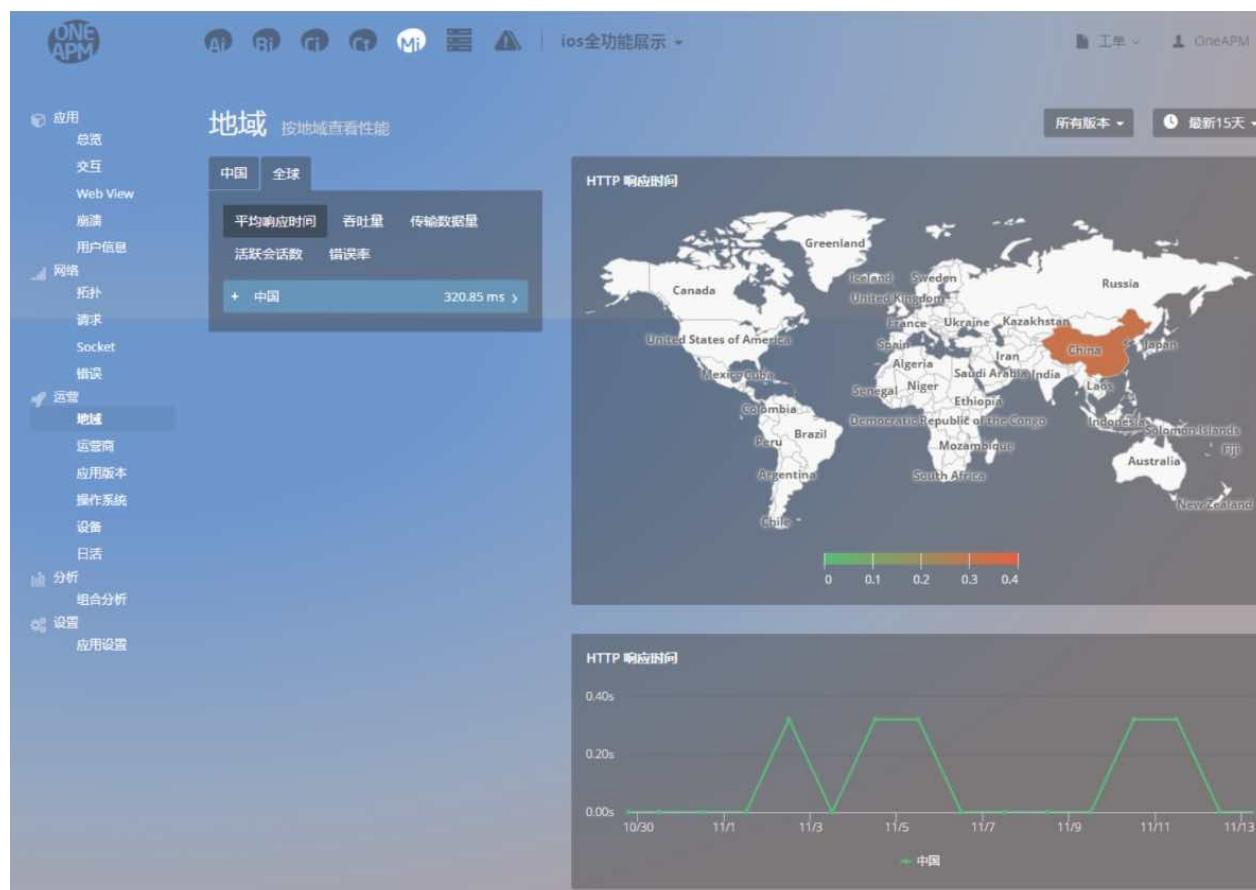
## 1.地域

该功能通过「地域」这个维度，显示App在全国以及全球不同地域的性能表现。

- 列表：可通过平均响应时间，吞吐量，传输数据量，活跃会话数，错误率选项对列表进行重新排序。点击一个具体的省市，可以查看对应的性能指标。
- HTTP 响应时间地图：以地图的方式显示 App 在全国不同省市的 HTTP 响应时间。
- HTTP 响应时间折线图：以折线图的方式显示 App 在全国不同省市的 HTTP 响应时间。



点击的“全球”选框卡，您可以查看应用在世界范围内的运行情况。



## 2. 运营商

在这个界面，您可以查看 App 接入网络的不同方式，比如：WIFI、移动 3G、联通 4G 等方式。

- 运营商列表：可按照平均响应时间、错误率、活跃会话数进行重新排序。
- HTTP 响应时间：查看 App 在以不同接入方式接入网络时的 HTTP 响应时间。
- 网络故障率：显示某一运营商在此时间范围内的错误率、错误次数、错误总次数。
- 活跃会话数：显示某一运营商在此时间范围内的活跃会话数。在运营商列表中，选择具体一个运营商可以查看对应的性能数据。



## 3. 应用版本

在发布多个版本的App后，您可以在「应用版本」这个功能查看不同版本 App 的交互时间和活跃会话。在应用版本列表中，选择具体一个版本可以查看对应的性能数据。



## 4. 操作系统

当App的用户量不断增加后，您可以通过该功能了解用户使用的操作系统分布情况，也可以通过该功能了解App在不同操作系统上的性能表现。提供的性能数据有：

- 执行时间：页面平均执行时间。
- HTTP 响应时间：发送HTTP请求开始,到收到所有响应内容的时间。
- 网络错误率：网络错误数量与请求数量的比率。其中发生网络错误的HTTP请求数指发生DNS解析错误、无法建连、连接超时等网络方面的错误的数量。
- 活跃会话数：用户交互过程中每个App终端每分钟与服务器的连接次数。



## 5. 设备

随着 App 用户量不断增加，您可以通过该功能了解用户使用的设备分布情况，也可以通过该功能了解App在不同设备上的性能表现。提供的性能数据有：

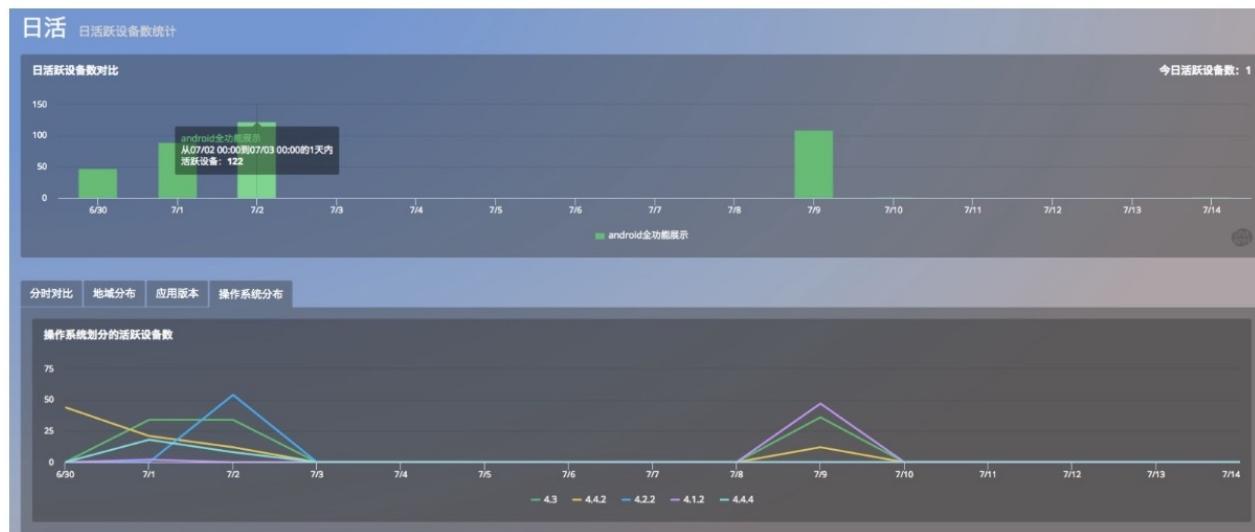
- 执行时间：页面平均执行时间。
- HTTP 响应时间：发送HTTP请求开始,到收到所有响应内容的时间。
- 网络故障率：网络错误数量与请求数量的比率。其中发生网络错误的HTTP请求数指发生DNS解析错误、无法建连、连接超时等网络方面的错误的数量。
- 活跃会话数：用户交互过程中每个App终端每分钟与服务器的连接次数。



## 6. 日活

「日活」功能显示15天的每日活跃数据，方便您轻松对比15天的 App 运营数据。

- 日活跃设备数对比：展示15天内,每日活跃设备数对比统计图。
- 分时对比：对比同一段时间，今日，昨日，7天前,15天前活跃设备数。
- 地域分布：展示活跃设备数TOP5地域随时间的推移的活跃设备数的对比趋势信息。
- 应用版本：按 App 版本，统计 15 天内活 跃用户数趋势与对比。
- 操作系统分布：按照操作系统，统计15天内活跃用户数趋势。



关键词：运营 日活 操作系统 地域

# 分析

Mobile Insight 的分析模块主要提供组合分析功能。可灵活设置各种指标和维度并可进行任意筛选，生成统计视图，帮助用户从多个角度和方位定位性能问题。

分类：

- 崩溃组合分析
- 网络组合分析

## 1. 崩溃组合分析

如下表所示，崩溃的组合分析以崩溃次数为首要指标，支持从多个维度进行对比分析，亦可以按照多个条件对信息进行筛选，达到更高的细粒度。

指标	维度	筛选条件
崩溃次数	App 版本，设备型号，操作系统版本，崩溃类型	App 版本，操作系统版本，崩溃类型，用户信息

## 2. 网络的组合分析

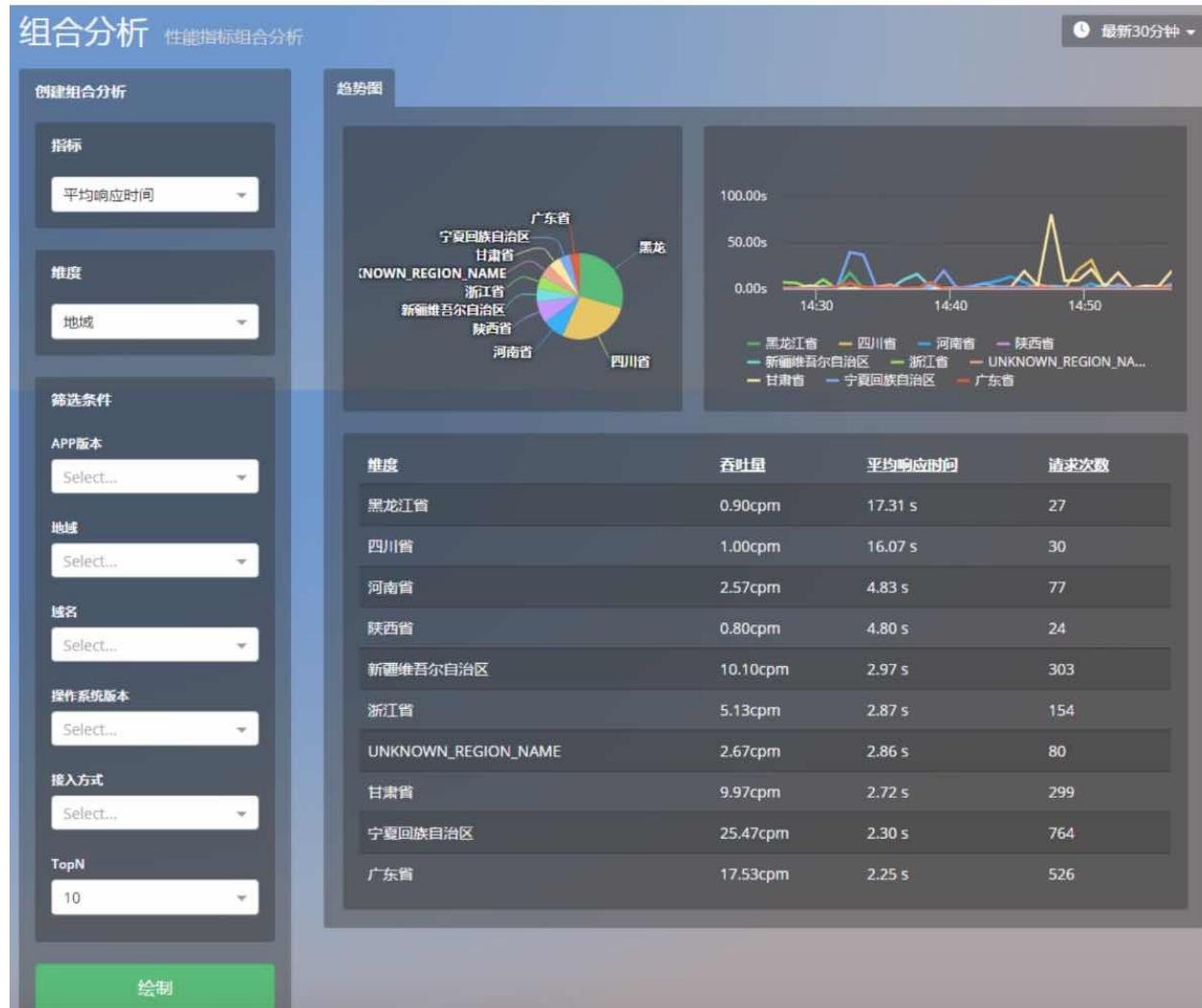
如下表所示，网络的组合分析包括HTTP错误数，HTTP错误率，网络错误数，网络错误率，响应时间，流量，吞吐量七个方面，每个方面都支持从多个维度进行对比分析，再按照指定的筛选条件对选定的维度进行更细致的筛选。

注意，维度仅支持单选，筛选条件可以多选。

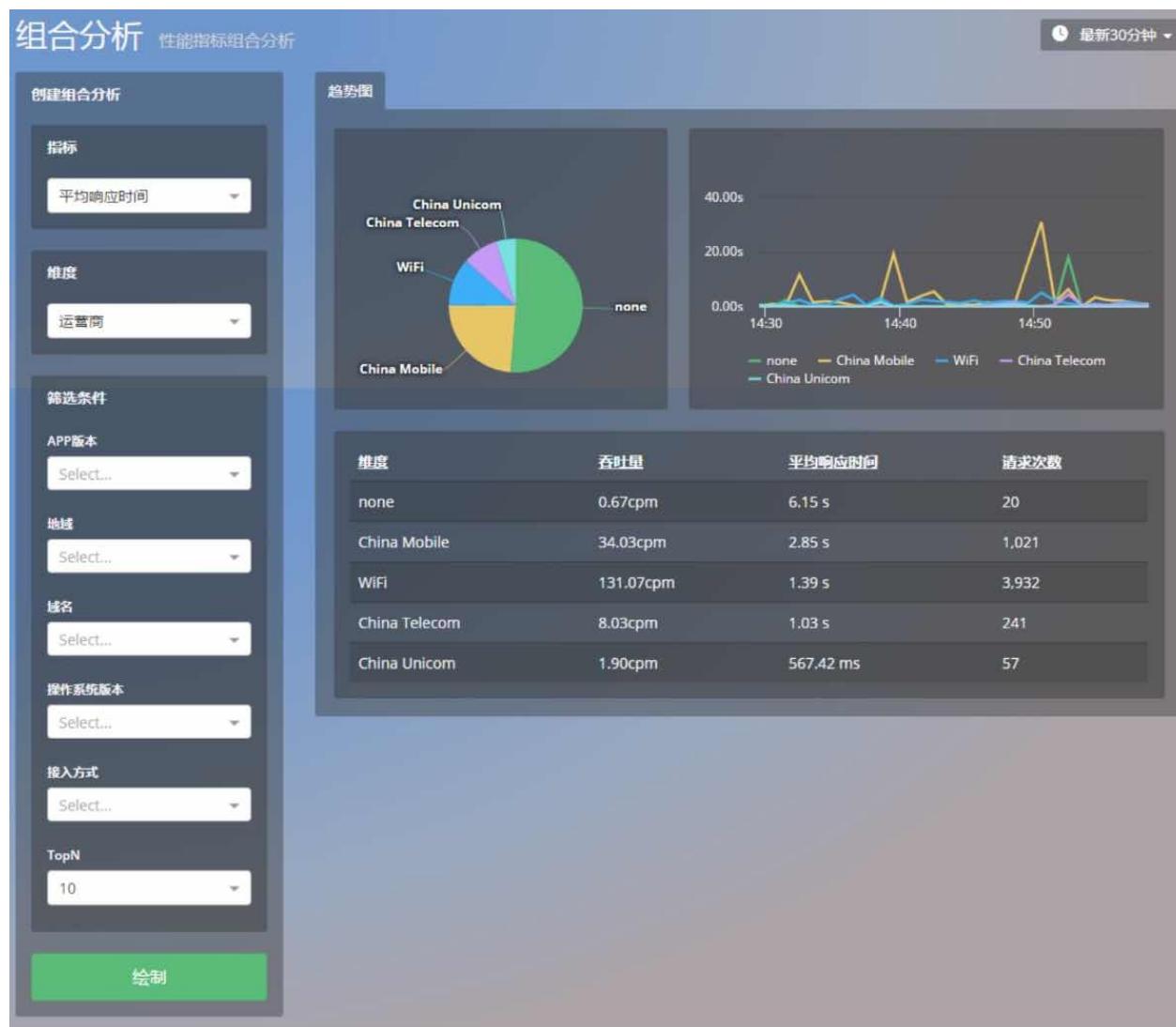
指标	维度	筛选条件
HTTP 错误数	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式, 状态码	App 版本, 地域, 状态码, 域名, 操作系统版本, 运营商, 接入方式
HTTP 错误率	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式, 状态码	App 版本, 地域, 状态码, 域名, 操作系统版本, 运营商, 接入方式
网络错误数	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式, 故障类型	App 版本, 地域, 故障类型, 域名, 操作系统版本, 运营商, 接入方式
网络错误率	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式, 故障类型	App 版本, 地域, 故障类型, 域名, 操作系统版本, 运营商, 接入方式
响应时间	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式	App 版本, 地域, 域名, 操作系统版本, 运营商, 接入方式
流量	APP 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式	App 版本, 地域, 域名, 操作系统版本, 运营商, 接入方式
吞吐量	App 版本, 地域, 域名, 操作系统版本, 设备厂商, 运营商, 接入方式	App 版本, 地域, 域名, 操作系统版本, 运营商, 接入方式

### 3. 如何诊断运营商在不同地域的网络性能

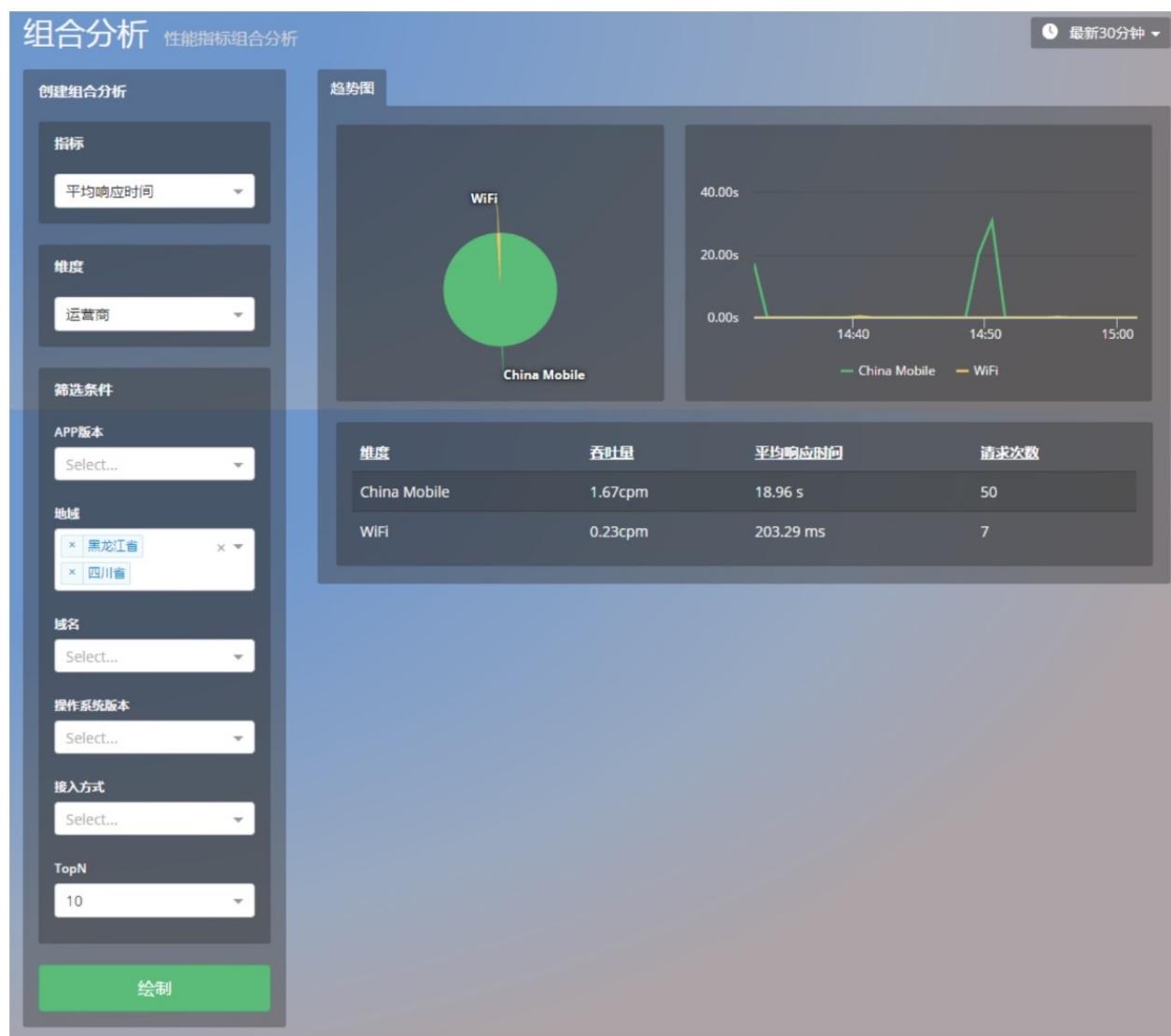
在以往，我们使用移动应用的「地域」功能，仅仅只可以查看各个地域的 HTTP 响应时间及排名。如果使用「运营商」功能，还可以查看各个运营商的 HTTP 响应时间以及排名。不过，此前这两个功能是各自独立的，并无什么关联，所以提供的信息过于粗放，不利于分析问题。如果想了解各个运营商在不同地域的网络性能，可以使用「组合分析」功能。首先，在组合分析页面，选择您关心的网络性能指标，譬如平均响应时间，维度选定为地域，点击绘制，就会出现下图：



通过视图我们可以非常直观的发现，来自黑龙江省和四川省的网络请求平均耗时最长。接下来，您可以将维度改选为运营商，点击绘制，就会出现下图：



该图展示了各个运营商的网络性能。接下来，在筛选条件栏目下，我们点击地域，选择网络性能最差的两个地区：黑龙江省和四川省，点击绘制，出现下图：



显而易见，应用在黑蜀两地的响应时间如此之慢，运营商 China Mobile 有着重大的责任。您也可以举一反三，排列组合我们提供的分析维度和筛选条件，更加细致地了解应用在不同条件下的网络性能。

关键词：分析 组合分析 崩溃分析 网络分析

# 设置

## 1. 目录

- 基本信息设置
- 参数设置

## 2. 基本信息设置

该部分提供修改应用基本信息的入口，还允许用户上传反混淆文件，优化探针的数据分析准确度。关于反混淆文件：Android应用发布时，为了防止反编译，会进行混淆，导致探针抓取的数据是混淆后的数据，可读性不高。为此，本次更新开放一个入口，允许用户上传反混淆文件，用以翻译探针抓取的数据，以提升其可读性。



点击设置一栏下面的应用信息，便可上传该文件。



### 3. 参数设置

该部分提供交互 trace、网络错误 trace、网络请求 trace、崩溃 trace 的采集开关，您可以自主选择是否采集这些数据。默认情况下，所有数据均采集。此外，您还可以选择交互 trace 的采集阈值。该默认值为 0.4 秒，即当交互执行时间超过 0.4 秒时，将视为慢交互予以记录。

注：目前该功能只支持 2.0.2 版本以上的 Android 探针。



关键词：设置 参数 反编译

# 自定义仪表盘

借助该功能，您可以个性化创建仪表盘，并为其添加想要监控的动态图，从而更有针对性地监控应用性能变化。

## 目录

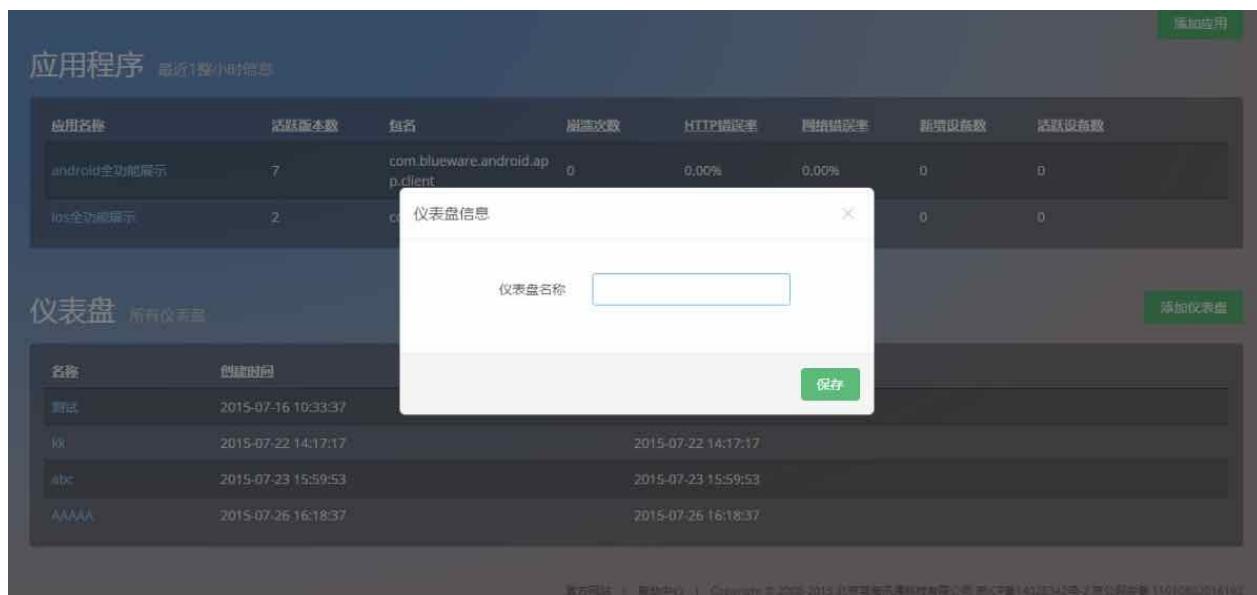
- 创建自定义仪表盘
- 为仪表盘添加动态图
- 删除仪表盘
- 删除动态图

## 创建自定义仪表盘

在 Mobile Insight 首页，点击“添加仪表盘”按钮，可直接创建仪表盘。



如图，输入名称，点击“保存”即可。注，每个账户至多可添加五个仪表盘。



## 为仪表盘添加动态图

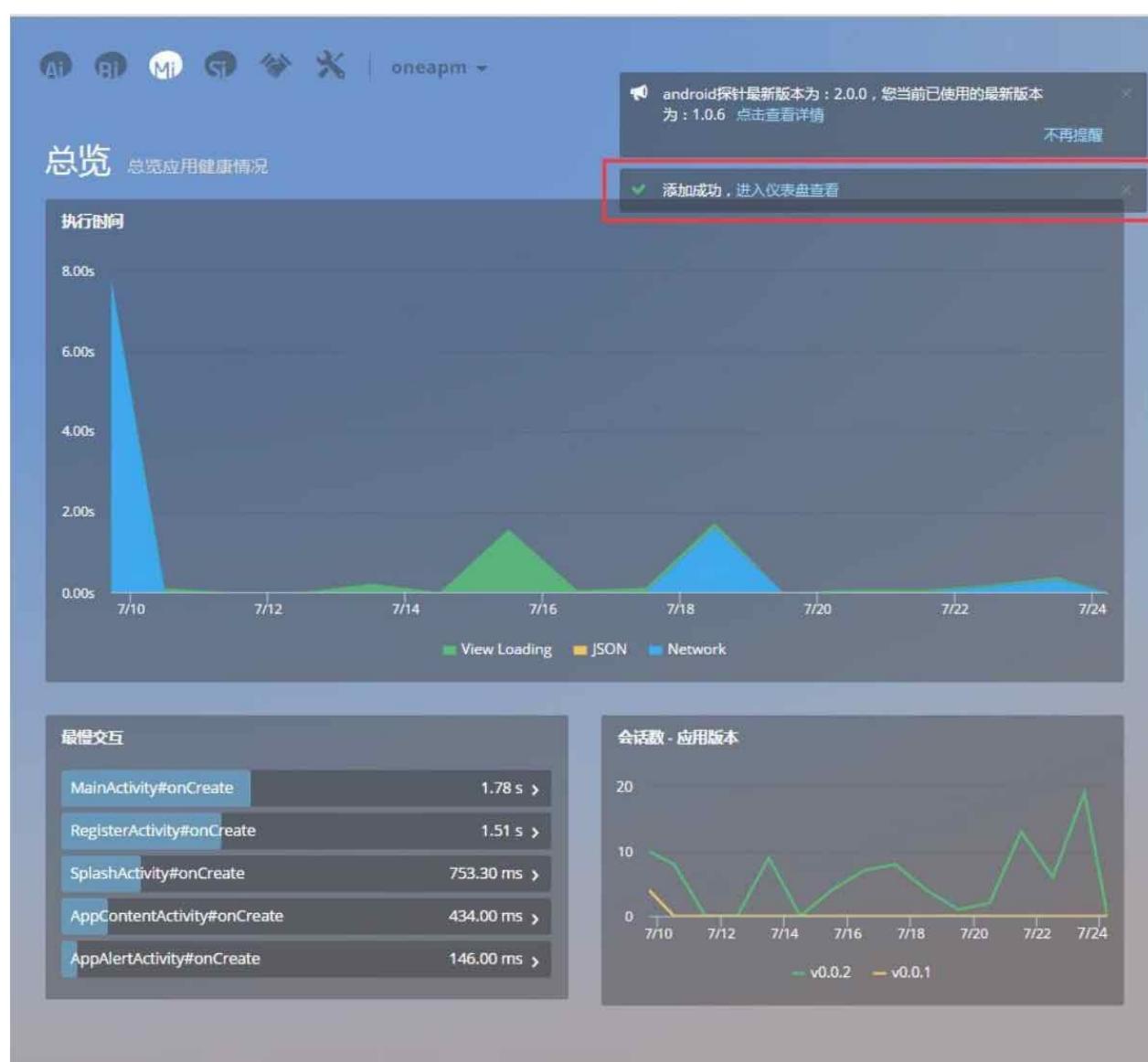
在应用功能界面，鼠标悬浮在想要添加至仪表盘的动态图右上角，会出现仪表盘图标。



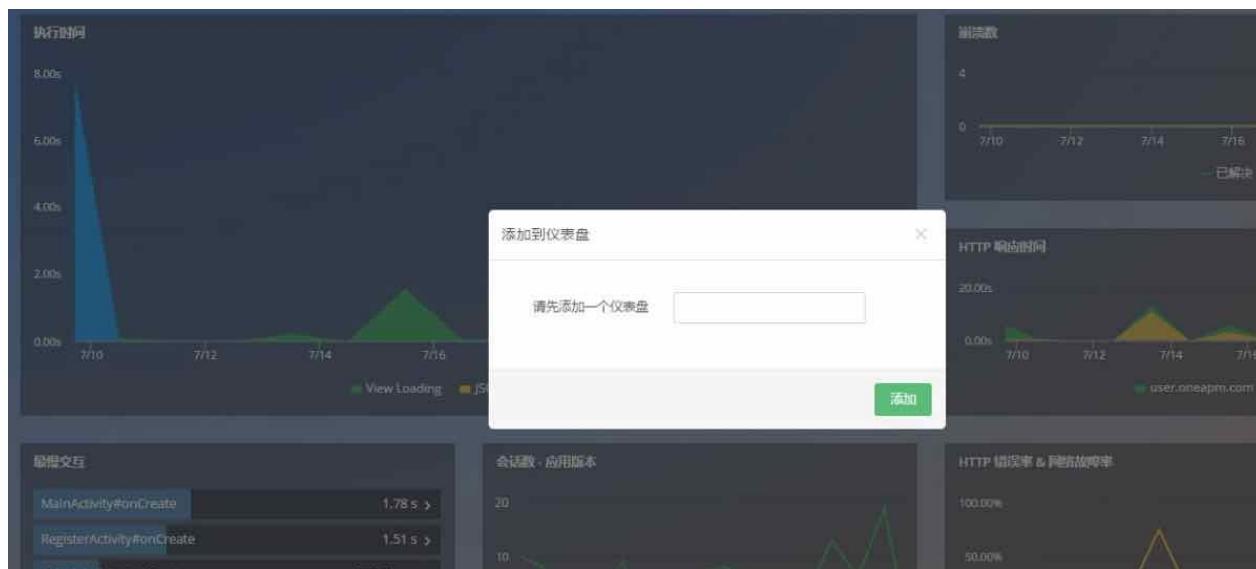
点击之后，选择添加的仪表盘，修改名称后点击添加即可。注，每个仪表盘至多可添加八张动态图。



添加完成之后，系统提示添加成功，并给出跳转至相关仪表盘的链接。



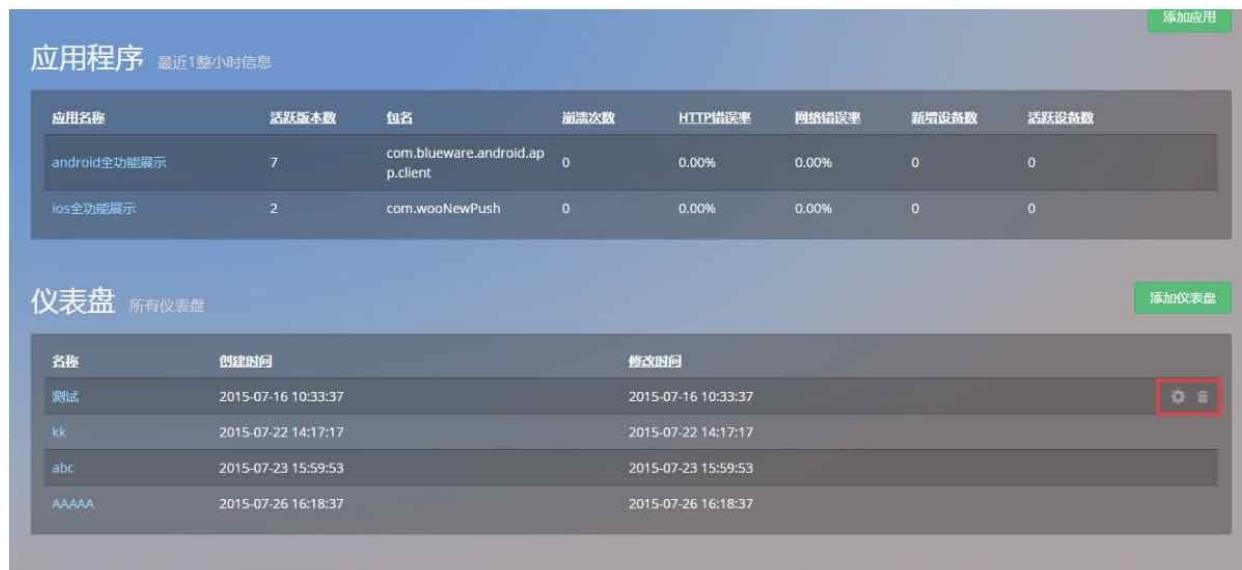
若此时没有创建好的仪表盘，系统会提示先添加一个仪表盘。



添加成功之后，后续操作不变。

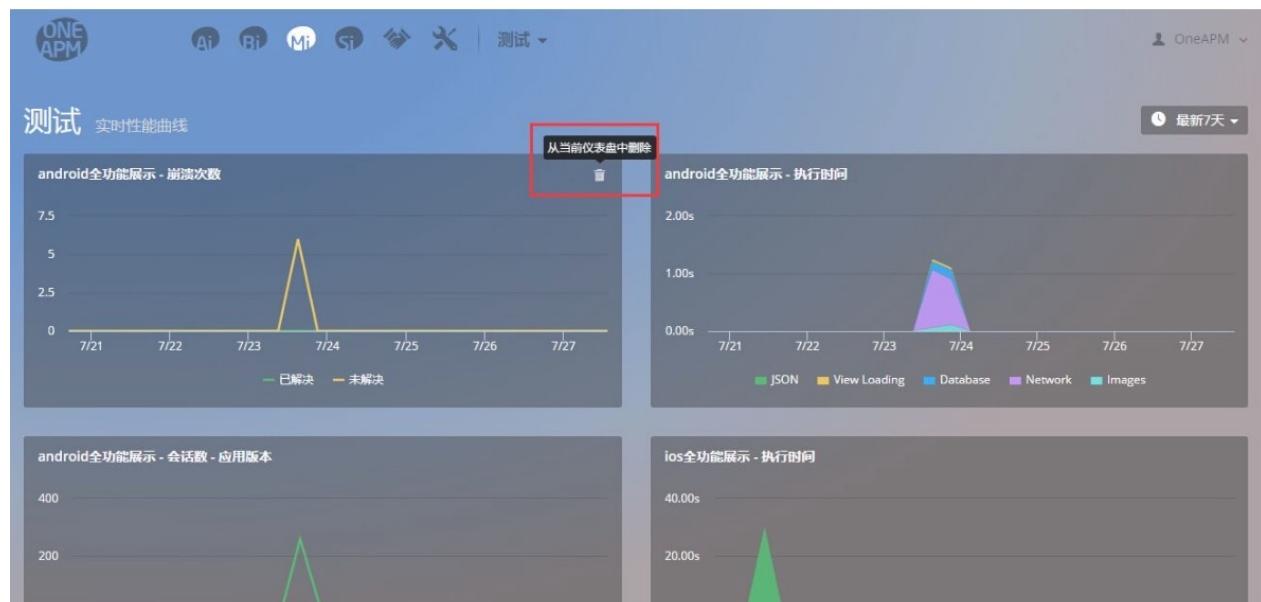
## 删除仪表盘

如图，鼠标悬浮在仪表盘列表右侧，出现修改仪表盘名称或删除仪表盘符号。



## 删除动态图

如图，鼠标悬浮在动态图右上角，出现删除动态图符号。



关键词：自定义仪表盘

# ANR 数据抓取及展示

ANR，全称 Application Not Responding 或 The Application of Non Response，意思是“应用无响应”。本文主要介绍 ANR 产生原因以及 OneAPM Mobile Insight 提供的 ANR 数据抓取及展示功能。引起 ANR 问题的根本原因，可以大致归为两类：

- 应用进程自身引起 例如：主线程阻塞、挂起、死循环。应用进程的其他线程 CPU 占用率过高，导致主线程无法抢占 CPU 时间片等。
- 其他进程间接引起 例如：当前应用进程与其他进程开展通信请求，其他进程长时间没有反馈。或者其他进程 CPU 占用率过高，导致当前应用进程无法抢占 CPU 时间片。

ANR 对于应用的影响并不亚于 Crash。那我们如何监控 ANR 呢？

OneAPM Mobile Insight 提供了 ANR 监控功能：



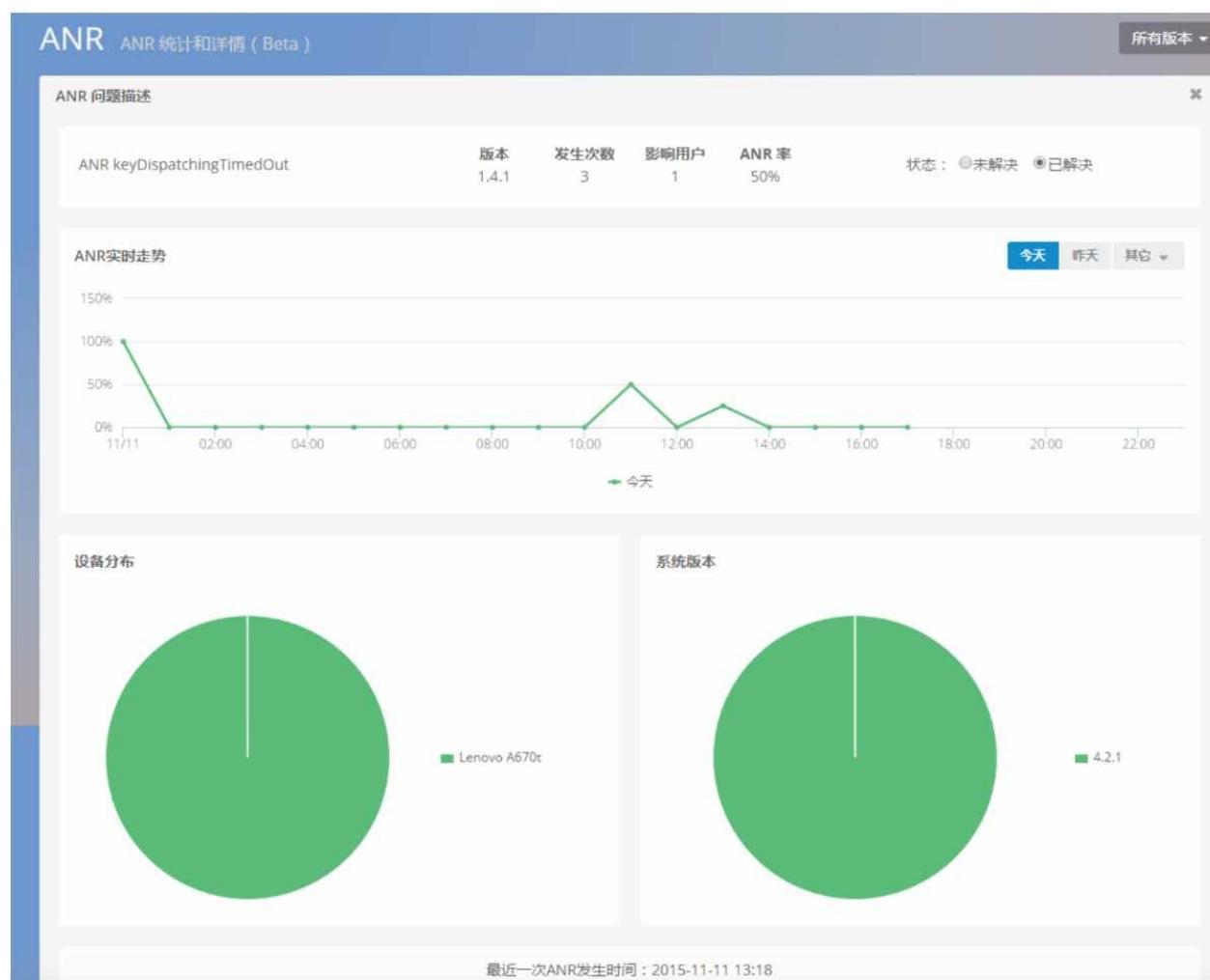
ANR 统计包括：ANR 趋势信息以及 ANR 率。默认情况下展示 ANR 当天趋势信息，若鼠标悬停，则展示此时 ANR 率。ANR 率 = ANR 影响用户数 / 活跃用户数。



ANR 类型列表：按 APP 版本与 ANR 类型进行分类展示 ANR 问题，展示某类 ANR 所影响的用户数，以此表征问题严重程度。

问题描述	APP 版本	发生次数	影响用户数	首次发生时间	最后发生时间	状态
ANR keyDispatchingTimedOut	1.4.1	3	1	2015/11/11 00:16:08	2015/11/11 13:18:43	已解决

点击列表中问题描述，查看 ANR 信息详情：查看某版本上该类 ANR 问题分布设备与操作系统版本的比例，以及 ANR 发生趋势，帮助您从宏观上控制 ANR 影响范围。



ANR 线程等信息记录，帮助用户定位造成 ANR 的原因。

(<) 第 1/3 个同类 ANR (>)

包名 : com.example.app.demo  
设备信息 : Lenovo A670t  
可用内存 : 63.06 MB  
渠道信息 : lanhaixuntong

ANR 部件 : .AnrActivity  
操作系统 : Android 4.2.1  
可用磁盘 : 31.64 MB  
SDK 信息 : android 2.0.3

ROOT : 有  
可用 SD 卡 : 1.03 GB

ANR 线程 其它线程 ANR Trace 文件 ANR Message

**com.example.app.demo/.AnrActivity**

```
at java.lang.VMThread.sleep(Native Method)
at java.lang.Thread.sleep(Thread.java:1010)
at java.lang.Thread.sleep(Thread.java:992)
at oneapm.demo.anr.TestAnrReceiver.onReceive(TestAnrReceiver.java:18)
at android.app.LoadedApk$ReceiverDispatcher$Args.run(LoadedApk.java:788)
at android.os.Handler.handleCallback(Handler.java:725)
at android.os.Handler.dispatchMessage(Handler.java:92)
at android.os.Looper.loop(Looper.java:153)
at android.app.ActivityThread.main(ActivityThread.java:5297)
at java.lang.reflect.Method.invokeNative(Native Method)
at java.lang.reflect.Method.invoke(Method.java:511)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:833)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:600)
at dalvik.system.NativeStart.main(Native Method)
```

# UI性能-卡顿

在手机App竞争越来越激烈的今天，App的各项性能，如CPU、内存消耗等都有了客观的衡量指标。但对于App使用过程是否流畅，一直没有一个可靠的指标将用户的客观感受和数据一一对应，Mobile Insight的卡顿可以直观地展示这些信息。

- 卡顿趋势图

流畅度即1s中之内绘图刷新信号中断的次数。次数越接近40时，用户能感知到卡顿，帧率在20以下卡顿比较严重。我们采集流畅度小于40的数据绘制成曲线图，以此预测卡顿趋势状况。



- 卡顿页面

卡顿页面包括页面名称、平均流畅度、最小流畅度、发生次数以及最近发生时间.

默认为线程执行时间超过0.4s且流畅度小于40，才会采集卡顿数据

卡顿 卡顿页面概况 ( Beta )

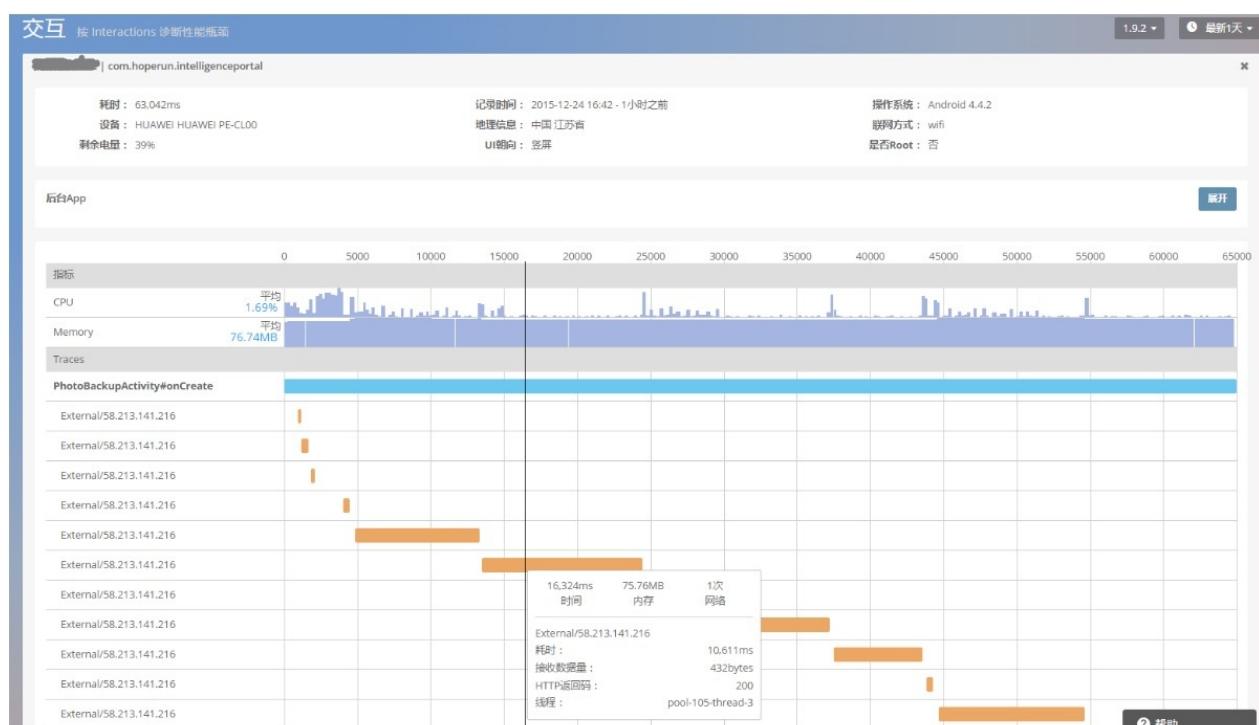
MainActivity#onCreate 至 2015-12-24 18:16:52 12次 卡顿

返回时间	流畅度(/s)	耗时(ms)	设备型号	APP 版本	系统版本	CPU 利用率
2015-12-24 18:16:52	31	1573	HUAWEI PLK-TL01H	1.9.2	5.0.2	
2015-12-24 17:49:16	6	60902	samsung SM-J7008	1.9.2	5.1	
2015-12-24 15:40:43	24	4618	ZTE X9180	1.9.2	4.4.2	
2015-12-24 14:52:27	22	9103	Xiaomi 2014501	1.9.2	4.4.2	
2015-12-24 13:47:05	23	16674	samsung SM-G9280	1.9.2	5.1.1	
2015-12-24 13:47:05	22	7674	samsung SM-G9280	1.9.2	5.1.1	
2015-12-24 13:23:47	25	1223	HUAWEI HUAWEI P6-C00	1.9.2	4.2.2	
2015-12-24 06:35:50	10	60827	Xiaomi 2014813	1.9.2	4.4.4	
2015-12-24 02:01:33	33	2639	HUAWEI HUAWEI MT7-CL00	1.9.2	4.4.2	
2015-12-23 22:37:33	22	59975	smartisan YQ601	1.9.2	4.4.4	

[显示更多](#)

卡顿详情列表展示：访问时间，发生卡顿时的流畅度，耗时，发生卡顿时的设备信息，APP版本，操作系统及版本，CPU信息

通过分析该页面信息可以清楚了解到卡顿来源，以便针对性优化



点击卡顿详情页的单条数据跳转至对应交互时序图，定位卡顿根因。

关键词：卡顿 页面加载慢 CPU

# 用户分析

- 新增用户
- 活跃分析

## 新增用户

按照版本号、渠道和日期三个筛选条件进行交叉筛选绘制新增用户趋势图，方便您从整体掌控应用的运营情况及增长动态。



新增用户为第一次启动应用的用户（以设备为判断标准），按日，周，月查看数据并且可进行版本，渠道的交叉筛选。

对比时段，对比你所关心时间的数据：点击新增用户趋势报表的“对比时段”可对所选时段的数据做上周同期以及上月同期数据对比。

新增用户趋势

日期	新增用户	新增用户占比
2016-01-25	0	0.00%
2016-01-24	0	0.00%
2016-01-23	0	0.00%
2016-01-22	0	0.00%

新增用户明细：展示所选日期范围内新增用户与新增用户占比信息

新增用户占比：某时段内新增用户占该时段活跃用户的比例

根据需要您可以点击导出按钮导出报表

## 活跃分析

按照版本号、渠道和日期三个筛选条件进行交叉筛选绘制活跃用户趋势图，活跃用户是指当日有使用应用（至少启动一次）的用户数。

活跃用户趋势

日期	活跃用户	活跃用户占比
2016-01-25	1	4.00%
2016-01-24	1	4.00%
2016-01-23	1	4.00%
2016-01-22	1	4.00%
2016-01-21	2	8.00%

活跃时段分析呈现的是在选定时间段内总的用户启动次数，以及活跃用户数，分别所占的时间区间段。在此可以清楚的了解到用户使用应用时间区间，在高峰时段我们建议您可以做一些推广活动，同时支持自选时间段数据比对。

高峰时段/低谷时段：根据选定的时间段，系统计算出繁忙时间段，低谷时间段，以及启动占比。便于用户判断，APP受用群活跃特征，针对此特征做相应的活动策划或制定相应策略。



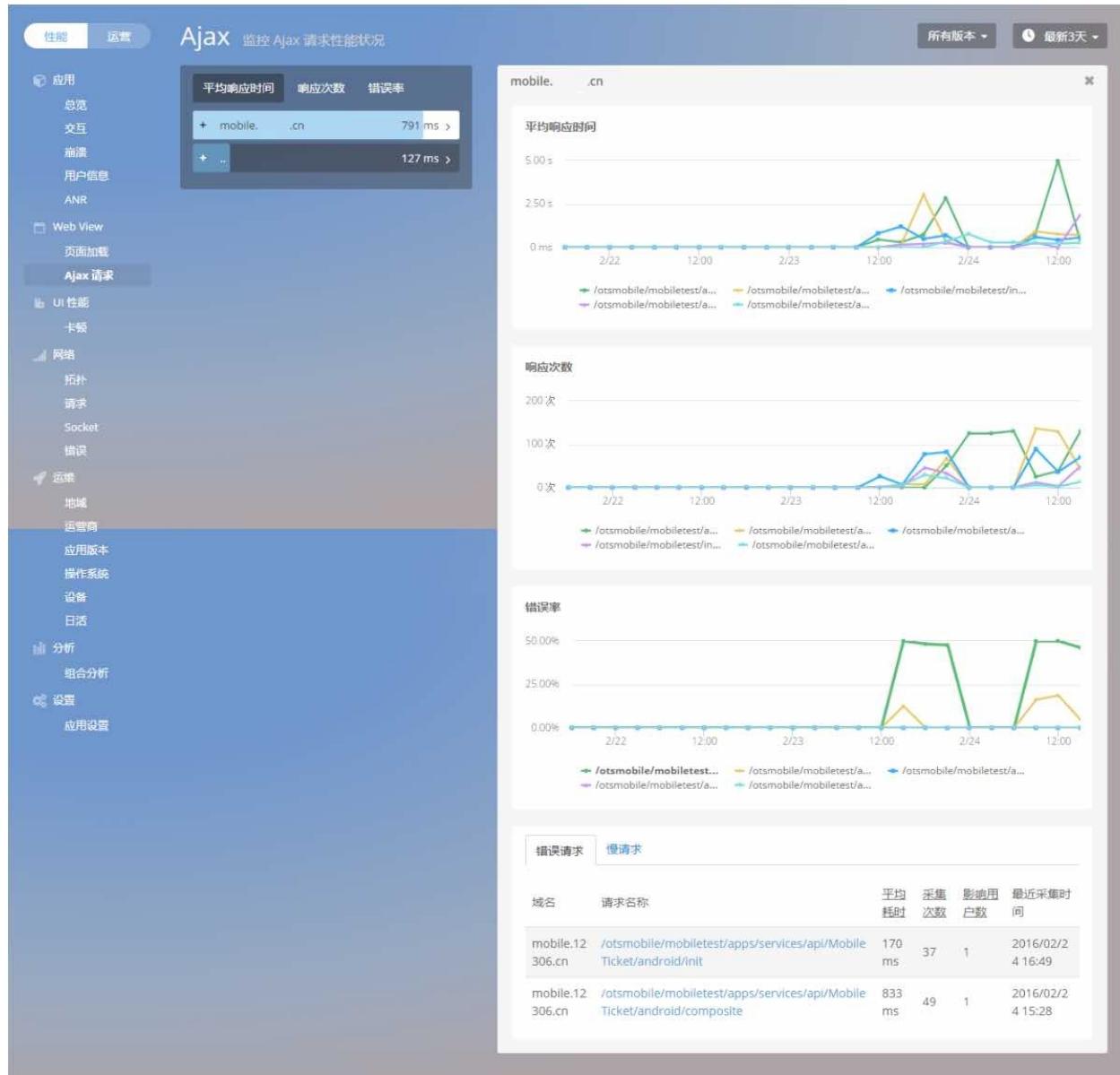
活跃用户明细：展示所选日期范围内活跃用户与活跃用户占比信息，其中活跃用户占比=活跃用户/所选时段内的活跃用户总数（不去重）。根据需要您可以点击导出按钮导出报表

关键词：活跃用户 用户分析

# Ajax请求

Webview->Ajax请求页

WebView 中Ajax请求，展现了Ajax请求的平均响应时间，响应次数，错误率的趋势图以及错误请求和慢请求追踪列表



## 1. 平均响应时间趋势信息图

平均响应时间趋势信息图展示响应时间TOP5请求（域名）的趋势信息的堆叠趋势图，便于查看APP总体平均响应时间趋势状况。鼠标悬停在请求曲线上则显示该段时间AJAX请求的详细信息。其中平均响应时间：单位秒，平均每次AJAX请求的响应时间

左侧列表选择单个域名：右侧图表展示该域名下TOP5请求平均响应时间趋势曲线对比图

左侧列表选择单个请求：右侧图表展示该请求平均响应时间的趋势曲线对比图



## 2. 响应次数趋势信息图

响应次数趋势信息图展示响应次数TOP5请求（域名）的趋势信息的堆叠趋势图，便于查看APP总体响应次数趋势状况。鼠标悬停在请求曲线上则显示该段时间AJAX请求的详细信息

左侧列表选择单个域名：右侧图表展示该域名下TOP5请求数量趋势曲线对比图

左侧列表选择单个请求：右侧图表展示该请求的响应次数趋势曲线对比图



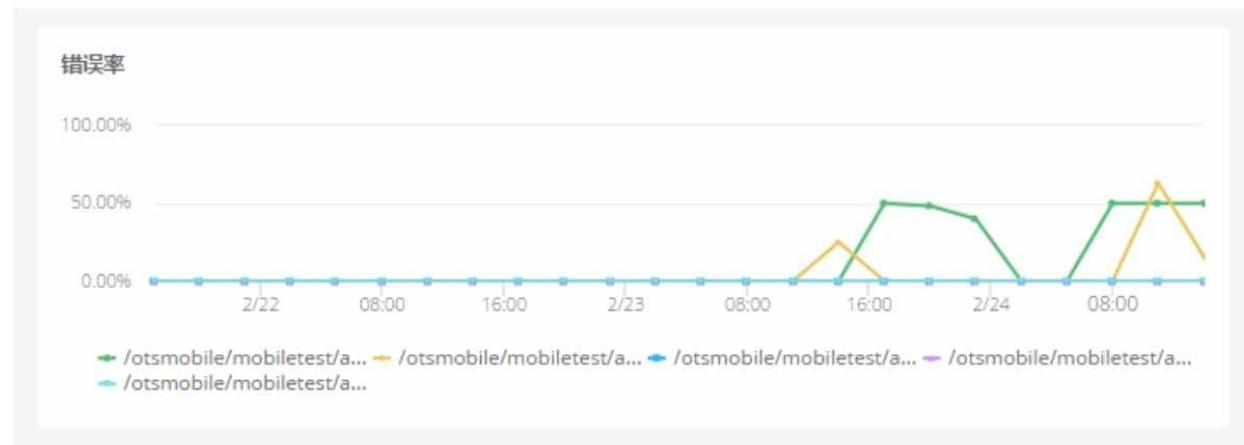
## 3. 错误率趋势信息图

错误率趋势信息图展示错误率TOP5请求（域名）的错误率对比趋势信息图，鼠标悬停在请求曲线上则显示该段时间AJAX请求错误的详细信息。

左侧列表选择单个域名：右侧图表展示该域名下TOP5请求数量趋势曲线对比图

左侧列表选择单个请求：右侧图表展示该请求的错误率趋势曲线对比图

Ajax请求 总错误率 = (所有Ajax请求) 总错误次数/总请求次数; 域名错误率= (该域名下的所有请求) 发生错误总次数/总请求次数; 单条请求错误率= (该请求) 发生错误总次数/总请求次数



## 4. Ajax 请求信息列表

Ajax请求信息列表分为错误请求列表与慢请求列表，默认展示：错误请求信息

错误请求列表内展示：域名，请求名称，平均耗时，采集次数，影响用户数，最近采集时间

错误请求	慢请求																		
<table border="1"> <thead> <tr> <th>域名</th> <th>请求名称</th> <th>平均耗时</th> <th>采集次数</th> <th>影响用户数</th> <th>最近采集时间</th> </tr> </thead> <tbody> <tr> <td>mobile.cn</td> <td>/otsmobile/mobiletest/apps/services/api/MobileTicket/android/init</td> <td>169 ms</td> <td>35</td> <td>1</td> <td>2016/02/24 15:19</td> </tr> <tr> <td>mobile.cn</td> <td>/otsmobile/mobiletest/apps/services/api/MobileTicket/android/composite</td> <td>833 ms</td> <td>49</td> <td>1</td> <td>2016/02/24 15:28</td> </tr> </tbody> </table>	域名	请求名称	平均耗时	采集次数	影响用户数	最近采集时间	mobile.cn	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/init	169 ms	35	1	2016/02/24 15:19	mobile.cn	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/composite	833 ms	49	1	2016/02/24 15:28	
域名	请求名称	平均耗时	采集次数	影响用户数	最近采集时间														
mobile.cn	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/init	169 ms	35	1	2016/02/24 15:19														
mobile.cn	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/composite	833 ms	49	1	2016/02/24 15:28														

点击Ajax错误请求信息列表中任意一条，进入Trace详情页：

The screenshot shows the Mobile Insight interface for monitoring Ajax requests. At the top, there's a toolbar with various icons and a user profile. Below it, a main header says "Ajax 监控 Ajax 请求性能状况". On the left, a sidebar has tabs for "平均响应时间", "响应次数", and "错误率", with "错误率" currently selected. It lists two items: "+ mobile... .cn" (791 ms) and "+ .." (127 ms). The main content area shows a detailed trace for a request to "/otsmobile/mobiletest/apps/services/api/MobileTicket/android/init". The trace details include:

- 请求类型:** 错误请求 **最近采集时间:** 2016-02-24 16:49 **采集次数:** 37 **平均响应时间:** 170 ms
- 状态码分布:** A chart showing the distribution of status codes, with 401 being the most prominent.
- 地域分布:** A chart showing the distribution of regions, with Beijing being the most prominent.
- 运营商分布:** A chart showing the distribution of operators, with WiFi being the most prominent.
- Ajax 请求详情:**
  - 请求开始时间:** 2016-02-24 16:50
  - 地域/运营商:** 中国北京市 / WiFi
  - 请求方式:** POST
  - headers:**

```
{
 "x-wl-app-version": "2.1",
 "Accept-Language": "zh_CH",
 "Content-type": "application/x-www-form-urlencoded; charset=UTF-8",
 "Accept": "text/javascript, text/html, application/xml, text/xml, */*",
 "X-Requested-With": "XMLHttpRequest",
 "x-wl-platform-version": "6.0.0"
}
```
  - 状态码:** 401
  - statusText:** Unauthorized
  - 响应的数据:** /\*-secure- {"challenges":{"wl\_deviceNoProvisioningRealm":{"token":"t6f8vii7potenh6cc566qirnns"}, "wl\_antiXSRFRealm":{"WL-Instance-Id":"nsn766juq997bukhjdaoo5n3t"}}}\*/
  - 响应的URL:**
  - 执行时间:** 126 ms

其中状态码分布、地域分布、运营商分布分别展示TOP10 分布占比 (TOP9+其它类型)

慢请求信息采集条件：Ajax请求耗时>0.65s，可在设置页面修改阈值。

域名	请求名称	平均耗时	采集次数	影响用户数	最近采集时间
mobile..n	/otsmobile/mobiletest/apps/services/reach	5.22 s	2	1	2016/02/24 15:19
mobile..n	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/hearbeat	3.89 s	21	1	2016/02/24 13:13
mobile..n	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/composite	1.58 s	103	1	2016/02/24 15:02
mobile..n	/otsmobile/mobiletest/invoke	1.01 s	31	1	2016/02/24 11:50
mobile..n	/otsmobile/mobiletest/apps/services/api/MobileTicket/android/query	2.84 s	180	1	2016/02/24 14:36

点击Ajax慢请求信息列表中任意一条，进入Trace详情页：

参数	值
请求开始时间	2016-02-24 15:19
地址/运营商	中国北京市 / WiFi
请求方式	GET
headers	{}
状态码	200
statusText	OK
响应的数据	...
响应的URL	...
执行时间	5.23 s

关键词：Ajax 慢请求 Webview

# 告警

告警与报警的区别？在SMAP应用监控系统中，告警和报警是有特殊含义的，并不是一个同义词。对所采集的数据做出监控健对象是否健康的判断后，会根据预先定义的5种方式生成告警记录。告警一般定义为：

- 严重
- 危险（含严重）
- 严重或未知
- 危险（含严重）或未知
- 未知 这些告警记录被送到订阅人时，就形成报警。简言之，告警是当性能指标超过阈值，系统记录的告警日志。报警是指将告警信息系统消息、邮件或短信等手段通知给相关负责人。

本次新增功能 1 报警策略总览 2 添加报警策略-报警项目：APP整体 3 报警指标：响应时间，崩溃率，HTTP错误率，网络错误率 4 报警策略：静态阈值设置

## 功能介绍

### 警报策略总览

警报策略							设置值得关注的报警策略	所有版本	最新1天
告警触发器列表							设置告警组	添加告警策略	
状态	名称	监控项	吞吐量	报警条件	创建人	操作			
OK	崩溃报警项01	APP	>=10	崩溃率：>0.3 响应时间：>500ms	aaa				
OK	崩溃报警项02	APP	>=10	HTTP错误率：>0.3 网络错误率：>0.3	aaa				
OK	崩溃报警项03	APP	>=10	崩溃率：>0.4 响应时间：>500ms	aaa				
WARNING	崩溃报警项02	APP	>=10	HTTP错误率：>0.3 网络错误率：>0.3	aaa				
OK	崩溃报警项03	APP	>=10	崩溃率：>0.4 响应时间：>500ms	aaa				
WARNING	崩溃报警项02	APP	>=10	HTTP错误率：>0.3 网络错误率：>0.3	aaa				
ALERT	崩溃报警项03	APP	>=10	崩溃率：>0.4 响应时间：>500ms	aaa				
OK	崩溃报警项02	APP	>=10	HTTP错误率：>0.3 网络错误率：>0.3	aaa				
OK	崩溃报警项03	APP	>=10	崩溃率：>0.4 响应时间：>500ms	aaa				
OK	崩溃报警项02	APP	>=10	HTTP错误率：>0.3 网络错误率：>0.3	aaa				

状态解释 报警ALERT：触发报警 告警WARN：触发告警 正常OK：没有达到报警条件 无数据NO DATA：监控的指标没有数据，此时不一定满足报警条件 禁用MUTED：不再提醒

## 添加报警策略

新建告警策略

策略名称: 策略名称 \*

监控项目: APP整体 网络服务 页面 关键元素

触发条件: 吞吐量  $\geq$  10 cpm

**响应时间**

触发条件: 吞吐量  $\geq$  10 cpm

警告阈值: 持续 5 分钟 大于 1000 ms

严重阈值: 持续 5 分钟 大于 2000 ms

分组方式:  总体  区域  区域和运营商

**预览**

2.00 s  
1000.00 ms  
0.00 ms

**http错误率**

触发条件: 吞吐量  $\geq$  10 cpm

**预览**

100.00%

保存

本次新增功能只有和APP整体相关的警报指标，有崩溃率，网络性能指标和交互性能指标包括：崩溃率，响应时间，HTTP错误率，网络错误率

如果需要设置多项告警，可以在下方选择新增阈值项，然后在新增的阈值中选择需要的告警项，最多添加4项，不可重复选择同一指标。

当您不在需要某个告警项时，可以单击右上方的删除按钮删除该告警项。中间区域您可以设置告警的各项阈值，完成设置后单击“确定”保存警报策略。 警报指标选项： 响应时间：App整体平均响应时间 崩溃率：App崩溃率（天平均），不支持分组统计 网络错误率：App整体网络错误率 HTTP错误率：App整体HTTP错误率

警报策略默认值： 响应时间 吞吐量  $\geq 100$  警告阈值： 持续 5分钟 大于1000 毫秒 严重阈值： 持续 5分钟 大于 2000 毫秒 吞吐量  $\geq 100$  例外 （即：10cpm  $\leq$  吞吐量  $\leq 100\text{cpm}$ ） 警告阈值： 持续 5分钟 大于 500 毫秒 严重阈值： 持续 5分钟 大于 2000 毫秒 崩溃率 吞吐量  $\geq 100$  警告阈值： 持续 5分钟 大于 2‰（千分之） 严重阈值： 持续5分钟 大于 10‰（千分之） HTTP错误率 吞吐量  $\geq 100$  警告阈值： 持续 5分钟 大于 3% 严重阈值： 持续 5分钟 大于 5% 网络错误率 吞吐量  $\geq 100$  警告阈值： 持续 5分钟 大于 3% 严重阈值： 持续 5分钟 大于 5%

## 设置告警组

告警触发器列表						
状态		名称	监控项	吞吐量	报警条件	创建人
ALERT	响应时间	APP	>= 1		网络错误率 : > 3%	SX
WARNING	网络故障率	APP	>= 1		网络错误率 : > 5%	SX
ALERT	12222222	APP	>= 1		崩溃率 : > 1% 响应时间 : > 2000 ms 网络错误率 : > 5% HTTP错误率 : > 5%	SX
ALERT	http错误率	APP	>= 1		HTTP错误率 : > 5%	SX
OK	崩溃率123	APP	>= 1		崩溃率 : > 1%	SX

点击红框中链接，跳转至OneAlert设置告警组

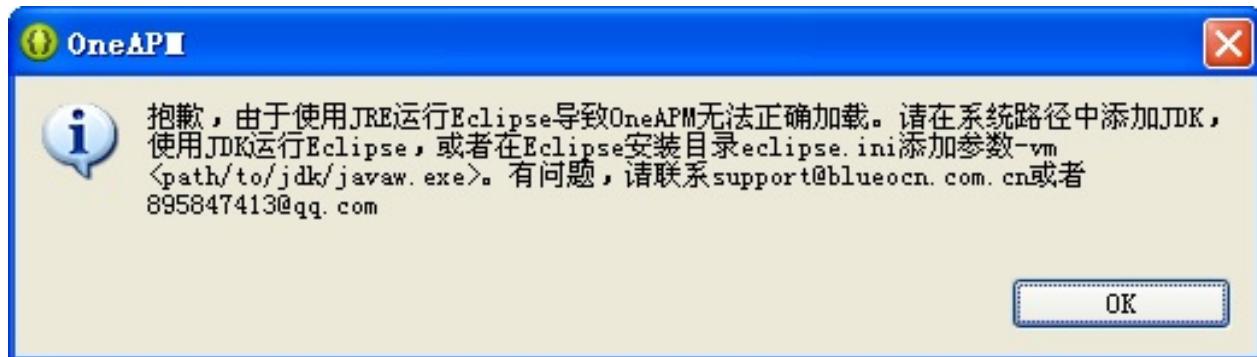
## 5.常见问题

# Android

# SDK 安装后提示错误信息：由于使用 JRE 运行 Eclipse 导致 OneAPM 无法正确加载

## 故障说明

Android SDK 通过 Eclipse 插件方法安装完成，重启后出现 OneAPM 启动异常



## 解决方案

- 第一步：检查 JAVA\_HOME 是否设置，如果设置了，将 JAVA\_HOME 设置在 PATH 环境变量的第一位；
- 第二步：找到 Eclipse 目录下的 eclipse.ini；
- 第三步：添加 vm 参数，将 -vm 添加到 openFile 参数之后；

```
-vm
<path>/bin/javaw.exe
```

注意：需要修改，设置成用户个人的 JDK 的路径，注意换行！

添加完之后 eclipse.ini 文件内容相对结构如下

```
openFile
--launcher.appendVmargs
-vm
C:/Program Files/Java/jdk1.6.0_43/bin/javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=1.6
-Xms40m
-Xmx512m
```

- 第四步：重启 Eclipse。

# Eclipse 安装 SDK 提示后提示找不到 content.xml 文件

Eclipse 安装 SDK 提示后提示找不到 content.xml 文件

问题回复：

首先，请检查安装的版本是否正确：

Eclipse 4.4 及之后版本请使用以下链接：

[https://download.oneapm.com/android\\_agent/eclipse\\_gt\\_4.4/](https://download.oneapm.com/android_agent/eclipse_gt_4.4/)

Eclipse 4.4 之前版本请使用以下链接：

[https://download.oneapm.com/android\\_agent/eclipse\\_lt\\_4.4/](https://download.oneapm.com/android_agent/eclipse_lt_4.4/)

如果还是不行，请检查本机是否使用了代理，如果使用了代理，关闭代理，再进行安装即可。

# 追踪用户信息接口使用说明文档

## 解读用户信息

有时候，发生了慢交互或者发生了 Crash 的时候，问题产生的原因多种多样，我们相对问题进一步分析如果没有详细的用户信息做支持会非常困难，为此我们引入了用户信息设置接口。有了这个接口，您可以在发送崩溃或者发生网络错误或者慢交互的时候能关联到您设置的用户信息，值得注意的是这个地方的用户信息是一个很抽象的概念，您可以把用户信息理解为用户名、邮箱、用户 ID，甚至你可以把它理解成设备标示，订单号等。一句话，用户信息的概念由您来定！

## 用法

在调用 Blueware.start() 接口的时候调用。

- 1、先定义一个 Hashmap 存放用户信息，设置一些信息放在 ContextConfig 中，这里设置的信息不建议太多，以避免发送数据量过大！

```
HashMap<String, String> extraData = new HashMap<String, String>();
extraData.put("电话", "189****6789");
extraData.put("ID", "123456");
extraData.put("邮箱", "aaaaa@oneapm.com");
ContextConfig contextConfig = new ContextConfig();
contextConfig.setExtra(extraData);
```

- 2、既然设置了那么多信息，怎么检索？设置一个检索字段吧！

```
// 搜索字段，例如“电话”或者“ID”
contextConfig.setSearchField("电话");
```

- 3、调用 withContextConfig 把用户信息设置到启动代码中

## 注意

2.0.3版本及之前的用户请安装如下操作：

```
BlueWare.withApplicationToken("---- You Token ----")
 .withContextConfig(contextConfig).start(this.getApplication());
```

2.0.4版本及之后的用户请安装如下操作：

```
BlueWare.withContextConfig(contextConfig);
OneApmAgent.init(this).setToken("---- You Token ----").start();
```

## 附加说明

可能这样写之后还是无法搜索到、看到数据，原因是没有发生Crash崩溃，所有探针没有上传用户的崩溃信息。

# Mac OS下 Android Studio 如何卸载探针?

很多人问在 Mac OS下 Android Studio 如何卸载探针?

因为Android Studio 为了提高编译的速度，加入了 daemon 的缓存机制，这个缓存可能是由于 bug，经常无法清理干净。为此，解决问题的方法也是手动去清理一下缓存。

- 1、Mac 环境的缓存路径是：

/Users/用户名/.gradle/daemon/2.4.\*/daemon

cd 进去删除 registry.bin 和 registry.bin.lock 两个文件即可。

- 2、Windows下面相比较为简单

例如： C:\Users\andy.gradle\daemon\2.0\daemon下面的，同样也删除以上两个文件即可。

附：版本信息：

- (1) 探针类型：Android SDK
- (2) 探针版本：2.0.\*
- (3) 操作系统版本：Mac OS
- (4) 开发工具：Android Studio 1.3.\*
- (5) JRE：jre1.8
- (6) JDK：jdk1.8
- (7) JVM：Java HotSpot(TM) 64-Bit Server VM

# OneAPM兼容Cordova（PhoneGap）

## 兼容Cordova

如果想对Cordova监控就需要在用到Cordova的Activity中获取Cordova的webView对象，然后调用如下方法即可

```
BlueWare.compatibleCordova(webView);
```

Cordova 5.0和以后的版本 使用如下代码获取WebView

```
WebView wV = (WebView)appView.getEngine().getView();
```

小于5.0版本的，在Activity中应该可以直接获取。

## 资料下载

<https://oneapm.kf5.com/posts/view/102001/>

# 升级Android Agent 2.0.4后嵌入Token那行代码报错？

- 1、版本信息：

- (1) 探针类型：Android SDK
- (2) 探针版本：2.0.4
- (3) 操作系统版本：无
- (4) 开发工具：无
- (5) JRE：jre1.8.60
- (6) JDK：jdk1.8.60
- (7) JVM：Java HotSpot(TM) 64-Bit Server VM

- 2、问题回复：

Android Agent 2.0.4 更新后的启动方式与之前版本略有不同，需要修改为：

- (1) 在默认启动的 Activity 中 import OneApmAgent类。

```
import com.oneapm.agent.android.OneApmAgent;
```

- (2) 在 onCreate() 方法中加入如下：

```
OneApmAgent.init(this).setToken("--Your Token--").start();
```

如图：

- (3) Clean Project，并重新启动应用程序。

运行嵌码完成后的APK，连接电脑使用LogCat过滤TAG = oneapm查看log日志。

若出现 "oneapm is running normally version: OneAPM-Version" 的 log 则表示嵌码成功。



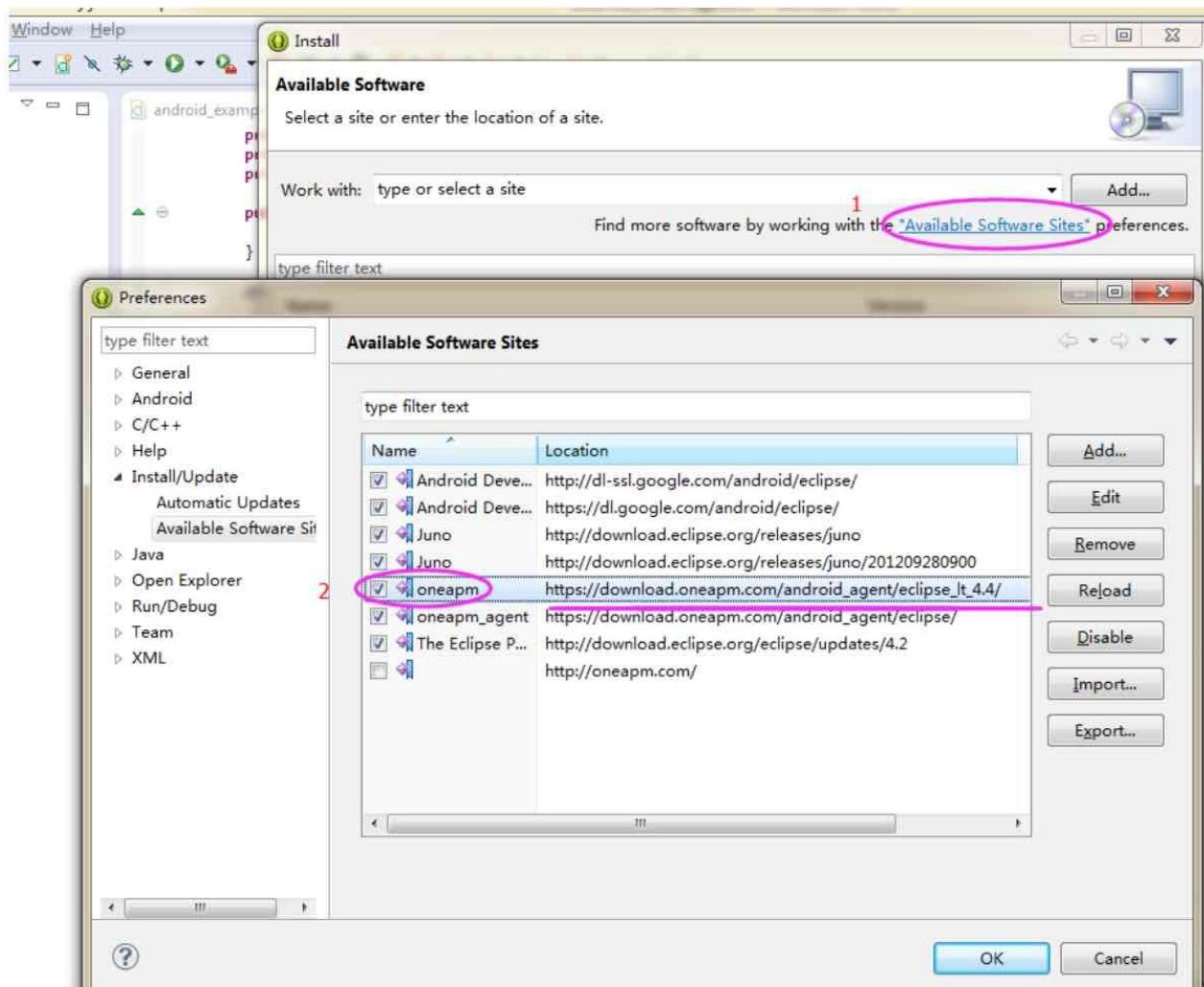
# Eclipse 安装 SDK 提示“The chosen operation is not currently available”

## 故障说明

错误提示：The chosen operation is not currently available

## 解决方案

- 查看开发工具 Eclipse 版本，去 Eclipse 的安根目录找到.eclipseproduct 打开查看 Eclipse 版本
- 菜单 Help->Install New Software
- 点击 Available Software Sites



- 查看安装的 SDK 版本是否和 Eclipse 版本是否匹配，如果不匹配，请卸载掉，重新安装 SDK
- 参考如下

Eclipse 4.4 之前版本请使用以下链接：

[https://download.oneapm.com/android\\_agent/eclipse\\_lt\\_4.4/](https://download.oneapm.com/android_agent/eclipse_lt_4.4/)

Eclipse 4.4 及之后版本请使用以下链接：

注：OneAPM Eclipse 4.4 插件需要 JDK 1.8。

[https://download.oneapm.com/android\\_agent/eclipse\\_gt\\_4.4/](https://download.oneapm.com/android_agent/eclipse_gt_4.4/)

# 集成了 OneAPM 探针后，Crash 功能界面无数据

## 故障说明

集成 OneAPM 探针之后，在 OneAPM Crash 功能界面没有看到抓取到的日志信息。

## 解决方案

如果您的 App 中使用了类似友盟的错误统计功能的 SDK 或者自己实现了类似 Crash 捕获的代码，将会影响 OneAPM 抓取 Crash 信息。

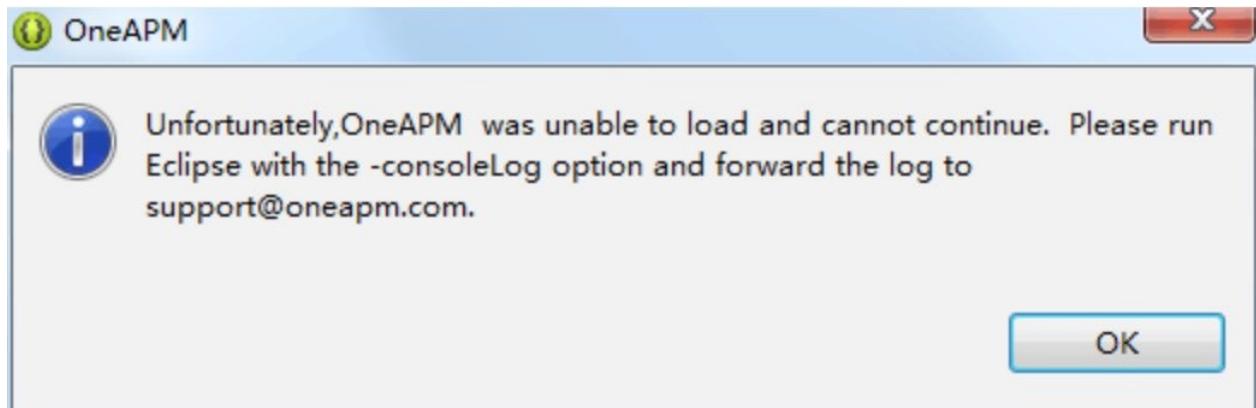
因为，有些第三方 SDK 或者开发者自己实现的 SDK 如果早于 OneAPM 启动，并且在使用了系统默认的异常处理器之后没有把这个处理器还给系统，就会导致 OneAPM 无法得到系统的异常处理器，从而无法抓取 Crash 日志。

请保证 OneAPM 探针启动的时间早于之前所述的代码的启动时间。

# SDK 安装后提示错误信息： No providers installed

## 故障说明

Android SDK 通过 Eclipse 插件方法安装完成，重启后出现 No providers installed 异常



## 解决方案

原因一：没有使用 Oracle JDK，请安装 Oracle 官方 JDK。

原因二：若安装了其他第三方性能监控的 Eclipse 插件，请卸载其他第三方性能监控插件或者重新解压一份新的 Eclipse 重新安装 OneAPM 插件。

原因三：使用了 Oracle JDK，并且 `JAVA_HOME` 指向了这个 JDK，但是 `path` 下的 "java" 命令不是这个 JDK 里面的 Java，而是操作系统给你默认安装的 JRE 下的，如：

```
c:\Program Files\java\....
```

请配置 Java 命令指向 JDK 下面的 Java。

原因四：电脑安装了不同版本的 JDK，但是把这些 JDK 都配置到了环境变量中。请去除多余的 Java 环境变量，在环境变量中配置一个 Java 版本即可。

# SDK 成功部署后 OneAPM 界面无数据显示

问题一：在 App 上没有发生用户操作。

OneAPM 监控的是真实用户体验，只有在用户进行交互操作才会有性能数据产生。

问题二：性能数据正在上传中。

App 每隔一分钟发送一次数据给 OneAPM。首次触发 App，性能数据不会立即上传到 OneAPM。请在交互操作之后，至少等待 1 分钟。

问题三：没有打开网络或网络不通。

在没有网络的问题下，性能数据无法上传到 OneAPM 平台。

问题四：不同的 App，使用同一个 `App Token`。

每个 App 的 `ApplicationId`，对应唯一的 `Token`。首个使用 `Token` 的 App 被认为是合法用户，之后使用同一 `Token` 的 App 被视为非法请求，性能数据将被过滤。



问题五：某些编译器没有对已缓存的代码进行二次编译。

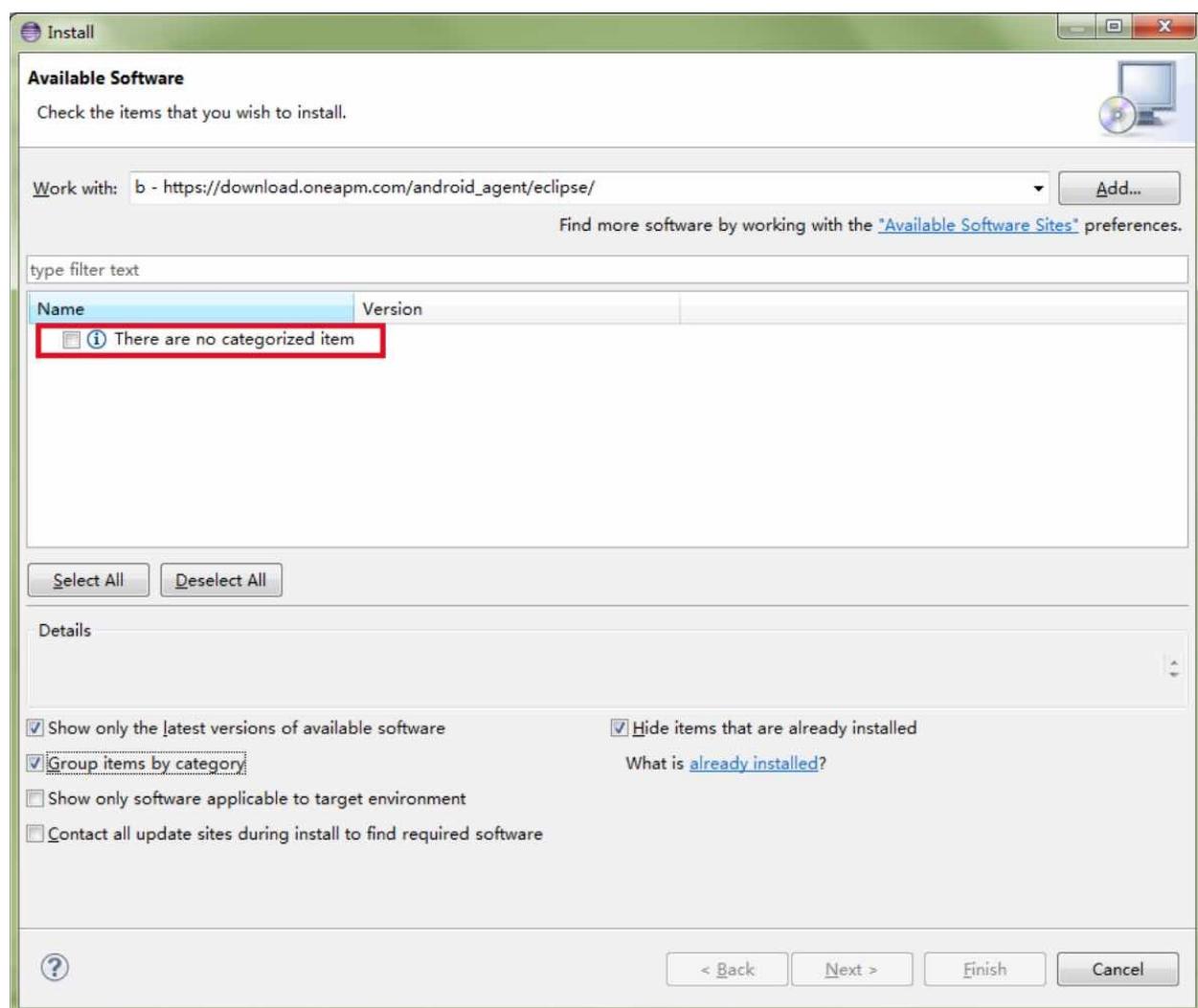
由于某些编译器，例如 Eclipse 在有些情况下，会缓存一些代码，为了提高代码编译的效率，对于一些已经缓存的代码，编译器不会做二次编译。所以为了确保成功集成 OneAPM，需要对于编译器生成的一些缓存文件例如 Eclipse 的 Android 工程下面的 `bin`、`gen` 文件夹等要执行手动删除操作，之后回到工作空间手动 `clean`，再次导出包安装运行，查看后台是否有数据。



# Eclipse 插件安装过程中无法勾选 OneAPM 插件并提示: There are no categorized item

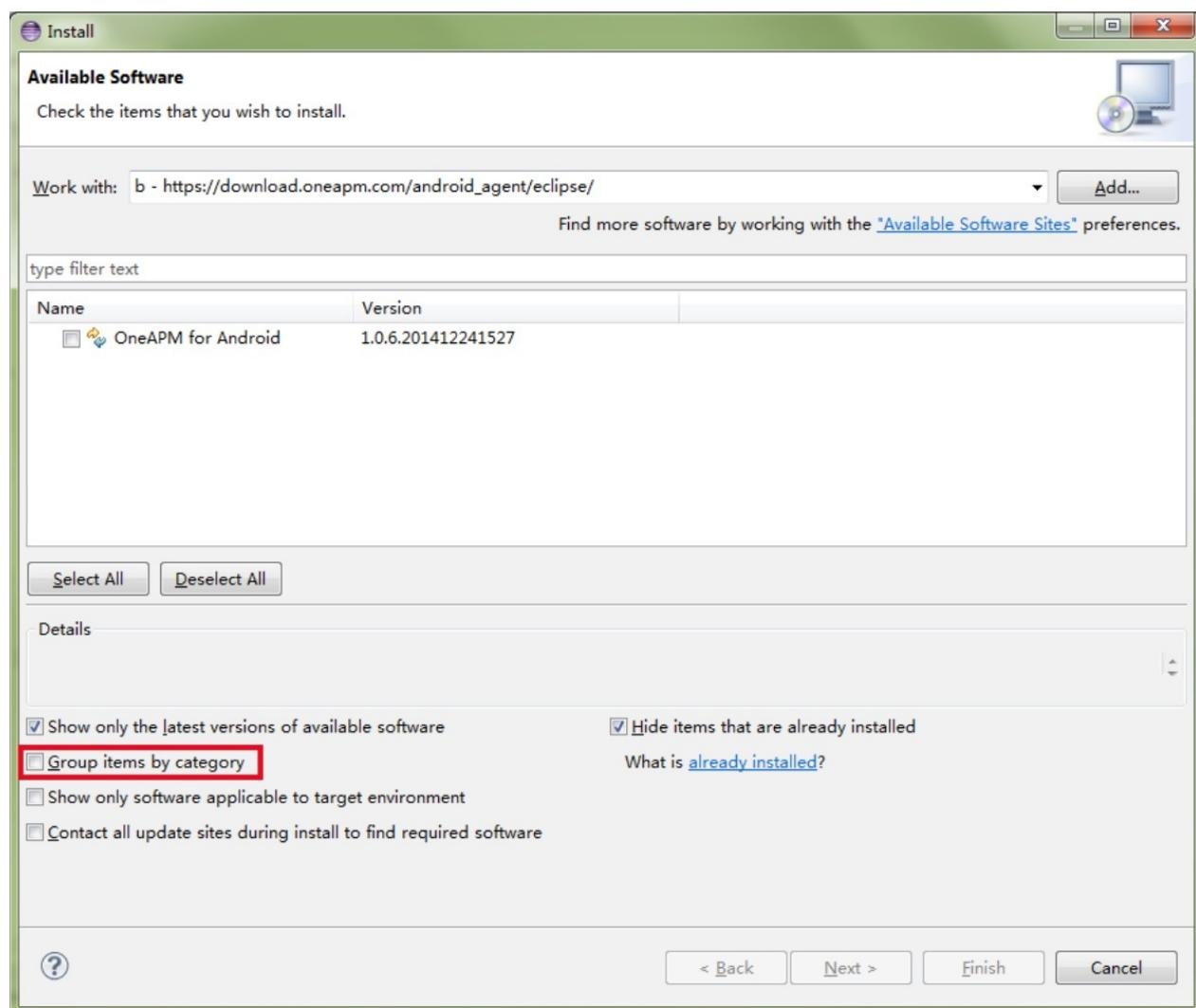
## 故障说明

Eclipse 插件安装过程中无法勾选 OneAPM 插件，提示: There are no categorized item，如下图：



## 解决方案

取消勾选 group items by category



Eclipse 插件安装过程中无法勾选 OneAPM 插件并提示: There are no categorized item

# WebView 性能监控使用说明

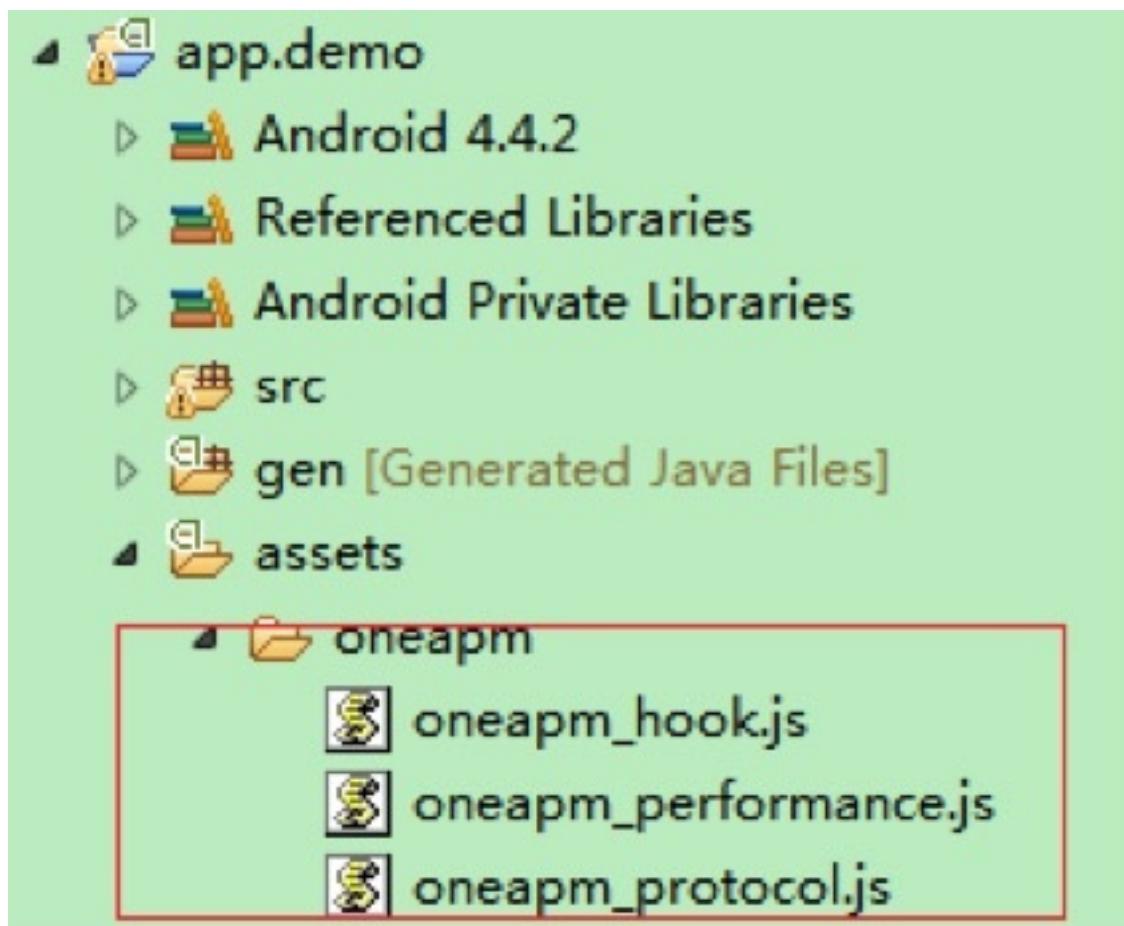
当您按照 OneAPM Android SDK 安装方法完成安装后，只需执行以下两步就可以使用 WebView 监控功能。

WebView 监控 JavaScript 文件下载：

[下载附件](#)

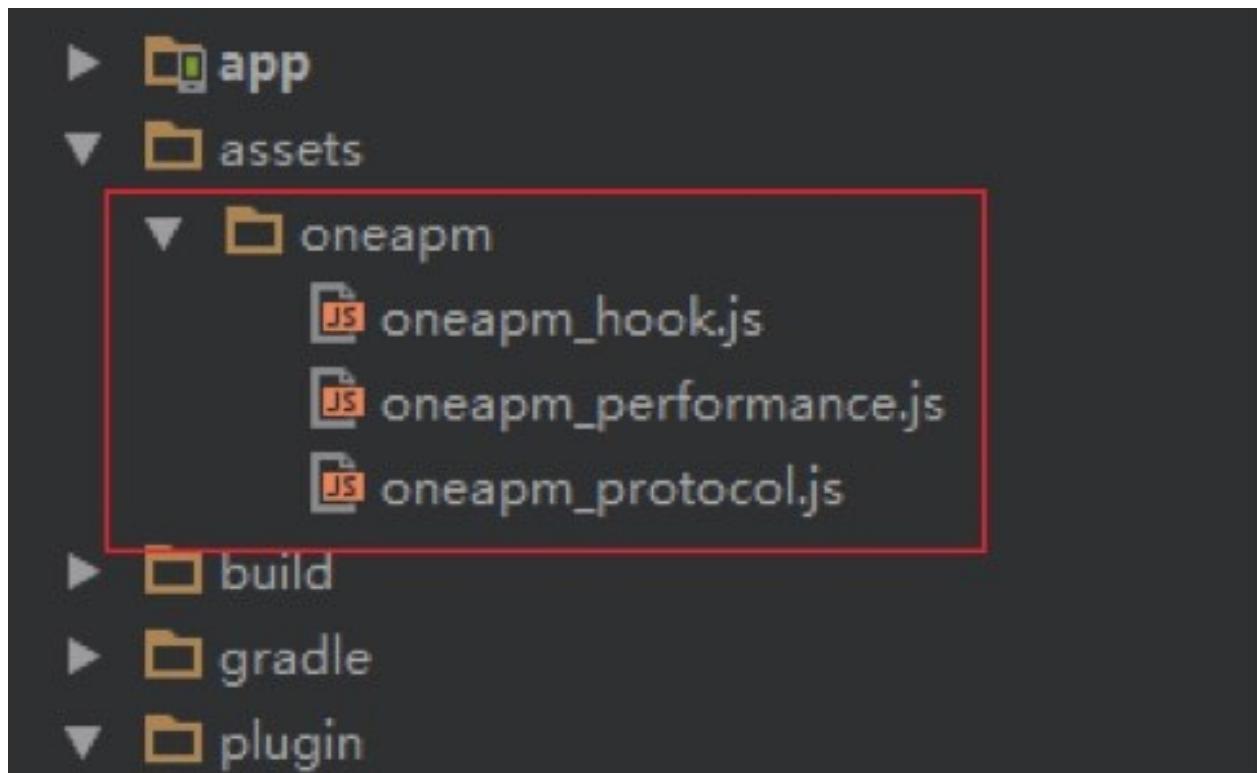
1.eclipse安装方式：

拷贝包含 oneapm\_hook.js、oneapm\_performance.js、oneapm\_protocol.js 文件的 oneapm 文件夹到 Android 项目工程的 assets 文件夹下，如果没有这个文件夹请手动添加。添加完成之后，项目结构如下图所示：



2.Android Studio安装方式：

拷贝包含 oneapm\_hook.js、oneapm\_performance.js、oneapm\_protocol.js 文件的 oneapm 文件夹到 app/src/main 下的 assets 文件夹下，如果没有这个文件夹请手动添加。添加完成之后，项目结构如下图所示：



3. 在自己的代码中找到需要监控的 WebView 对象，新建一个OneapmWebViewClient 对象，代码如下。

其中onPageFinished和shouldOverrideUrlLoading这两个代码必须要调用super方法。

说明：如果您自己有WebViewClient这个类，请保证这个类继承自 OneapmWebViewClient 这个类，并调用相关的父类super方法。

代码示例：

```
OneapmWebViewClient client = new OneapmWebViewClient(webView){

 @Override
 public void onPageFinished(WebView view, String url) {
 super.onPageFinished(view, url);
 }

 @Override
 public boolean shouldOverrideUrlLoading(WebView view, String url) {
 return super.shouldOverrideUrlLoading(view, url);
 }
};
```

自定义WebViewClient示例代码：

```
public class MyWebViewClient extends OneapmWebViewClient{

 public MyWebViewClient(WebView wView) {
 // TODO Auto-generated constructor stub
 super(wView);
 }

 @Override
 public void onPageFinished(WebView view, String url) {

 super.onPageFinished(view, url);

 }

 @Override
 public boolean shouldOverrideUrlLoading(WebView view, String url) {

 return super.shouldOverrideUrlLoading(view, url);

 }
}
```

注意： **super**方法中需要传入 **WebView**参数。

调用 **webView**的 **setWebViewClient**把步骤1的对象设置进去。

```
webView.setWebViewClient(client);
```

4. 测试，集成结束后，正常运行您的应用程序，打开包含 **WebView** 的页面，一段时间后就可登录 **OneAPM** 点击 **WebView** 查看是否出现 **WebView** 的性能数据。

注意： **WebView**监控功能 **Android 4.3** 及以上版本可用。

# 安装 Android 探针后启动发生异常： **finished with non-zero exit value 1**解决办法

## 错误描述

安装 Android 探针后启动发生异常，错误如下：

```
Error:Execution failed for task ':app:dexDevDebug'.
com.android.ide.common.process.ProcessException:org.gradle.process.internal.ExecException: Process 'command 'D:/Java/jdk1.7.0_79/bin/java.exe'' finished with
non-zero exit value 1
```

Execution failed for task ':app:dexDebug'.
> com.android.ide.common.process.ProcessException: org.gradle.process.internal.ExecException: Process 'command 'C:\Program Files\Java\jdk1.7.0\_79\bin\java.exe'' finished with non-zero exit
value 1

## 版本信息

- (1) 探针类型： android sdk
- (2) 探针版本： 2.0.3
- (3) 操作系统版本： Windows 7
- (4) 开发工具： Android Studio 1.3.2
- (5) JRE： jre1.7.0\_79
- (6) JDK： jdk1.7.0\_79/jdk1.8.0\_05/jdk1.8.0\_45
- (7) JVM： Java HotSpot(TM) 64-Bit Server VM

## 解决办法

这个问题是由于 Android Studio 编译时造成的，在启动之后可以在任务管理器中观察到 studio.exe 和 java.exe 运行时占用内存量都很高，导致内存占用很大才导致的这个问题，因此修改配置把内存调高即可。

解决办法：

在 app 下的 build.gradle 配置文件中加入如下配置：

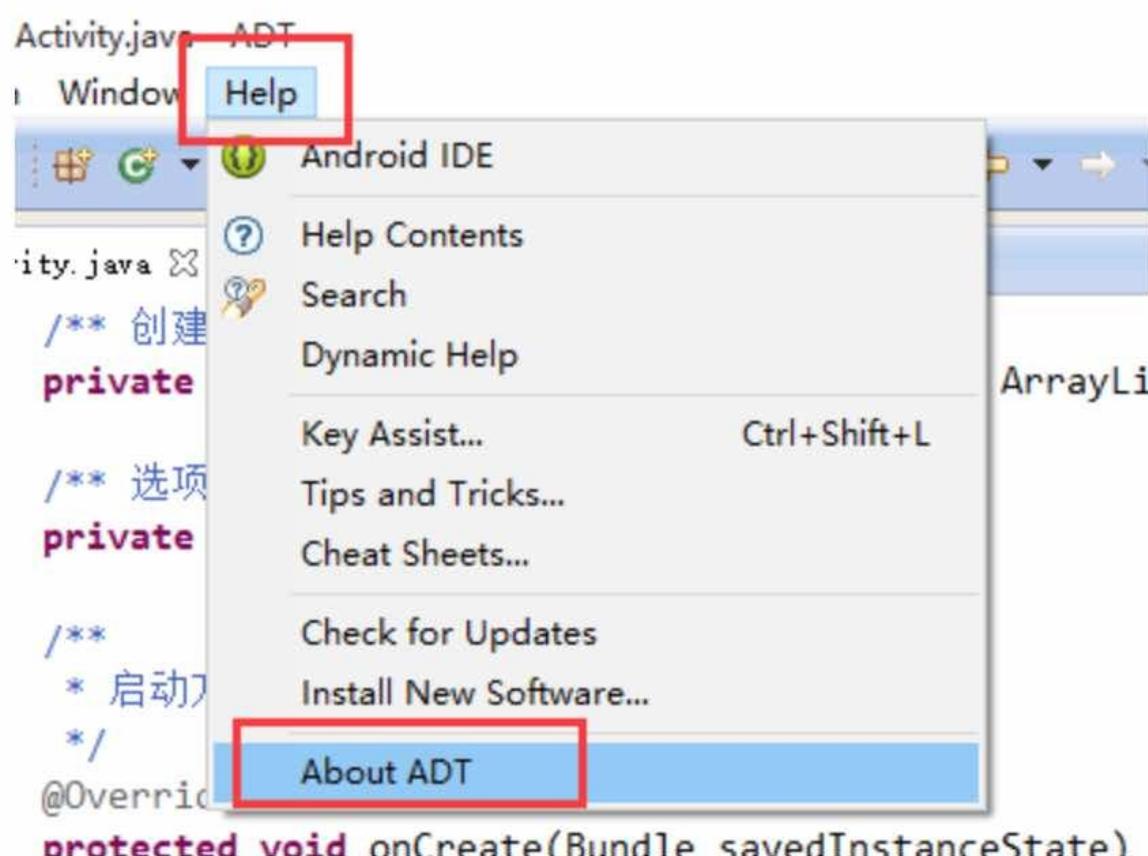
```
dexOptions {
 javaMaxHeapSize "xxg"
}
```

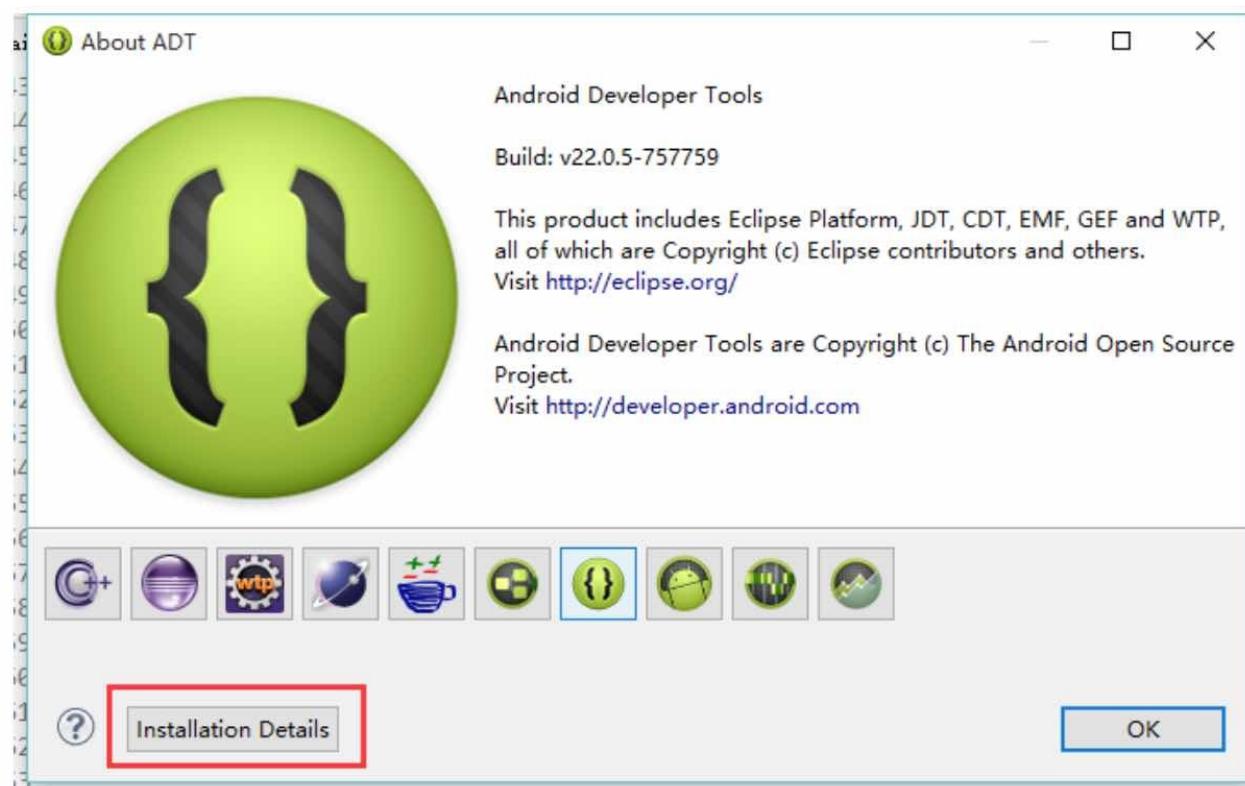
"xxg"可根据自己电脑配置修改，如："4g"，代表 JVM最大 Heap 内存为4G。

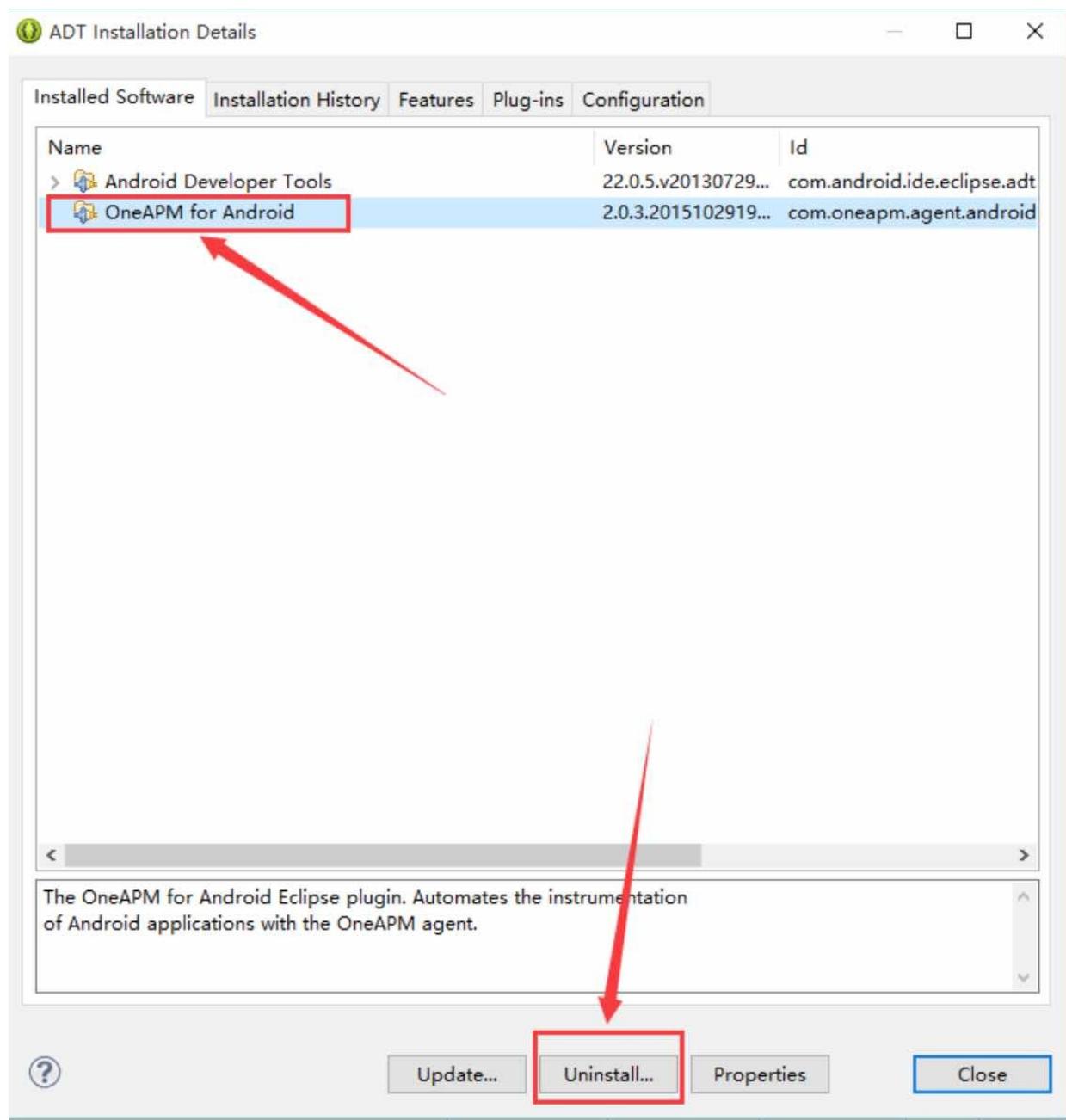
# Eclipse 完全卸载说明

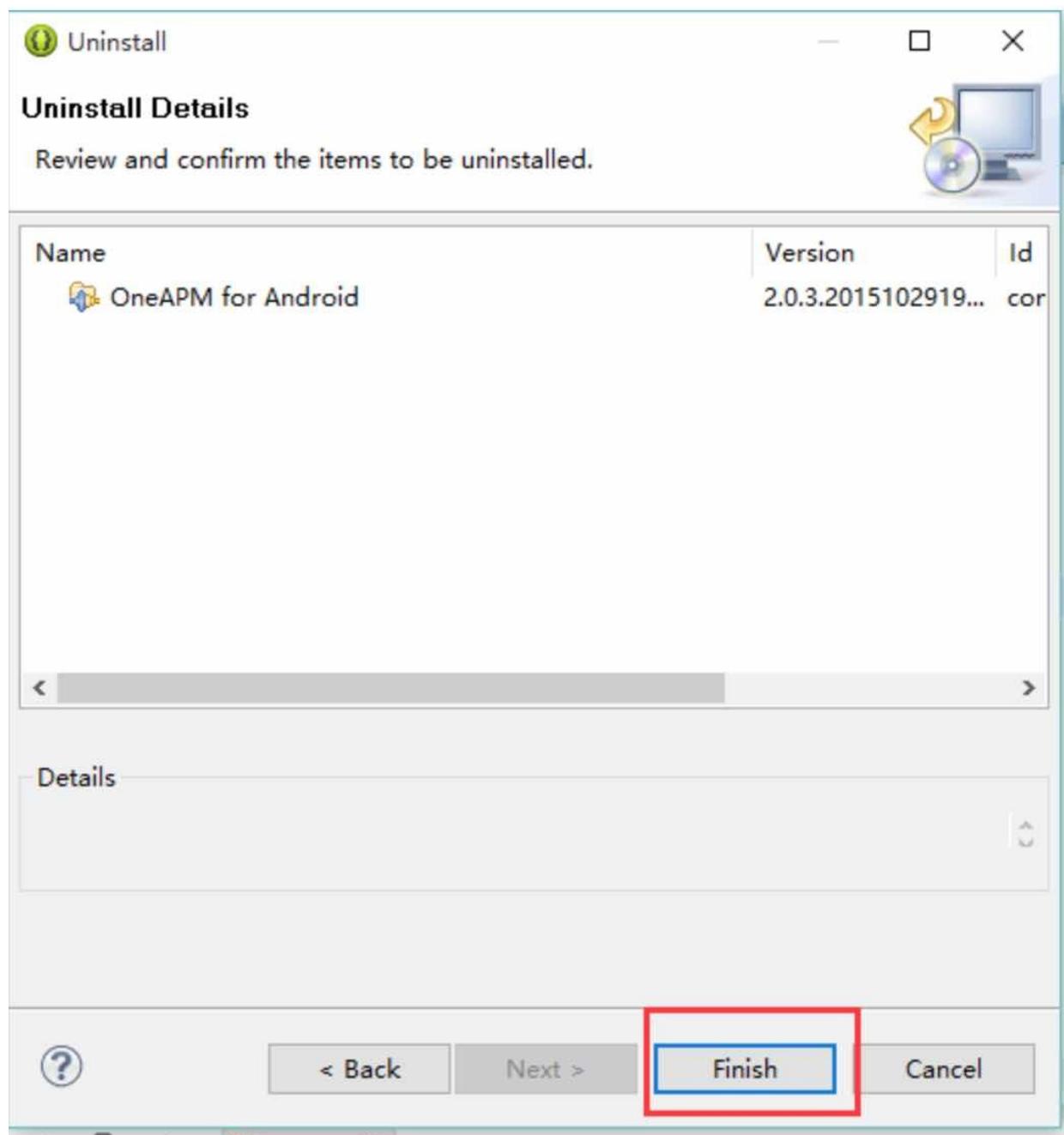
注意：Eclipse 安装 Android Agent 2.0.3.\* 版本升级到2.0.4.\* 需要先卸载旧探针再进行安装。

首先在 Eclipse 中卸载 OneAPM 的 Agent 插件： Help -> About ADT -> Installation Details。









打开Eclipse的安装目录: \eclipse\plugins

删除下面的目录:

名称	修改日期	类型	大小
com.android.ide.eclipse.adt.package_22.0.5.v201307292155--75...	2014/11/11 14:46	文件夹	
com.oneapm.agent.android.osgi.framework_2.0.3	2016/1/19 11:51	文件夹	
com.oneapm.android.eclipse.plugin_2.0.3	2016/1/19 11:51	文件夹	
org.apache.ant_1.8.3.v201301120609	2014/11/11 14:46	文件夹	
org.eclipse.cdt.core.win32.x86_64_5.2.0.201202111925	2014/11/11 14:46	文件夹	
org.eclipse.core.runtime.compatibility.registry_3.5.101.v2013010...	2014/11/11 14:46	文件夹	
org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.200.v20120...	2014/11/11 14:46	文件夹	

## \eclipse\features

删除下面的目录：

文件夹	最后更新	大小	更多
com.android.ide.eclipse.adt.package...	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.adt_22.0.5.v2...	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.ddms_22.0.5....	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.gddebugger_...	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.hierarchyvie...	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.ndk_22.0.5.v...	2014/11/11 14:46	文件夹	
com.android.ide.eclipse.traceview_22....	2014/11/11 14:46	文件夹	
com.oneapm.agent.android.eclipse.f...	2016/1/19 11:51	文件夹	
org.eclipse.cdt.gdb_7.0.0.2012021119...	2014/11/11 14:46	文件夹	
org.eclipse.cdt.gnu.build_8.0.2.20120...	2014/11/11 14:46	文件夹	
org.eclipse.cdt.gnu.debug_7.1.1.2012...	2014/11/11 14:46	文件夹	
org.eclipse.cdt.gnu.dsf_4.0.1.2012021...	2014/11/11 14:46	文件夹	
org.eclipse.cdt.platform_8.0.2.201202...	2014/11/11 14:46	文件夹	
org.eclipse.cdt_8.0.2.201202111925	2014/11/11 14:46	文件夹	
org.eclipse.e4.rcp_1.1.2.v20130130-1...	2014/11/11 14:46	文件夹	
"	2014/11/11 14:46	文件夹	

编辑 artifacts.xml 文件：

文件夹	最后更新	大小	更多
configuration	2016/1/19 11:51	文件夹	
dropins	2013/7/29 15:06	文件夹	
features	2016/1/19 11:51	文件夹	
p2	2014/12/23 3:45	文件夹	
plugins	2016/1/19 11:51	文件夹	
readme	2014/11/11 14:46	文件夹	
.eclipseproduct	2013/2/4 4:25	ECLIPSEPRODUC...	1 KB
artifacts.xml	2016/1/19 11:51	XML 文件	83 KB
eclipse.exe	2013/2/4 5:05	应用程序	305 KB
eclipse.ini	2016/1/19 11:51	配置设置	1 KB
eclipsec.exe	2013/2/4 5:05	应用程序	18 KB
epl-v10.html	2013/2/4 4:28	Firefox HTML D...	17 KB
notice.html	2013/2/4 4:28	Firefox HTML D...	9 KB

搜索“oneapm”关键字，删除所有包含“oneapm”的artifact节点元素。

```
<property name='download.size' value='458248' />
</properties>
</artifact>
<artifact classifier='osgi.bundle' id='com.oneapm.agent.android.osgi.framework' version='2.0.3'>
<properties size='1'>
<property name='download.size' value='4783' />
</properties>
<repositoryProperties size='1'>
<property name='artifact.folder' value='true' />
</repositoryProperties>
</artifact>
<artifact classifier='osgi.bundle' id='org.apache.commons.codec' version='1.3.0.v201101211617'>
<properties size='1'>
<property name='download.size' value='55011' />
```

重启Eclipse即可。

# iOS

# SDWebImage 造成的 crash 问题

## 故障说明

SDWebImage 是一个项目开发中必不可少的三方库，但是当与 OneAPM 的 SDK 结合开发项目时，若 SDWebImage 的版本不是最新的，会出现 crash 情况，在 iOS7 系统 iPhone4 设备下的频率尤其高，crash log 打印如下图：

```
3 libsystem_c.dylib 0x3ad10fe0 abort + 80
4 libc++abi.dylib 0x3a03fcd2 abort_message + 70
5 libc++abi.dylib 0x3a0586e0 default_terminate_handler() + 248
6 libobjc.A.dylib 0x3a79df62 _objc_terminate() + 190
7 libc++abi.dylib 0x3a0561c4 std::__terminate(void (*)()) + 76
8 libc++abi.dylib 0x3a055d28 __cxa_rethrow + 96
9 libobjc.A.dylib 0x3a79de12 objc_exception_rethrow + 38
10 CoreFoundation 0x2ff20f30 CFRunLoopRunSpecific + 636
11 CoreFoundation 0x2ff649ae CFRunLoopRun + 94
12 DouPaiTest 0x002a1e50 -[SDWebImageDownloaderOperation
start] (SDWebImageDownloaderOperation.m:110)
```



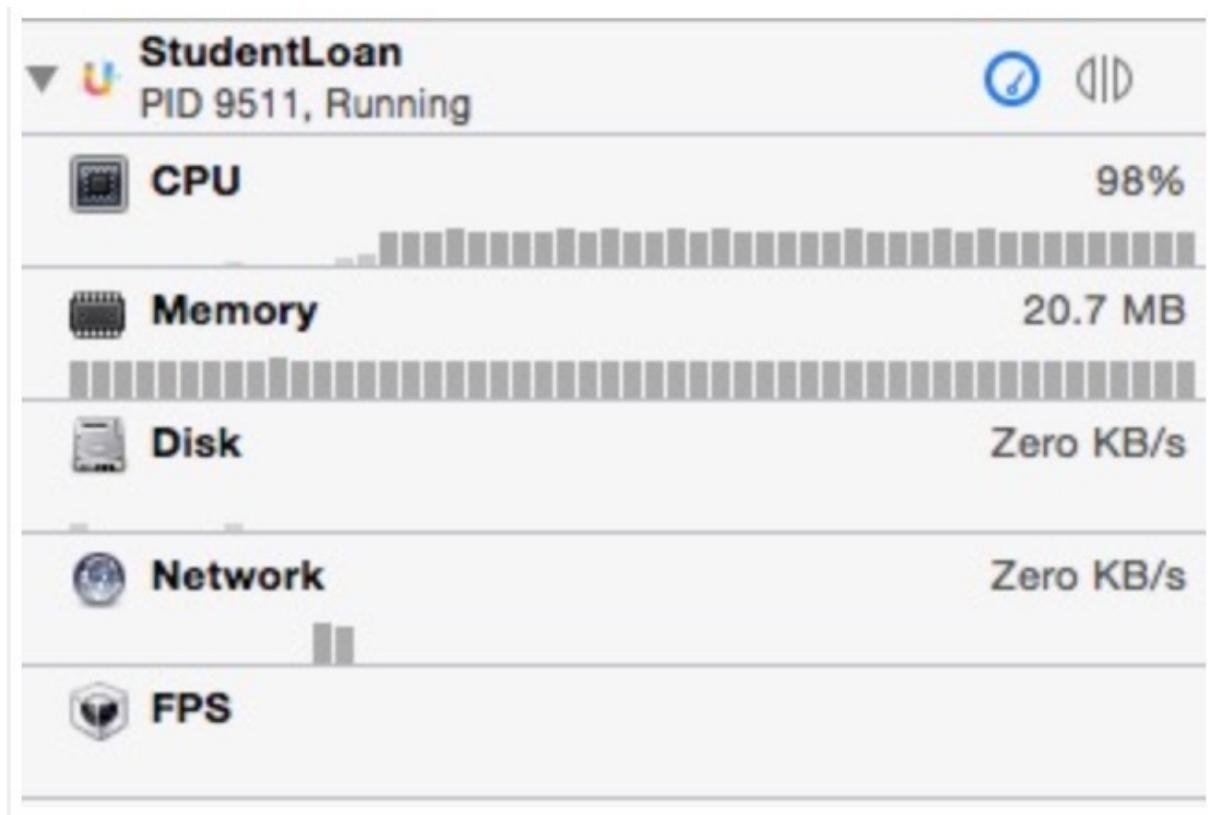
## 解决办法

此时建议用户去 GitHub 下载最新的 SDWebImage 库，  
<https://github.com/rs/SDWebImage>。

# Xcode 版本低于6.3，CPU 占用达到100%

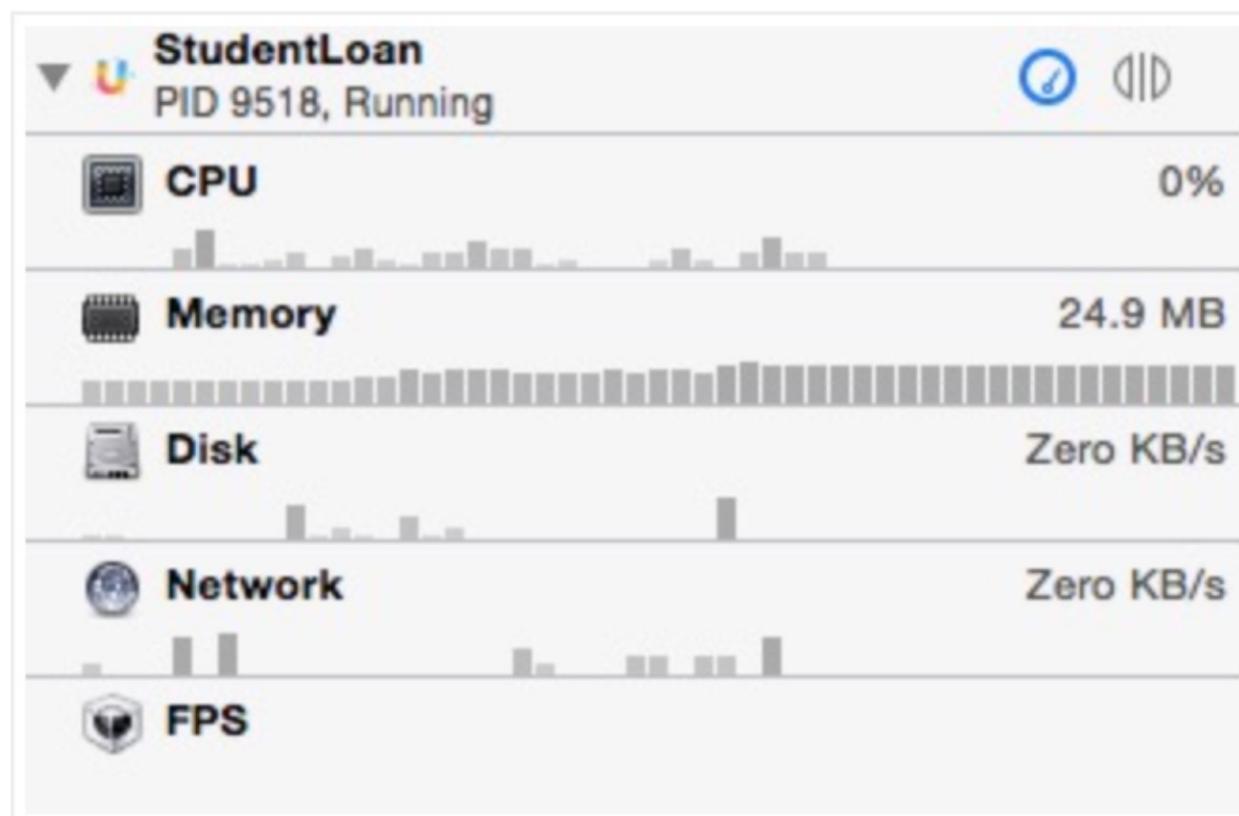
## 故障描述

当前最新的 Xcode 版本为6.3，我们最新的 SDK 是基于此版本进行开发的。若开发者的 Xcode 版本低于6.3，可能会出现 CPU 占用接近100%的情况，如下图。



## 解决办法

此时，建议您将 Xcode 升级为最新版本，这样有助于您后期的项目开发。升级后再次查看，CPU 显示若如下图，则为正常。



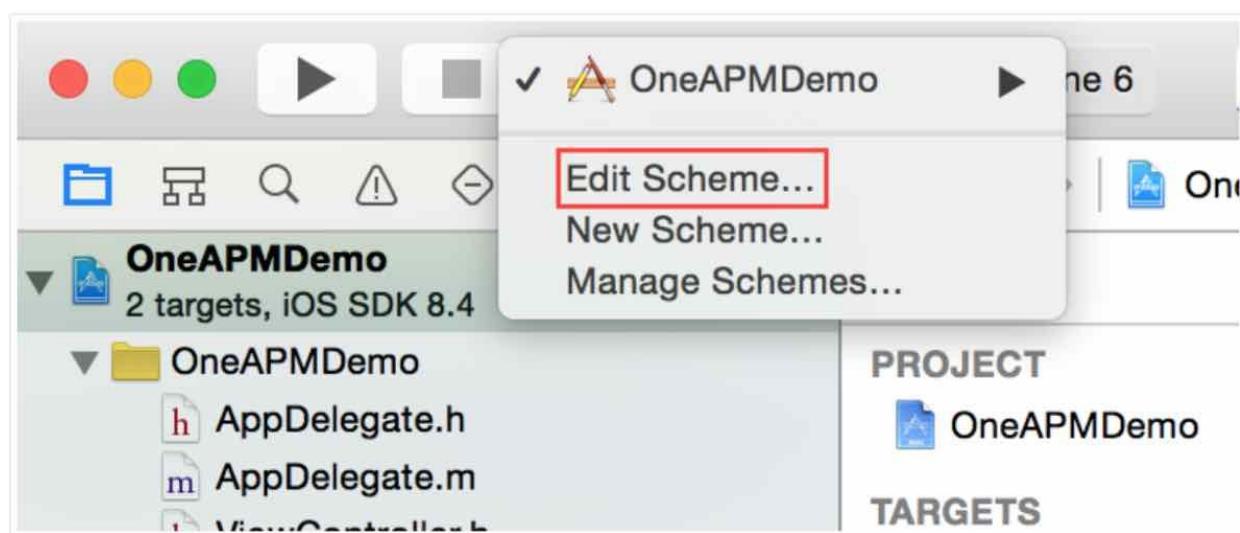
# 上传dSYM文件步骤

上传dSYM文件，分为 准备 和 上传 两大步骤：

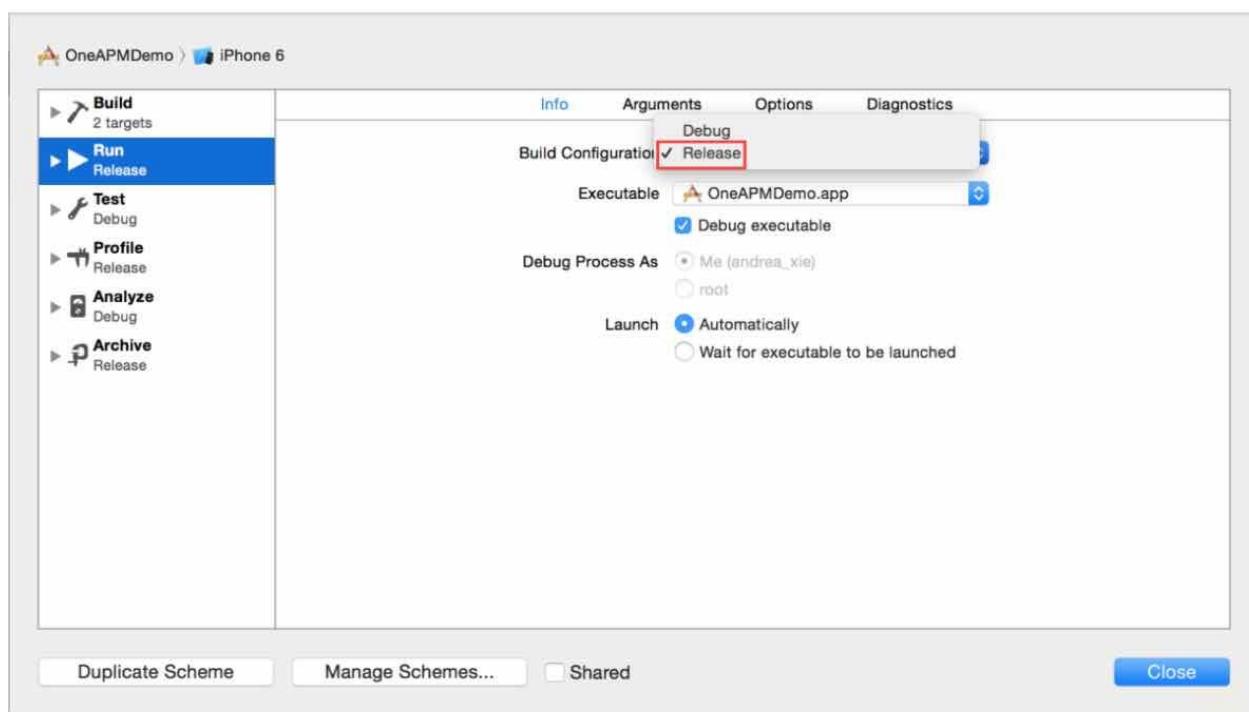
步骤一：准备dSYM文件

(一)

打开工程，选择Edit Scheme

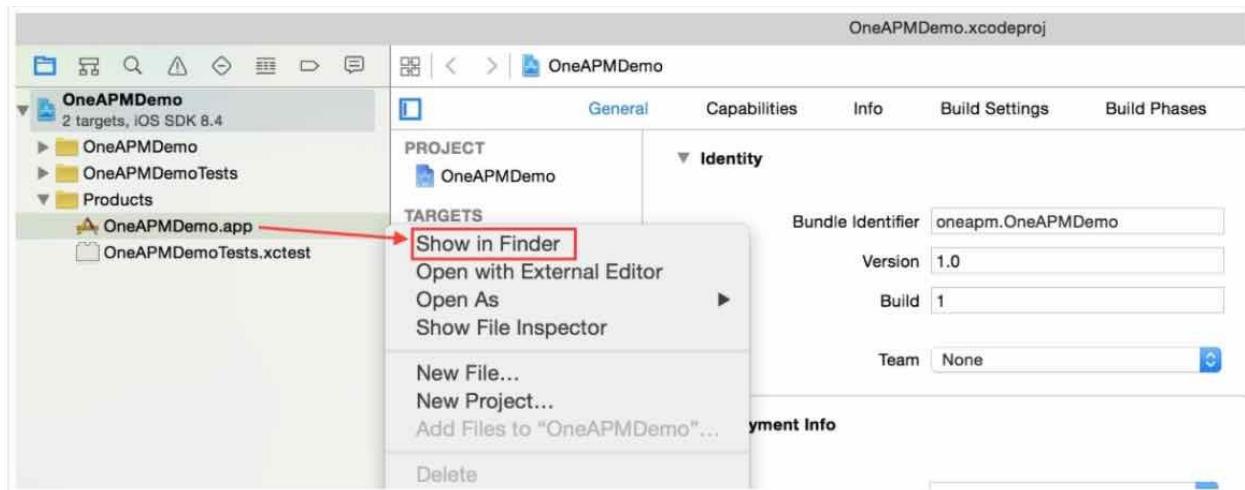


将工程由Debug模式转为Release模式,然后点击Close关闭弹框。

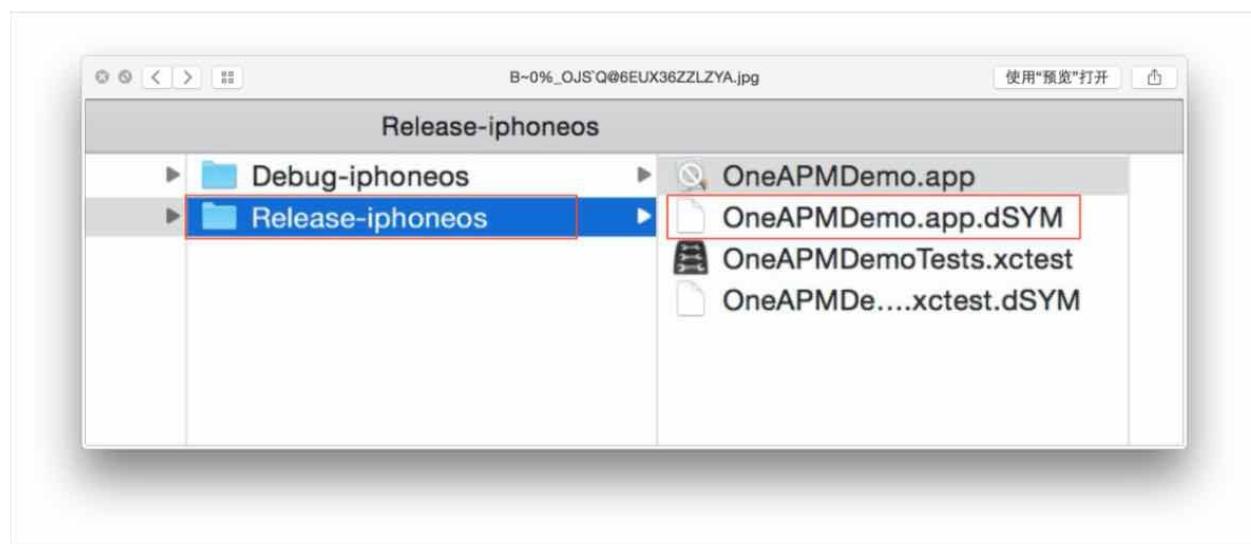


按住Command + B ,进行编译。

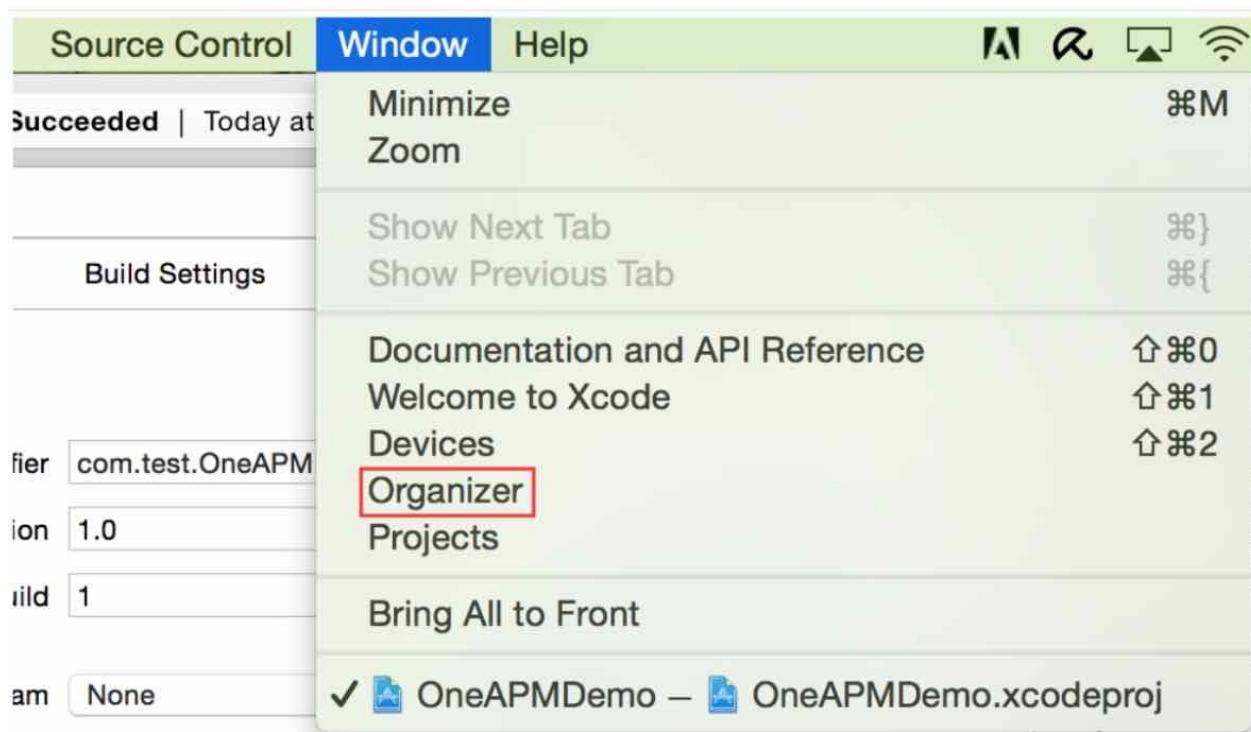
Xcode – “Products”下，右击XXX.app文件，选择“Show in Finder”，如下图所示：



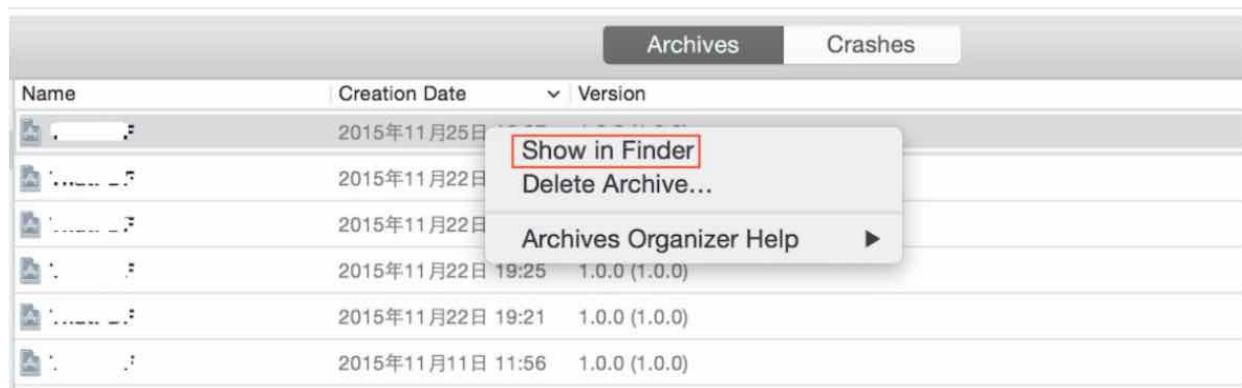
与XXX.app同目录下，即可找到XXX.app.dSYM文件，如下图所示：



(二) Archive模式Build时：选择Xcode – >Window – >Organizer,如下图所示：



在弹框中选择“Archives”选择对应APP右击，点击“Show in Finder”，如下图所示：



右击XXX.xcarchive文件，选择“显示包内容”，如下图所示：



进入dSYM目录下可以找到XXX.app.dSYM文件，接下来准备上传它。

步骤二：上传dSYM文件

浏览器登录OneAPM – Mobile 页面，<https://mi.oneapm.com/mobile/app#/>

按照如下操作：

- a.点击“崩溃”进入崩溃信息展示界面；
- b.点击崩溃信息界面右上角“上传dSYM文件”按钮，选择APP版本号，选择要上传的dSYM文件“选取”
- c.“保存”。



至此，dSYM文件已经上传成功，等待收集Crash数据吧！

# 为什么要上传dSYM文件？

首先向您解释一下dSYM文件（符号表）到底是什么？

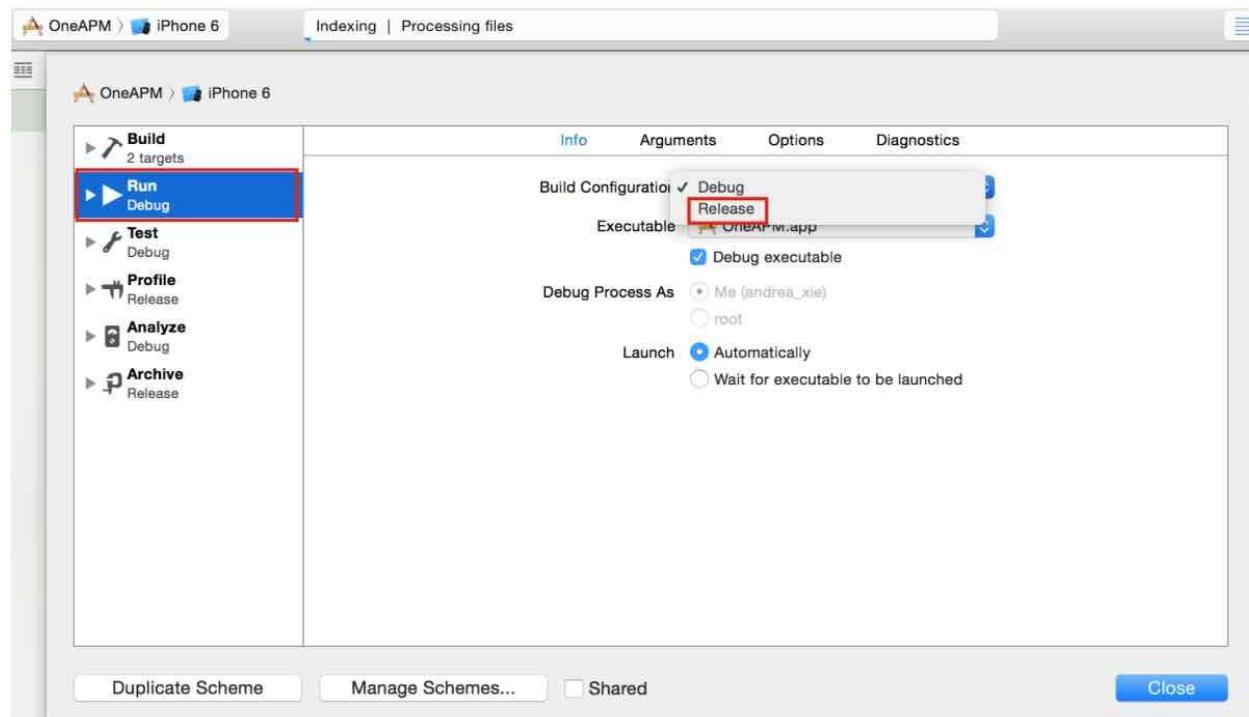
Xcode编译项目后，我们会看到一个同名的 dSYM 文件，dSYM 文件是保存 16 进制函数地址映射信息的中转文件。如果Debug调试模式的话，我们可以根据log的输出轻松定位到导致crash的原因，但当应用 release 模式打包或上线后出现Crash，这个时候就需要通过出错的函数地址去查询 dSYM 文件中程序对应的函数名和文件名。

为了帮助用户快速准确地定位App发生Crash的详细位置，OneAPM使用dSYM文件将 crash文件中的16进制地址转换成可读信息（内存地址、函数名、文件名、行号信息）。所以上传对应的dSYM文件是很有必要的。

# 崩溃数据统计不到

崩溃数据统计不到，建议您确认：

1. 需要在Release模式下真机测试，并断开真机与Xcode的连接。选择工程名 – – Edit Scheme... – – Run – – Build Configuration – – 将设置改为Release。



## 集成iOS sdk过程中搜索不到**libz.dylib**和**libstdc++.dylib**库

iOS9后 原来的dylib后缀名的库全部修改成了tbd,因此 libz.dylib相当于libz.tdb,  
libstdc++.dylib 相当于 libstdc++.tdb 如果Link Binary With Libraries中添加的时候 找不到  
libz.dylib和libstdc++.dylib库，直接搜索对应的libz.tdb和libstdc++.tdb即可。

# SDK成功部署后OneAPM界面无数据显示

SDK成功部署后OneAPM界面无数据显示，可能有以下原因，请依次排查：

1. 在 App 上没有发生用户操作。

OneAPM 监控的是真实用户体验，只有在用户进行交互操作才会有性能数据产生。

2. 性能数据正在上传中。

App 每隔一分钟发送一次数据给 OneAPM。首次触发 App，性能数据不会立即上传到 OneAPM。请在交互操作之后，至少等待 1 分钟。

3. 没有打开网络或网络不通。在没有网络的问题下，性能数据无法上传到 OneAPM 平台。

4. 不同的 App，使用同一个 App Token。每个 App 的 ApplicationId，对应唯一的 Token。首个使用 Token 的 App 被认为是合法用户，之后使用同一 Token 的 App 被视为非法请求，性能数据将被过滤。

5. 崩溃收集需要在真机 release，非xcode介入模式下运行才有效，不然出现崩溃就会被 Xcode 截断无法正常上传数据。

## 若由于网络情况，对数据收集失败，等网络情况恢复后，还能收集到吗？

基本可以。

网络较差，导致的数据无法上传，当网络恢复后，会将发送失败的数据重新发送。但如果用户在此期间卸载APP或者清空数据的话，有可能会导致数据的丢失。

# 不同应用可以使用同一Token值吗？

不可以。

事实上，只要包名不同，就要使用新的Token值。

在添加应用过程中，为应用程序取名之后，系统会根据时间戳自动生成唯一的Token值，嵌入Token之后，就会与对应包名的应用产生绑定，以识别并上传对应的应用性能数据。因此一旦包名发生改变，为避免数据发生混乱或遗漏，建议您重新申请Token。