ALIBABA CLOUD

阿里云

应用实时监控服务 ARMS

App监控

文档版本：20220713

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

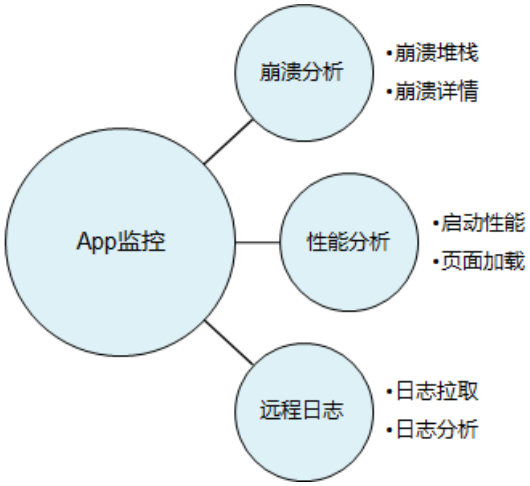
1.App监控概述 07
2.快速入门：创建监控任务 10
3.接入指南 11
3.1. App监控接入概述 11
3.2. 崩溃分析 11
3.2.1. Android SDK接入（Maven集成） 11
3.2.2. Android SDK接入（本地集成） 17
3.2.3. Android SDK接口说明 22
3.2.4. iOS SDK接入（Pod集成） 23
3.2.5. iOS SDK接入（手动集成） 27
3.2.6. iOS SDK接口说明 30
3.3. 性能分析 31
3.3.1. Android SDK接入（Maven集成） 31
3.3.2. Android SDK接入（本地集成） 37
3.3.3. iOS SDK接入（Pod集成） 45
3.3.4. iOS SDK接入（手动集成） 48
3.4. 远程日志 53
3.4.1. Android SDK接入（Maven集成） 53
3.4.2. Android SDK接入（本地集成） 60
3.4.3. iOS SDK接入（Pod集成） 68
3.4.4. iOS SDK接入（手动集成） 72
4.崩溃分析 79
4.1. 实时数据 79
4.2. 崩溃/卡顿/异常数据 86
4.3. 高级搜索 92
4.4. 多维分析 99

4.5. 告警管理 101
4.5.1. 管理联系人 101
4.5.2. 新建告警规则 102
4.5.3. 查看告警历史 103
4.5.4. 获取钉钉机器人Webhook地址 104
5.性能分析 106
5.1. 概览 106
5.1.1. 启动速度 106
5.1.2. 页面性能 108
5.1.3. 网络请求 110
5.2. 启动 112
5.3. 页面 113
5.4. 网络 114
5.4.1. 响应时间 114
5.4.2. 网络错误 117
5.4.3. HTTP状态码 120
5.4.4. 网络劫持 123
5.4.5. 地域 126
5.4.6. 运营商 127
5.4.7. 版本 129
5.4.8. 设备 131
5.5. 地域 134
5.6. 设置URL过滤 135
5.7. 设置域名及分组 135
5.8. 术语解释 135
6.参考信息 138
6.1. 崩溃指标说明 138
7.故障排除 140

7.1. 为什么RAM用户查看应用详情时出现错误？ 140
8.App监控常见问题 142

1.App监控概述

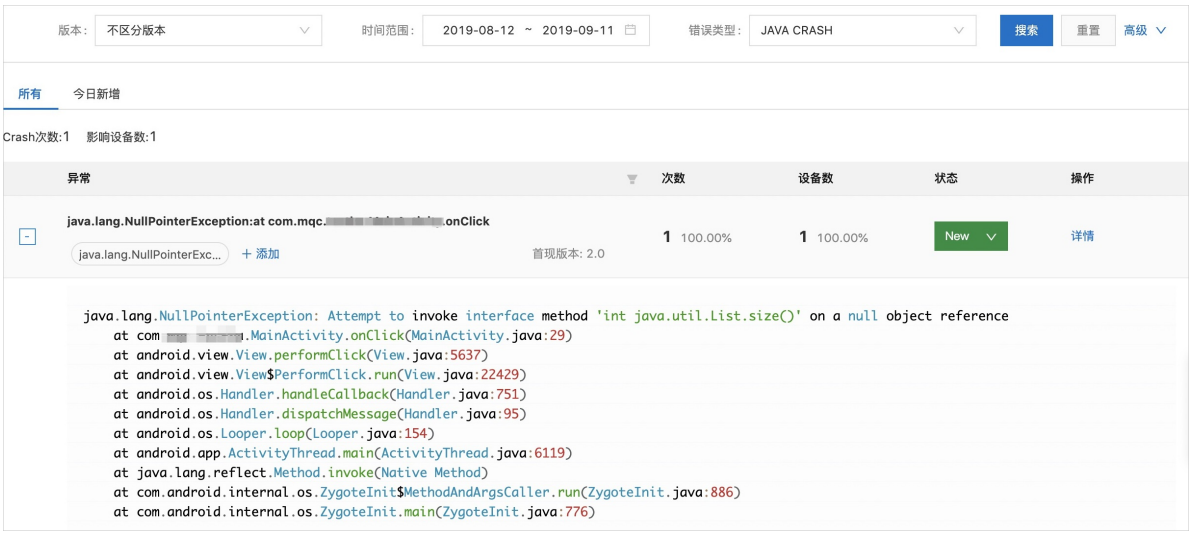
ARMS App监控是专注于监控移动设备上的应用性能和用户体验的工具，分别从崩溃分析、性能分析和远程日志这三个方面来帮助您精确衡量App应用的性能，并且能够实时监控、快速定位性能和可用性问题，帮助您以低成本、高效率发现App应用中的各类隐患。



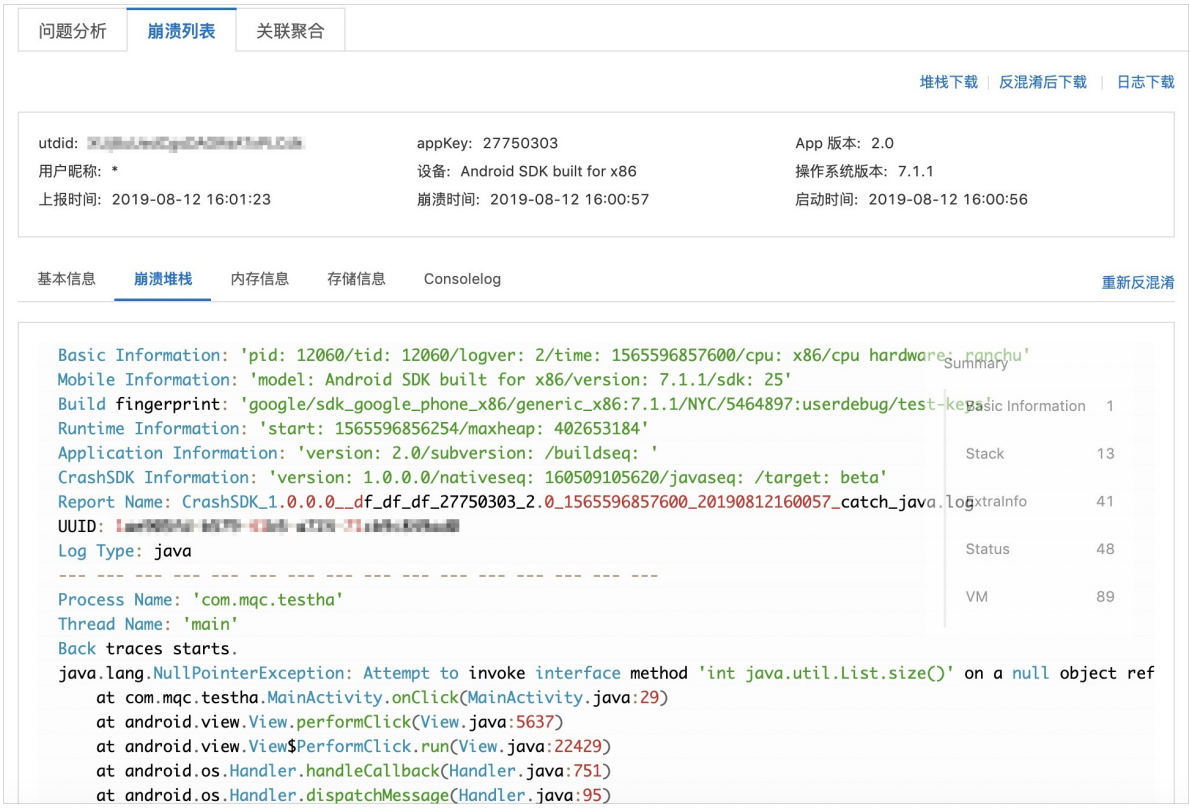
崩溃分析

从用户体验的角度来看，移动端App应用的崩溃是影响App应用可用性的最大问题。由于崩溃采集困难，崩溃分析缺乏上下文对照，数据分析缺乏深入挖掘，所以对于崩溃的分析一直是App应用性能管理的难点所在。而ARMS App监控基于这些问题，将Android和iOS平台常见用户的崩溃问题进行归类分析，帮助您快速发现定位问题：

- 崩溃统计



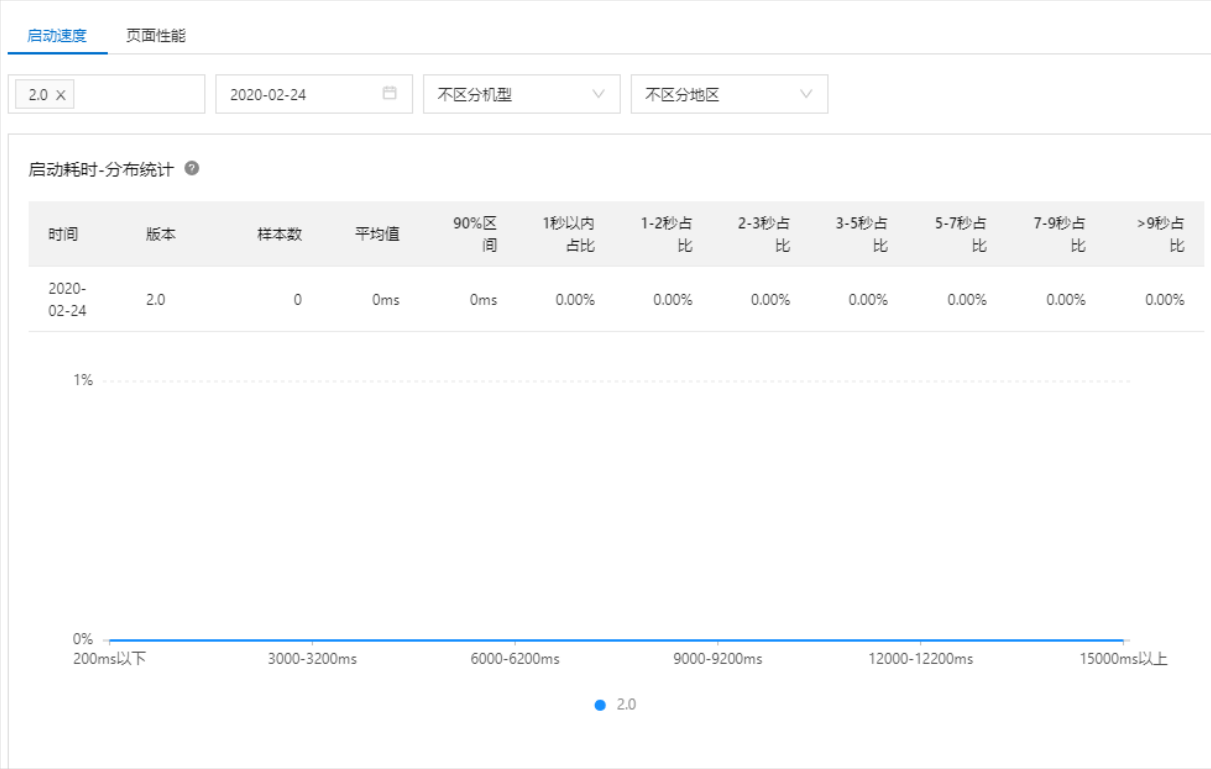
- 崩溃详情



性能分析

性能分析是针对用户上线后的性能问题所提供的服务，可以提供以下功能：

- 线上性能度量：包括启动速度、页面加载耗时、页面流畅度、机型、地域等。
- 影响面聚合分析：包括性能影响用户数和用户百分比。
- 版本对比：不同版本性能情况分析对比。



远程日志

远程日志是针对用户上线后的复杂问题所提供的服务，可以提供以下功能：

- 拉取全量移动端上的崩溃异常日志，还原出错现场，快速定位复杂问题。
- 单独客户反馈异常问题，可快速拉取日志进行排查。

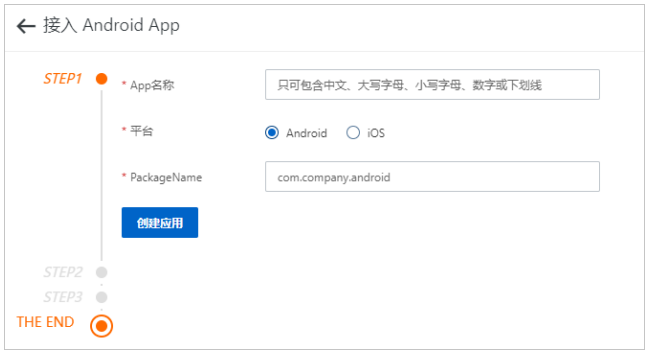
2.快速入门：创建监控任务

在您使用ARMS监控App应用之前，需要先创建监控任务。本文以Android平台应用为例，介绍如何将应用接入到App监控。

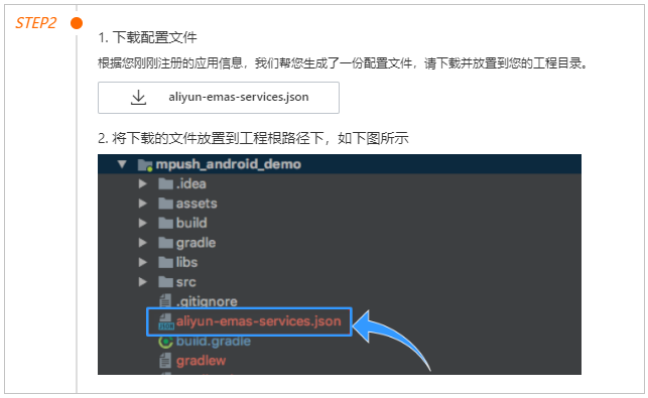
操作步骤

1. 登录ARMS控制台，在左侧导航栏中单击**App监控**。
2. 在**App监控**页面，单击右上角的**创建App监控任务**。
3. 在**接入中心**面板，单击**Android App**。

4. 在接入面板的**Step 1**区域输入App名称和PackageName，选择App的平台，然后单击**创建应用**。



5. 在**Step 2**区域，根据页面提示下载*aliyun-emas-services.json*文件。



6. 根据需求选择将应用接入App监控的崩溃分析、性能分析或远程日志功能。

   各功能接入操作如下：

   - 接入崩溃分析。
   - 接入性能分析。
   - 接入远程日志。

执行结果

在完成上述步骤之后，您可以测试App的监控任务，并登录ARMS控制台查看数据报表。

3.接入指南

3.1. App监控接入概述

您可以根据应用的类型、使用场景选择合适的方式接入App监控的崩溃分析、性能分析和远程日志。

崩溃分析

iOS App
- iOS App（Pod集成）崩溃分析
- iOS App（手动集成）崩溃分析

Android App
- Android App（Maven集成）崩溃分析
- Android App（本地集成）崩溃分析

性能分析

iOS App
- iOS App（Pod集成）性能分析
- iOS App（手动集成）性能分析

Android App
- Android App（Maven集成）性能分析
- Android App（本地集成）性能分析

远程日志

iOS App
- iOS App（Pod集成）远程日志
- iOS App（手动集成）远程日志

Android App
- Android App（Maven集成）远程日志
- Android App（本地集成）远程日志

3.2. 崩溃分析

3.2.1. Android SDK接入（Maven集成）

本文介绍如何通过Maven集成方式添加依赖接入崩溃分析服务的Android SDK。

> 说明
> - 崩溃分析服务的Android SDK接入可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用本地集成方式添加依赖，操作方法参见Android SDK接入（本地集成）。

前提条件

已获取AppKey、AppSecret，获取方式请参见EMAS 快速入门>下载配置文件。

使用限制

- 推荐使用Gradle管理依赖的Android Studio项目。
- 仅支持Android 4.0及以上版本。
- 仅支持arm64-v8a//armeabi-v7a/x86/x86_64架构。

操作步骤

1. **添加依赖**
   在build.gradle项目文件中进行以下操作：
   i. 添加阿里云Maven仓库地址。

```
repositories {
    maven { url "http://maven.aliyun.com/nexus/content/repositories/releases" }
}
```

   ii. 在 `android{}` 代码段修改默认配置项。

```
android {
    ......
    defaultConfig {
        applicationId "com.xxx.xxx"
        ......
        ndk {
            //选择要添加的对应cpu类型的.so库，当前支持四种
            abiFilters 'arm64-v8a',  'armeabi-v7a', 'x86', 'x86_64'
        }
        ......
    }
    ......
}
```

   iii. 在 `dependencies{}` 代码段添加SDK依赖。

```
dependencies {
    ......
    compile('com.aliyun.ams:alicloud-android-ha-adapter:1.1.5.1-open')

    compile('com.aliyun.ams:alicloud-android-ha-crashreporter:1.2.5')
    ......
}
```

2. **接入服务**

i. 定义Application类，编写onCreate方法，启动服务。

> **说明**
> 建议将崩溃分析服务的SDK初始化代码段，放在所有业务代码之前，确保App在启动时，优先加载崩溃分析服务，保障后续崩溃的信息，可以即时获取并上传至控制台。

```
public class MyApplication extends Application {
   @Override
   public void onCreate() {
       initHa();
   }
   private void initHa() {
        AliHaConfig config = new AliHaConfig();
        config.appKey = "xxxxxxxx";
        config.appVersion = "x.xx";
        config.appSecret = "xxxxxxxxxxxx";
        config.channel = "mqc_test";
        config.userNick = null;
        config.application = this;
        config.context = getApplicationContext();
        config.isAliyunos = false;
        //启动CrashReporter
        AliHaAdapter.getInstance().addPlugin(Plugin.crashreporter);
        AliHaAdapter.getInstance().start(config);
   }
}
```

参数	说明
appKey	用于指定App的AppKey。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无

参数	说明
appVersion	用于设置App的版本号。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度。 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
appSecret	用于指定App的AppSecret。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
channel	用于设置渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
userNick	用于设置用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。 【命名规范】自定义

参数	说明
application	用于指定本应用。 **注意** 不能指向其他应用。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
context	用于指定App的上下文对象，设置 `getApplicationContext();` 即可。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
isAliyunos	用于判断App所在平台是否为YunOS。 【数据类型】布尔型 【取值范围】false/true 【是否必选】否 【是否可为空】是 【默认值】fal

注意

- 为避免在日志中泄漏参数 `appkey` / `appsecret` 或App运行过程中产生的数据，建议线上版本关闭SDK调试日志。
- 由于所有用户使用统一的SDK接入，在接入过程中需要在代码中设置 `appkey` / `appsecret` 参数，而此类参数与计量计费密切相关，为防止恶意反编译获取参数造成信息泄漏，建议您开启混淆，并进行App加固后再发布上线。

ii. 在AndroidManifest.xml中添加代码段注册Application。

```
<application
        android:name=".MyApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
</application>
```

3. **添加高级设置**
   Android SDK提供接口，用于上报自定义信息/错误。

```
//上报自定义信息
AliHaAdapter.getInstance().addCustomInfo("key", "value"); //配置项：自定义环境信息

//按异常类型上报自定义信息
AliHaAdapter.getInstance().setErrorCallback(new ErrorCallback() {
    @Override
    public Map<String, String> onError(ErrorInfo callbackInfo) {
        Map<String, String> infos = new HashMap<>();
        infos.put("key", "value"); //配置项：异常信息
        return infos;
    }
});

//上报自定义错误
AliHaAdapter.getInstance().reportCustomError(new RuntimeException("custom error")); //
配置项：自定义错误
```

具体说明参见：Android SDK接口说明

4. **混淆配置**
   如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
#keep crashreporter
-keep class com.alibaba.motu.crashreporter.**{ *;}
-keep class com.uc.crashsdk.**{*;}
-keep interface com.ut.mini.crashhandler.*{*;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable
,*Annotation*,EnclosingMethod
```

5. **编译**
   如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败，请参见SDK UTDID冲突解决方案解决。

6. **接入验证**
   Android SDK接入操作完成后，需进行功能验证。

   i. 编写测试代码，模拟/触发移动端崩溃。例如：

```
throw new NullPointerException();
```

   ii. 重启移动端，大概2分钟后在控制台查看是否显示崩溃信息。

> **说明**
> 崩溃数据从采集到上传到控制台显示，存在大约2~3分钟延迟。

   - 显示崩溃数据：SDK接入成功
   - 数据未显示：按照c步骤进行排查

   iii. 在模拟/触发崩溃及重启移动设备期间，使用Charles抓包，查看能否捕获包含 `https://adash-emas.cn-hangzhou.aliyuncs.com/upload` 的HTTP请求：

   - 捕获：崩溃信息已上报。可能原因：后端未接入；Appkey/Secret信息有误。
   - 未捕获：崩溃信息未上报。可能原因：SDK接入失败；SDK未捕获崩溃；数据发送失败。请联系技术支持解决。

样例代码

崩溃分析服务Android SDK接入工程样例参见：Demo工程

常见问题

- App启动后崩溃闪退，但相关信息未上报至控制台
- 接入服务时，提示缺失so文件：libcrashsdk.so
- SDK UTDID冲突解决方案

3.2.2. Android SDK接入（本地集成）

本文介绍如何通过本地集成方式添加依赖接入崩溃分析服务的Android SDK。

> 说明
>
> - 崩溃分析服务的Android SDK接入可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用Maven集成方式添加依赖，操作方法参见Android SDK接入（Maven集成）。
> - 崩溃分析服务新增网络性能监控功能已进行灰度发布，如需体验，可联系技术支持获取SDK包。

前提条件

- 已获取AppKey、AppSecret，请参见快速入门：创建监控任务中下载的配置文件。
- 已下载崩溃分析SDK，具体操作，请参见下载SDK。

使用限制

- 仅支持Android 4.0及以上版本。
- 仅支持arm64-v8a、armeabi、armeabi-v7a和x86架构。

操作步骤

1. **添加依赖。**
   i. 将SDK包内所有文件拷贝至项目libs目录下。
   ii. 在项目级 `build.gradle` 项目文件中，添加本地SDK文件目录地址。

```
repositories {
   flatDir {
       dirs 'libs'
   }
}
```

iii. 在应用级 `build.gradle` 项目文件的 `dependencies{}` 代码段添加SDK依赖。

```
    //1、本地jar库引入。
    compile fileTree(include: ['*.jar'], dir: 'libs')
    //2、公共库。
    compile (name: 'alicloud-android-ha-adapter-1.1.3.7-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-core-1.1.0.6.1-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-protocol-1.1.1.0-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-tbrest-1.1.1.0-open', ext: 'aar')
    compile (name: 'alicloud-android-utdid-2.5.1-proguard', ext: 'jar')
    compile (name: 'fastjson-1.1.54.android', ext: 'jar')
    //3、崩溃分析。
    compile (name: 'alicloud-android-ha-crashreporter-1.2.4', ext: 'aar')
    compile (name: 'alicloud-android-ha-watch-1.1.0.6.1-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-bizerrorreporter-1.1.0.9-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-olympic-1.0.4.37', ext: 'aar')
```

2. **接入服务。**

i. 定义Application类，编写onCreate方法，启动服务。

> **说明**
> 建议将崩溃分析服务的SDK初始化代码段，放在所有业务代码之前，确保App在启动时，优先加载崩溃分析服务，保障后续崩溃的信息，可以即时获取并上传至控制台。

```
public class MyApplication extends Application {
   @Override
   public void onCreate() {
       initHa();
   }
   private void initHa() {
        AliHaConfig config = new AliHaConfig();
        config.appKey = "xxxxxxxx";
        config.appVersion = "x.xx";
        config.appSecret = "xxxxxxxxxxxx";
        config.channel = "mqc_test";
        config.userNick = null;
        config.application = this;
        config.context = getApplicationContext();
        config.isAliyunos = false;
        //启动CrashReporter。
        AliHaAdapter.getInstance().addPlugin(Plugin.crashreporter);
        AliHaAdapter.getInstance().start(config);
   }
}
```

配置说明如下：

参数	说明

参数	说明
appKey	用于指定App的AppKey。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
appVersion	用于设置App的版本号。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度。 说明 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 说明 不支持中文字符、特殊字符。
appSecret	用于指定App的AppSecret。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
channel	用于设置渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 说明 不支持中文字符、特殊字符。

参数	说明
userNick	用于设置用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 说明 不支持中文字符、特殊字符。 【命名规范】自定义
application	用于指定本应用。注意：不能指向其他应用。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
context	用于指定App的上下文对象，设置getApplicationContext();即可。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
isAliyunos	用于判断App所在平台是否为YunOS。 【数据类型】布尔型 【取值范围】false或true 【是否必选】否 【是否可为空】是 【默认值】false

ii. 在Androidmanifest.xml中添加代码段注册Application。

```
<application
        android:name=".MyApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
</application>
```

3. **添加高级设置。**
   Android SDK提供接口，用于上报自定义信息或错误。

```
//上报自定义信息。
AliHaAdapter.getInstance().addCustomInfo("key", "value");     //配置项：自定义环境信息。

//按异常类型上报自定义信息。
AliHaAdapter.getInstance().setErrorCallback(new ErrorCallback() {
    @Override
    public Map<String, String> onError(ErrorInfo callbackInfo) {
        Map<String, String> infos = new HashMap<>();
        infos.put("key", "value");     //配置项：异常信息。
        return infos;
    }
});

//上报自定义错误。
AliHaAdapter.getInstance().reportCustomError(new RuntimeException("custom error"));
//配置项：自定义错误。
```

具体说明参见：Android SDK接口说明

4. **混淆配置。**
   如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
#keep crashreporter
-keep class com.alibaba.motu.crashreporter.MotuCrashReporter{ *;}
-keep class com.alibaba.motu.crashreporter.ReporterConfigure{*;}
-keep class com.alibaba.motu.crashreporter.utrestapi.UTRestReq{*;}
-keep interface com.alibaba.motu.crashreporter.IUTCrashCaughtListener{*;}
-keep interface com.alibaba.motu.crashreporter.ICrashReportSendListener{*;}
-keep interface com.alibaba.motu.crashreporter.ICrashReportDataListener{*;}
-keep interface com.ut.mini.crashhandler.*{*;}
-keep class com.uc.crashsdk.**{*;}
-keep class com.alibaba.motu.crashreporter.YouKuCrashReporter{public *;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable
,*Annotation*,EnclosingMethod
```

5. **编译。**
   如同时使用其他阿里云产品，可能会因为依赖中存在UT DID冲突，造成编译失败，请参见SDK UT DID冲突解决方案解决。

6. **接入验证。**
   Android SDK接入操作完成后，需进行功能验证。

   i. 编写测试代码，模拟或触发移动端崩溃。例如：

   ```
   throw new NullPointerException();
   ```

   ii. 重启移动端，大概2分钟后在控制台查看是否显示崩溃信息。

   > **说明**
   > 崩溃数据从采集到上传到控制台显示，存在大约2~3分钟延迟。

   - 显示崩溃数据：SDK接入成功
   - 数据未显示：按照c步骤进行排查

iii. 在模拟或触发崩溃及重启移动设备期间，使用Charles抓包，查看能否捕获包含 `https://adash-emas.cn-hangzhou.aliyuncs.com/upload` 的HTTP请求：

- 捕获：崩溃信息已上报。可能原因：后端未接入；Appkey或Secret信息有误。
- 未捕获：崩溃信息未上报。可能原因：SDK接入失败；SDK未捕获崩溃；数据发送失败。请联系技术支持解决。

样例代码

崩溃分析服务Android SDK接入工程样例参见：Demo工程

3.2.3. Android SDK接口说明

本文介绍Android SDK提供的接口。

上报自定义信息

用于在移动应用发生Crash时，保存自定义环境信息，并随异常信息上报至控制台。

```
AliHaAdapter.getInstance().addCustomInfo("key", "value"); //配置项：自定义环境信息
```

> **说明**
> key和value的字符串长度，合计须小于10K；否则，超出部分将被丢弃。

按异常类型上报自定义信息

用于在移动应用发生Crash时，针对特定异常类型，保存自定义环境信息，并随异常信息上报至控制台。

```
AliHaAdapter.getInstance().setErrorCallback(new ErrorCallback() {
    @Override
    public Map<String, String> onError(ErrorInfo callbackInfo) {
        Map<String, String> infos = new HashMap<>();
        infos.put("key", "value"); //配置项：异常信息
        return infos;
    }
});
```

其中，Crash回调类定义：

```
public interface ErrorCallback {
    Map<String, String> onError(ErrorInfo var1);
}
```

异常类型参照ErrorInfo定义：

```
public class ErrorInfo {
    /** Java crash */
    public static final int HA_CRASH_JAVA = 1;
    /** Native crash */
    public static final int HA_CRASH_NATIVE = 2;
    /** ANR */
    public static final int HA_CRASH_ANR = 3;
    /** Memory leak */
    public static final int HA_MEM_LEAK = 4;
    /** Main thread block */
    public static final int HA_MAIN_THREAD_BLOCK = 5;
    /** Main thread io */
    public static final int HA_MAIN_THREAD_IO = 6;
    /** Big bitmap*/
    public static final int HA_BIG_BITMAP = 7;
    /** File description over flow*/
    public static final int HA_FD_OVERFLOW = 8;
    /** Resource leak */
    public static final int HA_RESOURCE_LEAK = 9;
    /** Custom error */
    public static final int HA_CUSTOM_ERROR = 10;

    /**
     * 获取错误类型
     * @return 返回错误类型
     */
    public int getErrorType() {
        return this.mErrorType;
    }

    /**
     * 获取异常信息
     * @return 返回异常实例
     */
    public Throwable getThrowable() {
        return this.mThrowable;
    }
}
```

上报自定义错误

用于将自定义错误上报至控制台。

```
AliHaAdapter.getInstance().reportCustomError(new RuntimeException("custom error")); //配置项：自定义错误
```

3.2.4. iOS SDK接入（Pod集成）

本文介绍如何使用Pod集成方式接入崩溃分析服务的iOS SDK。

> 说明
> - iOS SDK接入可采用Pod集成和手动集成2种方式。推荐使用Pod集成方式接入崩溃分析服务，可大幅简化接入操作。
> - 如需使用手动集成方式接入崩溃分析服务的iOS SDK，操作方法参见iOS SDK接入（手动集成）。

前提条件

已下载配置文件，请参见快速入门：创建监控任务。

使用限制

- 仅支持iOS 8.0及以上的App。
- 推荐使用CocoaPods管理依赖的Xcode项目。

操作步骤

1. **添加依赖。**
   i. 指定官方仓库和阿里云仓库。

```
source "https://github.com/CocoaPods/Specs.git"
source "https://github.com/aliyun/aliyun-specs.git"
```

   ii. 添加崩溃分析服务依赖。

```
pod 'AlicloudCrash' , '~> 1.2.0'
```

> 说明
> 执行 `pod search AlicloudCrash` 命令，查询AlicloudCrash最新版本。

   iii. 执行 `pod update` 命令，保存设置。

2. **接入服务。**
   i. 将iOS配置文件AliyunEmasService-Info.plist拷贝至项目根目录。
   ii. 在AppDelegate文件的application:didFinishLaunchingWithOptions方法中初始化SDK。
   引入头文件：

```
#import <AlicloudCrash/AlicloudCrashProvider.h>
#import <AlicloudHAUtil/AlicloudHAProvider.h>
```

   添加代码段：

```
NSString *appVersion = @"x.x"; //app版本，会上报。
NSString *channel = @"xx";     //渠道标记，自定义，会上报。
NSString *nick = @"xx";        //nick 昵称，自定义，会上报。
[[AlicloudCrashProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:nick];
[AlicloudHAProvider start];
```

参数	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。 【命名规范】自定义

3. **添加高级设置。**
   iOS SDK提供接口，用于上报自定义信息或错误。

```
//上报自定义信息。
[AlicloudCrashProvider configCustomInfoWithKey:@"key" value:@"value"];   //配置项：自定义环境信息（configCustomInfoWithKey/value）。

//按异常类型上报自定义信息。
[AlicloudCrashProvider setCrashCallBack:^NSDictionary * _Nonnull(NSString * _Nonnull type) {
    return @{@"key":@"value"};   //配置项：异常信息（key/value）。
}];

//上报自定义错误。
NSError *error = [NSError errorWithDomain:@"customError" code:10001 userInfo:@{@"errorInfoKey":@"errorInfoValue"}];
[AlicloudCrashProvider reportCustomError:error];   //配置项：自定义错误信息（errorWithDomain/code/userInfo）。
```

具体说明参见：iOS SDK接口说明

4. **编译。**

   i. **在项目的 `Build Setting` 中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。**

   ii. **执行编译。**

> **说明**
>
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案。

5. **功能验证。**

   iOS SDK接入操作完成后，需进行功能验证。

   i. 编写测试代码，模拟或触发移动端崩溃。例如：

```
NSMutableArray *array = @[];
[array addObject:nil];
```

> **说明**
>
> 更多内容参考样例代码。

   ii. 重启移动端大概2分钟后在控制台查看是否显示崩溃信息。

> **说明**
>
> 崩溃数据从采集到上传到控制台显示，存在大约2~3分钟延迟。

   - 显示崩溃数据：SDK接入成功
   - 数据未显示：按照c步骤进行排查

iii. 在模拟或触发崩溃及重启移动设备期间，使用Charles抓包，查看能否捕获包含 `https://adash-emas.cn-hangzhou.aliyuncs.com/upload` 的HTTP请求：

- 捕获：崩溃信息已上报。可能原因：后端未接入；Appkey或Secret信息有误。
- 未捕获：崩溃信息未上报。可能原因：SDK接入失败；SDK未捕获崩溃；数据发送失败。请联系技术支持解决。

样例代码

崩溃分析服务Android SDK接入工程样例参见：Demo工程

3.2.5. iOS SDK接入（手动集成）

本文介绍如何使用手动集成方式接入崩溃分析服务的iOS SDK。

前提条件

- 已下载应用配置文件，请参见快速入门：创建监控任务。
- 已下载崩溃分析SDK，请参见下载SDK。

使用限制

仅支持iOS 8.0及以上的App。

操作步骤

1. **集成SDK。**

   i. **在Xcode中，将下载好的SDK目录中的framework文件拖入Target目录，在弹出框选中 `Copy items if needed` 选项。**

   ii. 打开**Build Phases > Link Binary With Libraries**，添加Xcode自带的公共包文件：

   - libc++.tbd
   - SystemConfiguration.framework

2. **接入服务。**

   i. 将iOS配置文件 `AliyunEmasServices-Info.plist` 拷贝至项目根目录。

   ii. 在 `AppDelegate.m` 文件的 `application:didFinishLaunchingWithOptions` 方法中初始化SDK。

   引入头文件：

   ```
   #import <AlicloudCrash/AlicloudCrashProvider.h>
   #import <AlicloudHAUtil/AlicloudHAProvider.h>
   ```

   添加代码段：

   ```
   NSString *appVersion = @"x.x"; //app版本，会上报。
   NSString *channel = @"xx";     //渠道标记，自定义，会上报。
   NSString *nick = @"xx";        //nick 昵称，自定义，会上报。
   [[AlicloudCrashProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:nick];
   [AlicloudHAProvider start];
   ```

   参数说明：

参数	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。 【命名规范】自定义

3. **添加高级设置。**
   iOS SDK提供接口，用于上报自定义信息或错误。

```
//上报自定义信息。
[AlicloudCrashProvider configCustomInfoWithKey:@"key" value:@"value"];   //配置项：自定义环境信息（configCustomInfoWithKey/value）。

//按异常类型上报自定义信息。
[AlicloudCrashProvider setCrashCallBack:^NSDictionary * _Nonnull(NSString * _Nonnull type) {
    return @{@"key":@"value"};   //配置项：异常信息（key/value）。
}];

//上报自定义错误。
NSError *error = [NSError errorWithDomain:@"customError" code:10001 userInfo:@{@"errorInfoKey":@"errorInfoValue"}];
[AlicloudCrashProvider reportCustomError:error];   //配置项：自定义错误信息（errorWithDomain/code/userInfo）。
```

具体说明参见：iOS SDK接口说明

4. **编译。**
   i. **在项目的 `Build Setting` 中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。**
   ii. 执行编译。

> **说明**
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案。

5. **功能验证。**
   iOS SDK接入操作完成后，需进行功能验证。
   i. 编写测试代码，模拟或触发移动端崩溃。例如：

```
NSMutableArray *array = @[];
[array addObject:nil];
```

> **说明**
> 更多内容参考样例代码。

   ii. 重启移动端大概2分钟后在控制台查看是否显示崩溃信息。

> **说明**
> 崩溃数据从采集到上传到控制台显示，存在大约2~3分钟延迟。

   - 显示崩溃数据：SDK接入成功
   - 数据未显示：按照c步骤进行排查

iii. 在模拟或触发崩溃及重启移动设备期间，使用Charles抓包，查看能否捕获包含 `https://adash-emas.cn-hangzhou.aliyuncs.com/upload` 的HTTP请求：

- 捕获：崩溃信息已上报。可能原因：后端未接入；Appkey或Secret信息有误。
- 未捕获：崩溃信息未上报。可能原因：SDK接入失败；SDK未捕获崩溃；数据发送失败。请联系技术支持解决。

样例代码

崩溃分析服务Android SDK接入工程样例参见：Demo工程

3.2.6. iOS SDK接口说明

本文介绍iOS SDK提供的接口。

上报自定义信息

用于在移动应用发生Crash时，保存自定义环境信息，并随异常信息上报至控制台。
接口定义：

```
/*!
* @brief 设置用户信息
* @details 设置用户信息，崩溃时带上。总数据量要求小于10Kb
* @param key key
* @param value value
*/
+ (void)configCustomInfoWithKey:(NSString *)key value:(NSString *)value;
```

使用示例：

```
[AlicloudCrashProvider configCustomInfoWithKey:@"key" value:@"value"];//配置项：自定义环境信息（configCustomInfoWithKey/value）
```

> **说明**
> key和value的字符串长度，合计须小于10Kb；否则，超出部分将被丢弃。

按异常类型上报自定义信息

用于在移动应用发生Crash时，针对特定异常类型，保存自定义环境信息，并随异常信息上报至控制台。
接口定义：

```
/*!
* @brief 设置崩溃回调信息
* @details 设置崩溃回调信息
* @param crashReporterAdditionalInformationCallBack 返回字典不超过10kb，不要有耗时操作，只支持字符串
*/
+ (void)setCrashCallBack:(NSDictionary * (^)(NSString * type))crashReporterAdditionalInformationCallBack;
```

使用示例：

```
[AlicloudCrashProvider setCrashCallBack:^NSDictionary * _Nonnull(NSString * _Nonnull type)
{
    return @{@"key":@"value"};//配置项：异常信息（key/value）
}];
```

上报自定义错误

用于将自定义错误上报至控制台。

接口定义：

```
/*!
* @brief 用户自定义错误上报
* @details 用户自定义错误上报
* @param error 用户把错误封装为标准NSError
*/
+ (void)reportCustomError:(NSError *)error;
```

使用示例：

```
NSError *error = [NSError errorWithDomain:@"customError" code:10001 userInfo:@{@"errorInfoKey":@"errorInfoValue"}];
[AlicloudCrashProvider reportCustomError:error];//配置项：自定义错误信息（errorWithDomain/code/userInfo）
```

3.3. 性能分析

3.3.1. Android SDK接入（Maven集成）

本文介绍如何通过Maven集成方式添加依赖接入性能分析服务的Android SDK。

> **说明**
>
> - 性能分析服务的Android SDK接入可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用本地集成方式添加依赖，操作方法参见：Android SDK接入（本地集成）

前提条件

已下载Android配置文件，并在文件中获取了AppKey、AppSecret和PackageName。具体操作，请参见快速入门：创建监控任务

使用限制

- 仅支持Android 4.2及以上版本。
- 仅支持arm64-v8a、armeabi、armeabi-v7a和x86架构。
- 仅支持gradle 3.0.0及以上版本。
- 仅支持以下网络库版本：
  - okhttp2：2.0.0-2.7.5
  - okhttp3: 3.0.0-3.14.7
  - okhttp4: 4.0.0-4.8.1

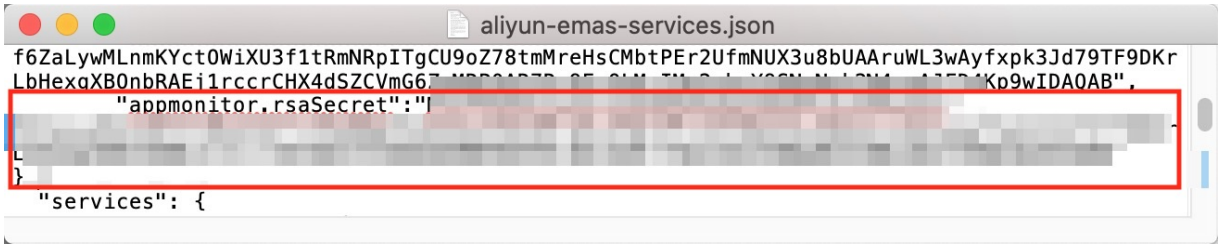
接入概述

1. 准备：获取公钥。
2. 应用插件：添加插件依赖；应用插件。
3. 添加依赖：采用Maven方式集成SDK。SDK功能变更历史参见：SDK版本说明
4. 接入服务：添加自定义Application，以及SDK初始化代码。
5. 混淆配置：如App对代码进行乱序混淆，则修改混淆配置文件。
6. 编译：常见编译问题排查。

准备

打开Android配置文件，查询 `appmonitor.rsaSecret` 字段内容，即为性能分析公钥。

> **说明**
> Android配置文件的获取方式参见：前提条件



应用插件

1. 在项目级 `build.gradle` 项目文件的 `buildscript{}` 代码段添加插件依赖。

> **注意**
> 此处仅添加插件依赖，应用依赖须添加至 `build.gradle` 项目文件的 `dependencies{}` 代码段。
> 具体说明参见：添加依赖

```
buildscript {
    repositories {
        google()
        jcenter()
    }
dependencies {
    classpath 'com.android.tools.build:gradle:${gradle-version}'
    classpath 'com.aliyun.ams:alicloud-android-networkmonitor-plugin:1.2.0-open'}
}
```

配置说明如下：

配置项	说明

配置项	说明
gradle-version	用于指定实际开发环境的gradle版本。 【示例】 `classpath 'com.android.tools.build:gradle:4.8.1'`

2. 在应用级 `build.gradle` 项目文件中添加代码段应用插件。

```
apply plugin: 'com.aliyun.emas.networkmonitor'
```

添加依赖

1. 在项目级 `build.gradle` 项目文件的 `repositories{}` 代码段中添加阿里云Maven仓库地址。

```
repositories {
    maven { url "http://maven.aliyun.com/nexus/content/repositories/releases" }
}
```

2. 在应用级 `build.gradle` 项目文件的 `android{}` 代码段中设置应用包名和.so库。

```
android {
    ......
    defaultConfig {
        applicationId "com.xxx.xxx"
        ......
        ndk {
            //选择要添加的对应cpu类型的.so库，当前支持四种。
            abiFilters 'arm64-v8a', 'armeabi', 'armeabi-v7a', 'x86'
        }
        ......
    }
    ......
}
```

配置说明如下：

参数	说明
applicationId	用于指定App的PackageName。 【数据类型】字符串 【获取方式】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无

参数	说明
ndk	用于指定App的架构，添加对应CPU类型的.so库。 【数据类型】枚举型 【取值范围】 ○ arm64-v8a ○ armeabi ○ armeabi-v7a ○ x86 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】否

3. 在应用级 `build.gradle` 项目文件的 `dependencies{}` 代码段中添加SDK依赖。

```
dependencies {
    ......
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "com.squareup.okhttp3:okhttp:${okhttp3version}"

    //线上测试。
    implementation 'com.aliyun.ams:alicloud-android-ha-adapter:1.1.3.8-open'
    implementation 'com.aliyun.ams:alicloud-android-apm:1.0.8.2-open'
    ......
}
```

配置说明如下：

参数	说明
okhttp3version	用于设置本地开发环境可支持的网络库版本。 【示例】 `implementation "com.squareup.okhttp3:okhttp:4.8.1"`

接入服务

1. 定义 `Application` 类，编写 `onCreate` 方法，启动服务。

```
public class MyApplication extends Application {
        @Override
        public void onCreate() {
            initHa();
        }
        private void initHa() {
             AliHaConfig config = new AliHaConfig();
             config.appKey = "xxxxxxxx";
             config.appVersion = "x.xx";
             config.appSecret = "xxxxxxxxxxxx";
             config.channel = "mqc_test";
             config.userNick = null;
             config.application = this;
             config.context = getApplicationContext();
             config.isAliyunos = false;
             config.rsaPublicKey = "xxxxxxx";
             AliHaAdapter.getInstance().addPlugin(Plugin.apm);
             AliHaAdapter.getInstance().start(config);
        }
    }
```

配置说明如下：

参数	说明
okhttp3version	用于设置本地开发环境可支持的网络库版本。 【示例】 `implementation "com.squareup.okhttp3:okhttp:4.8.1"`

2. 在 `AndroidManifest.xml` 中添加代码段注册Application。

```
<application
        android:name=".MyApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
</application>
```

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
-keep class com.taobao.monitor.APMLauncher{*;}
-keep class com.taobao.monitor.impl.logger.Logger{*;}
-keep class com.taobao.monitor.impl.logger.IDataLogger{*;}
-keep class com.taobao.monitor.impl.data.AbsWebView{*;}
-keep class com.taobao.monitor.impl.data.GlobalStats{*;}
-keep class com.taobao.monitor.impl.common.Global{*;}
-keep class com.taobao.monitor.impl.data.WebViewProxy{*;}
-keep class com.taobao.monitor.impl.logger.Logger{*;}
-keep class com.taobao.monitor.impl.processor.pageload.IProcedureManager{*;}
-keep class com.taobao.monitor.impl.processor.pageload.ProcedureManagerSetter{*;}
-keep class com.taobao.monitor.impl.util.TimeUtils{*;}
-keep class com.taobao.monitor.impl.util.TopicUtils{*;}
-keep class com.taobao.monitor.impl.common.DynamicConstants{*;}
-keep class com.taobao.application.common.data.DeviceHelper{*;}
-keep class com.taobao.application.common.impl.AppPreferencesImpl{*;}
-keep class com.taobao.monitor.impl.processor.launcher.PageList{*;}
-keep class com.taobao.monitor.impl.processor.fragmentload.FragmentInterceptorProxy{*;}
-keep class com.taobao.monitor.impl.processor.fragmentload.IFragmentInterceptor{*;}
-keep class com.taobao.monitor.impl.logger.DataLoggerUtils{*;}
-keep interface com.taobao.monitor.impl.data.IWebView{*;}
-keep interface com.taobao.monitor.impl.processor.IProcessor{*;}
-keep interface com.taobao.monitor.impl.processor.IProcessorFactory{*;}
-keep interface com.taobao.monitor.impl.logger.IDataLogger{*;}
-keep interface com.taobao.monitor.impl.trace.IDispatcher{*;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable,*An
notation*,EnclosingMethod
```

编译

如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

样例代码

性能分析服务Android SDK接入工程样例参见：Demo工程

功能验证

Android SDK接入操作完成后，可操作App，查看性能分析服务控制台显示数据，进行功能验证。

1. 手机端：启动App。（2分钟后）控制台：查看**概览**页签的**启动速度**是否显示数据。
2. 手机端：在App中跳转几个页面。（2分钟后）控制台：查看**概览**页签的**加载时间**是否显示数据。

> **说明**
> 数据从App采集到控制台显示，存在大约2分钟延迟。

如数据显示正常，则Android SDK接入成功。
否则，可能的原因是：SDK接入失败、SDK未获取数据、数据发送失败、后端问题，请联系技术支持解决。

SDK版本说明

版本号	发布日期	变更说明
1.0.8.2-open	2020-12-18	代码优化。

版本号	发布日期	变更说明
1.0.8.1-open	2020-11-24	优化编译构建问题。
1.0.8.0-open	2020-11-12	新增网络异常分析功能。
其他历史版本。		

3.3.2. Android SDK接入（本地集成）

本文介绍如何通过本地集成方式添加依赖接入性能分析服务的Android SDK。

> 说明
> - 性能分析服务的Android SDK接入可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用Maven集成方式添加依赖，操作方法参见：Android SDK接入（Maven集成）

前提条件

已下载Android配置文件，并在文件中获取AppKey、AppSecret和PackageName。具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持Android 4.2及以上版本。
- 仅支持arm64-v8a、armeabi、armeabi-v7a和x86架构。
- 仅支持gradle 3.0.0及以上版本。
- 仅支持以下网络库版本：
  - okhttp2：2.0.0-2.7.5
  - okhttp3: 3.0.0-3.14.7
  - okhttp4: 4.0.0-4.8.1

接入概述

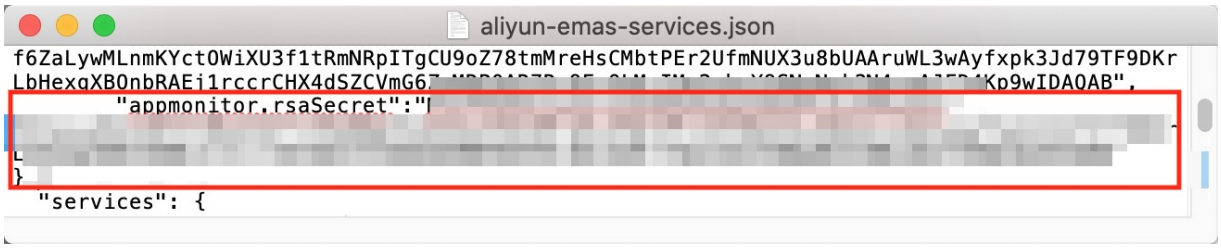
1. 准备：获取公钥；下载SDK包。
2. 应用插件：添加插件依赖；应用插件。
3. 添加依赖：采用本地方式集成SDK。
4. 接入服务：添加自定义Application，以及SDK初始化代码。
5. 混淆配置：如App对代码进行乱序混淆，则修改混淆配置文件。
6. 编译：常见编译问题排查。

准备

获取性能分析公钥

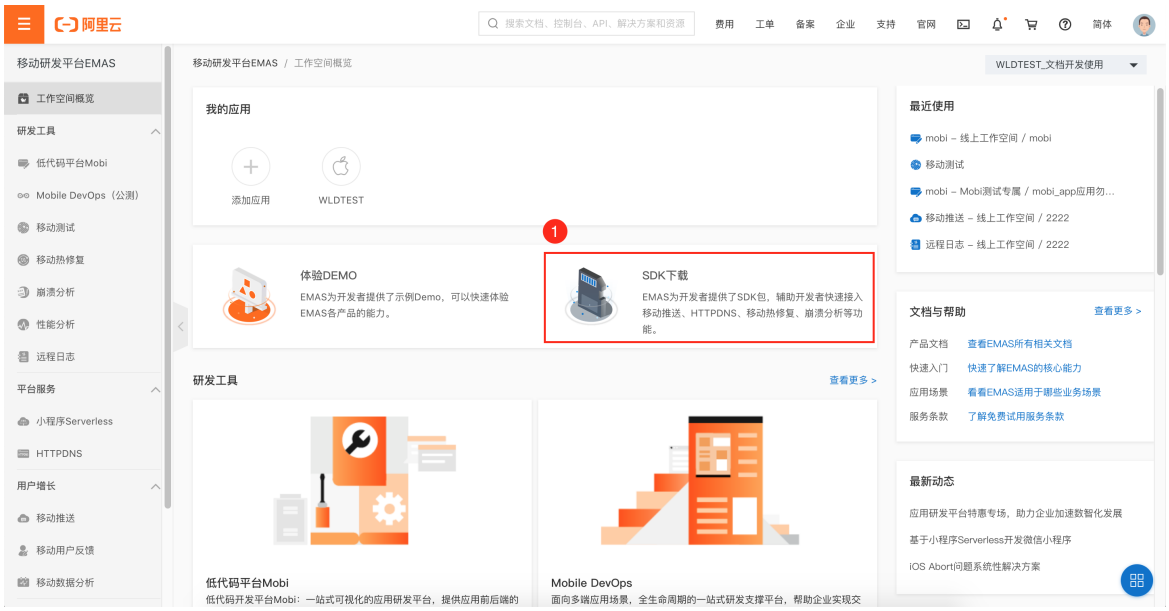
打开Android配置文件，查询 `appmonitor.rsaSecret` 字段内容，即为性能分析公钥。

> **说明**
> Android配置文件的获取方式参见：前提条件



下载SDK包

1. 登录移动研发平台的管理控制台，默认打开**我的工作空间**页面。
2. 在**我的工作空间**页面，单击工作空间标签的**进入**按钮，打开指定工作空间的概览页面。
3. 在工作空间概览页面，单击**SDK下载**区域，打开**SDK下载**右侧栏。



4. 在**SDK下载**右侧栏，选中**性能分析**复选框，单击**下载Android版本**按钮，下载性能分析服务的SDK包。

> **说明**
> 在**性能分析**行的**Android版**列，单击版本号链接，可查看版本变更记录。

5. 检查SDK包，确保内容完整。SDK包文件列表如下：

- alicloud-android-apm-1.0.8.2-open.aar
- alicloud-android-ha-adapter-1.1.3.8-open.aar
- alicloud-android-ha-apm-impl-1.0.7.1-open.aar
- alicloud-android-ha-core-1.1.0.6.1-open.aar
- alicloud-android-ha-fulltrace-1.0.1.12.aar
- alicloud-android-ha-protocol-1.1.1.0-open.aar
- alicloud-android-ha-tbrest-1.1.1.0-open.aar
- alicloud-android-networkmonitor-1.4.0.aar
- alicloud-android-rest-1.4.0-open.aar
- alicloud-android-utdid-2.5.1-proguard.jar
- fastjson-1.1.54.android.jar

应用插件

1. 在项目级 `build.gradle` 项目文件的 `buildscript{}` 代码段添加插件依赖。

> **说明**
> 此处仅添加插件依赖，应用依赖须添加至 `build.gradle` 项目文件的 `dependencies{}` 代码段。
> 具体说明参见：添加依赖

```
buildscript {
    repositories {
        google()
        jcenter()
    }
dependencies {
    classpath 'com.android.tools.build:gradle:${gradle-version}'
    classpath 'com.aliyun.ams:alicloud-android-networkmonitor-plugin:1.2.0-open'}
}
```

配置说明如下：

配置项	说明
gradle-version	用于指定实际开发环境的gradle版本。 【示例】 `classpath 'com.android.tools.build:gradle:4.8.1'`

2. 在应用级 `build.gradle` 项目文件中添加代码段应用插件。

```
apply plugin: 'com.aliyun.emas.networkmonitor'
```

添加依赖

1. 将SDK包内所有文件拷贝至项目的libs目录下。
2. 在项目级 `build.gradle` 项目文件中，添加本地SDK文件目录地址。

```
repositories {
   flatDir {
       dirs 'libs'
   }
}
```

3. 在应用级 `build.gradle` 项目文件的 `dependencies{}` 代码段，添加SDK依赖。

```
    //1、本地jar库引入。
    compile fileTree(include: ['*.jar'], dir: 'libs')
    //2、公共库。
    compile (name: 'alicloud-android-ha-adapter-1.1.3.8-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-core-1.1.0.6.1-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-protocol-1.1.1.0-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-tbrest-1.1.1.0-open', ext: 'aar')
    compile (name: 'alicloud-android-utdid-2.5.1-proguard', ext: 'jar')
    compile (name: 'fastjson-1.1.54.android', ext: 'jar')
    //3、性能监控，不接入可注释掉。
    compile (name: 'alicloud-android-ha-apm-1.0.8.2-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-apm-impl-1.0.7.1-open', ext: 'aar')
    compile (name: 'alicloud-android-ha-fulltrace-1.0.1.12', ext: 'aar')
    compile (name: 'alicloud-android-networkmonitor-1.4.0', ext: 'aar)
    compile (name: 'alicloud-android-rest-1.4.0-open', ext: 'aar)
```

接入服务

1. 定义 `Application` 类，编写 `onCreate` 方法，启动服务。

```
public class MyApplication extends Application {
        @Override
        public void onCreate() {
            initHa();
        }
        private void initHa() {
             AliHaConfig config = new AliHaConfig();
             config.appKey = "xxxxxxxx";
             config.appVersion = "x.xx";
             config.appSecret = "xxxxxxxxxxxx";
             config.channel = "mqc_test";
             config.userNick = null;
             config.application = this;
             config.context = getApplicationContext();
             config.isAliyunos = false;
             config.rsaPublicKey = "xxxxxxx";
             AliHaAdapter.getInstance().addPlugin(Plugin.apm);
             AliHaAdapter.getInstance().start(config);
        }
    }
```

配置说明如下：

参数	说明
appKey	用于指定App的AppKey。 【数据类型】字符串 【获取方式】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
appVersion	用于设置App的版本号。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字 **说明** 该参数不支持中文字符、特殊字符。

参数	说明
appSecret	用于指定App的AppSecret。 【数据类型】字符串 【获取方式】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
channel	用于设置渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字 **说明** 该参数不支持中文字符、特殊字符。
userNick	用于设置用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义
application	用于指定本应用。 **注意** 该参数不能指向其他应用。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
context	用于指定App的上下文对象，设置 `getApplicationContext();` 即可。【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无

参数	说明
isAliyunos	用于判断App所在平台是否为YunOS。 【数据类型】布尔型 【取值范围】false或true 【是否必选】否 【是否可为空】是 【默认值】false
rsaPublicKey	用于指定性能分析公钥。 【数据类型】字符串 【获取方式】参见：准备 【是否必选】是 【是否可为空】否 【默认值】无

2. 在 `AndroidManifest.xml` 中添加代码段注册Application。

```
<application
        android:name=".MyApplication"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
</application>
```

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
-keep class com.taobao.monitor.APMLauncher{*;}
-keep class com.taobao.monitor.impl.logger.Logger{*;}
-keep class com.taobao.monitor.impl.logger.IDataLogger{*;}
-keep class com.taobao.monitor.impl.data.AbsWebView{*;}
-keep class com.taobao.monitor.impl.data.GlobalStats{*;}
-keep class com.taobao.monitor.impl.common.Global{*;}
-keep class com.taobao.monitor.impl.data.WebViewProxy{*;}
-keep class com.taobao.monitor.impl.logger.Logger{*;}
-keep class com.taobao.monitor.impl.processor.pageload.IProcedureManager{*;}
-keep class com.taobao.monitor.impl.processor.pageload.ProcedureManagerSetter{*;}
-keep class com.taobao.monitor.impl.util.TimeUtils{*;}
-keep class com.taobao.monitor.impl.util.TopicUtils{*;}
-keep class com.taobao.monitor.impl.common.DynamicConstants{*;}
-keep class com.taobao.application.common.data.DeviceHelper{*;}
-keep class com.taobao.application.common.impl.AppPreferencesImpl{*;}
-keep class com.taobao.monitor.impl.processor.launcher.PageList{*;}
-keep class com.taobao.monitor.impl.processor.fragmentload.FragmentInterceptorProxy{*;}
-keep class com.taobao.monitor.impl.processor.fragmentload.IFragmentInterceptor{*;}
-keep class com.taobao.monitor.impl.logger.DataLoggerUtils{*;}
-keep interface com.taobao.monitor.impl.data.IWebView{*;}
-keep interface com.taobao.monitor.impl.processor.IProcessor{*;}
-keep interface com.taobao.monitor.impl.processor.IProcessorFactory{*;}
-keep interface com.taobao.monitor.impl.logger.IDataLogger{*;}
-keep interface com.taobao.monitor.impl.trace.IDispatcher{*;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable,*An
notation*,EnclosingMethod
```

编译

如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

样例代码

性能分析服务Android SDK接入工程样例参见：Demo工程

功能验证

Android SDK接入操作完成后，可操作App，查看性能分析服务控制台显示数据，进行功能验证。

1. 手机端：启动App。（2分钟后）控制台：查看**概览**页签的**启动速度**是否显示数据。
2. 手机端：在App中跳转几个页面。（2分钟后）控制台：查看**概览**页签的**加载时间**是否显示数据。

> **说明**
> 数据从App采集到控制台显示，存在大约2分钟延迟。

如数据显示正常，则Android SDK接入成功；
否则，可能的原因是：SDK接入失败、SDK未获取数据、数据发送失败、后端问题，请联系技术支持解决。

SDK版本说明

版本号	发布日期	变更说明
1.0.8.2-open	2020-12-18	代码优化。

版本号	发布日期	变更说明
1.0.8.1-open	2020-11-24	优化编译构建问题。
1.0.8.0-open	2020-11-12	新增网络异常分析功能。
其他历史版本。		

3.3.3. iOS SDK接入（Pod集成）

本文介绍如何使用Pod集成方式接入性能分析服务的iOS SDK。

> **说明**
> - iOS SDK接入可采用Pod集成和手动集成2种方式。推荐使用Pod集成方式接入性能分析服务，可大幅简化接入操作。
> - 如需使用手动集成方式接入性能分析服务的iOS SDK，操作方法参见：iOS SDK接入（手动集成）

前提条件

已下载iOS配置文件，具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持iOS 8.0及以上的App。

接入概述

通过iOS SDK接入性能分析服务的操作步骤如下：

1. 添加依赖：采用Pod方式集成SDK。
2. 接入服务：初始化SDK，接入服务。
3. 编译：修改编译设置。

添加依赖

1. 指定官方仓库和阿里云仓库。

```
source "https://github.com/CocoaPods/Specs.git"
source "https://github.com/aliyun/aliyun-specs.git"
```

2. 添加依赖。

```
pod 'AlicloudAPM', '1.1.1'
```

> **说明**
> 执行 `pod search AlicloudAPM` 命令，查询AlicloudAPM最新版本。

3. 执行 `pod update` 命令，保存设置。

接入服务

1. 将iOS配置文件 `AliyunEmasServices-Info.plist` 拷贝至项目根目录。

> **说明**
> iOS配置文件的获取方式参见：前提条件

2. 在 `AppDelegate.m` 文件的 `application:didFinishLaunchingWithOptions` 方法中初始化SDK。

引入头文件：

```
#import <AlicloudAPM/AlicloudAPMProvider.h>
#import <AlicloudHAUtil/AlicloudHAProvider.h>
```

添加代码段：

```
NSString *appVersion = @"xxx";
NSString *channel = @"xxx";
NSString *nick = @"xxx";
[[AlicloudAPMProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:nic
k];
[AlicloudHAProvider start];
```

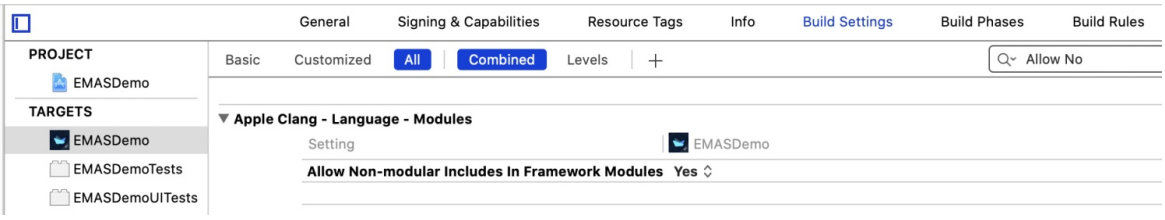
配置说明：

配置项	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *appVersion = @"2.3";`

配置项	说明
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *channel = @"appstore";`
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义 【示例】`NSString *nick = @"john";`

编译

1. 在项目的 `Build Setting` 中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。



2. 执行编译。

> **说明**
>
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

样例代码

性能分析服务iOS SDK接入工程样例参见：Demo工程

功能验证

iOS SDK接入操作完成后，可操作App，查看性能分析服务控制台显示数据，进行功能验证。

1. 手机端：启动App。（2分钟后）控制台：查看**概览**页签的**启动速度**是否显示数据。
2. 手机端：在App中跳转几个页面。（2分钟后）控制台：查看**概览**页签的**加载时间**是否显示数据。

> **说明**
>
> 数据从App采集到控制台显示，存在大约2分钟延迟。

如数据显示正常，则iOS SDK接入成功；
否则，可能的原因是：SDK接入失败、SDK未获取数据、数据发送失败、后端问题，请联系技术支持解决。

3.3.4. iOS SDK接入（手动集成）

本文介绍如何使用手动集成方式接入性能分析服务的iOS SDK。

> **说明**
>
> - iOS SDK接入可采用Pod集成和手动集成2种方式。推荐使用Pod集成方式接入性能分析服务，可大幅简化接入操作。
> - 如需使用Pod集成方式接入性能分析服务的iOS SDK，操作方法参见：iOS SDK接入（Pod集成）

前提条件

已下载iOS配置文件，具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持iOS 8.0及以上的App。

接入概述

通过iOS SDK接入性能分析服务的操作步骤如下：

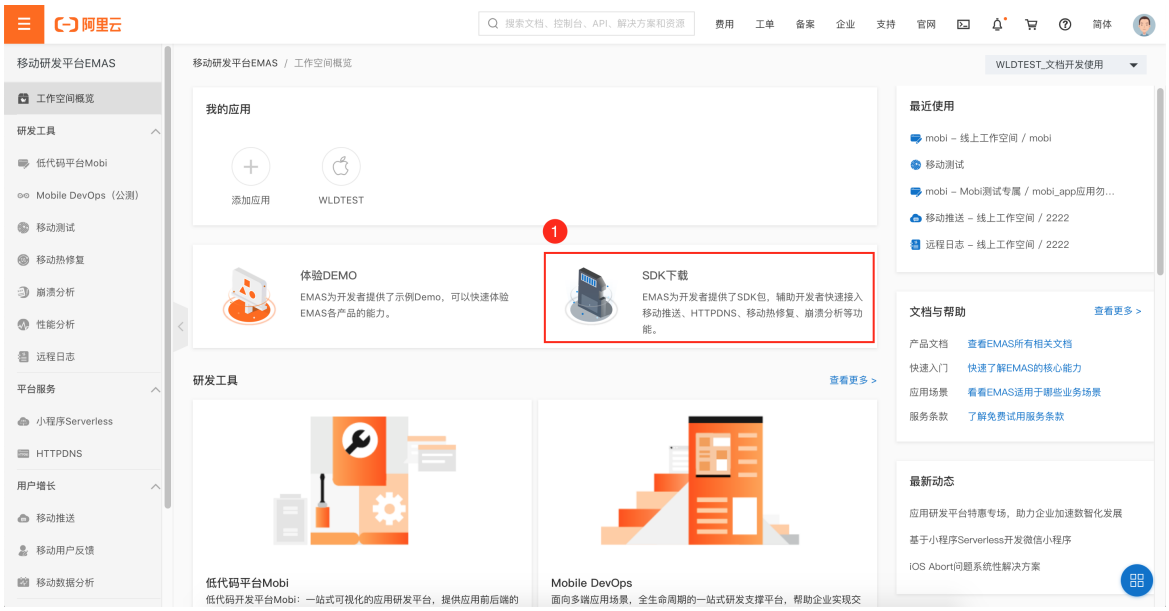
1. 准备：下载SDK包；下载开源库文件。
2. 添加依赖：采用手动方式集成SDK。
3. 接入服务：接入服务。
4. 编译：修改编译设置。

准备

下载SDK包和开源库文件。

下载SDK包

1. 登录移动研发平台的管理控制台，默认打开**我的工作空间**页面。
2. 在**我的工作空间**页面，单击工作空间标签的**进入**按钮，打开指定工作空间的概览页面。
3. 在工作空间概览页面，单击**SDK下载**区域，打开**SDK下载**右侧栏。



4. 在**SDK下载**右侧栏，选择**性能分析**复选框，单击**下载iOS版本**按钮，下载性能分析服务的iOS SDK包。

> **说明**
> 在**SDK列表**中，单击**性能分析**行**iOS版**列的版本号链接，查看性能分析iOS SDK更新记录。

5. 检查SDK包，确保内容完整无缺失。SDK包文件（24个）列表如下：

- AliAPMInterface.framework
- AlicloudAPM.framework

- AlicloudHAUtil.framework
- AliCloudNetworkMonitor.framework
- AlicloudUtils.framework
- AliHACore.framework
- AliHADataHub4iOS.framework
- AliHADataHubAssembler.framework
- AliHADeviceEvaluation.framework
- AliHALogEngine.framework
- AliHAMemoryMonitor.framework
- AliHAMethodTrace.framework
- AliHAPerformanceMonitor.framework
- AliHAProtocol.framework
- AliHASecurity.framework
- AliRemoteDebugInterface.framework
- BizErrorReporter4iOS.framework
- EMASRest.framework
- JDYThreadTrace.framework
- TBJSONModel.framework
- TBRest.framework
- UTDID.framework
- UTMini.framework
- ZipArchive.framework

下载开源库文件

性能分析服务须引用的开源库文件包括：

- FBAllocationTracker：下载
- FBMemoryProfiler：下载
- FBRetainCycleDetector：下载

添加依赖

1. 在Xcode中，将SDK目录中的framework文件拖入Target目录，在弹出框选中**Copy items if needed**选项。
2. 相同方式引入开源库文件：

- FBAllocationTracker
- FBMemoryProfiler
- FBRetainCycleDetector
- rcd_fishhook

> **说明**
>
> `rcd_fishhook` 在 `FBRetainCycleDetector` 工程目录下。

3. 选择**Build Phases** > **Link Binary With Libraries**，添加Xcode自带的公共包文件：

- libc++.tbd
- SystemConfiguration.framework

4. 选择**Build Phases** > **Compile Sources**，为下列文件添加Compiler Flags：`-fno-objc-arc`

- FBAssociationManager.mm
- FBBlockStrongRelationDetector.m
- FBBlockStrongLayout.m
- FBClassStrongLayoutHelpers.m
- NSObject+FBAllocationTracker.mm
- FBAllocationTrackerNSZombieSupport.mm

接入服务

1. 将iOS配置文件`AliyunEmasServices-Info.plist`拷贝至项目根目录。

> **说明**
> iOS配置文件的获取方式参见：前提条件

2. 在`AppDelegate.m`文件的`application:didFinishLaunchingWithOptions`方法中初始化SDK。

引入头文件：

```
#import <AlicloudAPM/AlicloudAPMProvider.h>
#import <AlicloudHAUtil/AlicloudHAProvider.h>
```

添加代码段：

```
NSString *appVersion = @"xxx";
NSString *channel = @"xxx";
NSString *nick = @"xxx";
[[AlicloudAPMProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:nick];
[AlicloudHAProvider start];
```

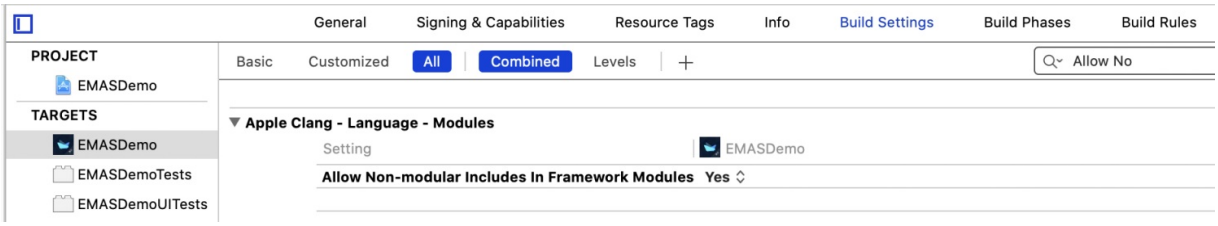
配置说明如下：

配置项	说明

配置项	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *appVersion = @"2.3";`
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *channel = @"appstore";`
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义 【示例】`NSString *nick = @"john";`

编译

1. 在项目的 `Build Setting` 中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。



2. 执行编译。

> **说明**
>
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UT DID冲突，造成编译失败。解决办法参见：SDK UT DID冲突解决方案

样例代码

性能分析服务iOS SDK接入工程样例参见：Demo工程

功能验证

iOS SDK接入操作完成后，可操作App，查看性能分析服务控制台显示数据，进行功能验证。

1. 手机端：启动App。（2分钟后）控制台：查看**概览**页签的**启动速度**是否显示数据。
2. 手机端：在App中跳转几个页面。（2分钟后）控制台：查看**概览**页签的**加载时间**是否显示数据。

> **说明**
>
> 数据从App采集到控制台显示，存在大约2分钟延迟。

如数据显示正常，则iOS SDK接入成功。
否则，可能的原因是：SDK接入失败、SDK未获取数据、数据发送失败、后端问题，请联系技术支持解决。

3.4. 远程日志

3.4.1. Android SDK接入（Maven集成）

本文介绍如何通过Maven集成方式添加依赖接入远程日志服务。

> **说明**
>
> - 接入远程日志服务的Android SDK可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用本地集成方式添加依赖，操作方法参见：Android SDK接入（本地集成）

前提条件

已下载Android配置文件，并在文件中获取AppKey、AppSecret和PackageName。具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持Android 4.2及以上版本。
- 仅支持armeabi、armeabi-v7a和x86架构。
- 日志在手机端最多存储7天。

接入概述

1. 准备：获取公钥。
2. 添加依赖：采用Maven集成方式。
3. 接入服务：添加自定义Application，以及初始化代码；配置ABI；设置日志拉入级别。
4. 打印日志：引入头文件；在代码中打印日志信息。
5. 混淆配置：如App对代码进行乱序混淆，则修改混淆配置文件。
6. 编译。

准备

打开配置文件，查询 `appmonitor.tlog.rsaSecret` 字段内容，即为远程日志公钥。

添加依赖

1. 在项目级 `build.gradle` 文件中的 `repositories{}` 代码段内添加阿里云Maven仓库地址。

```
repositories {
    maven { url "http://maven.aliyun.com/nexus/content/repositories/releases" }
}
```

2. 在应用级 `build.gradle` 文件中的 `dependencies{}` 代码段内添加依赖：

```
implementation('com.aliyun.ams:alicloud-android-ha-adapter:1.1.3.8-open@aar') {
    transitive=true
}
implementation('com.aliyun.ams:alicloud-android-tlog:1.1.3.1-open@aar') {
    transitive=true
}
```

3. 在应用级 `build.gradle` 文件的 `defaultConfig{}` 代码段内根据实际需要配置ABI。

```
ndk {
        abiFilters 'armeabi'     //配置项。可选取值：armeabi、armeabi-v7a和x86。
}
```

> **注意**
> 远程日志服务目前仅支持armeabi、armeabi-v7a和x86架构。

接入服务

1. 定义 `Application` 类，并编写 `onCreate` 方法启动服务：

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        initHa();
    }
    private void initHa() {
         AliHaConfig config = new AliHaConfig();
         config.appKey = "xxxxxxxx"; //配置项：appkey。
         config.appVersion = "x.xx"; //配置项：应用的版本号。
         config.appSecret = "xxxxxxxxxxxx"; //配置项：appsecret。
         config.channel = "mqc_test"; //配置项：应用的渠道号标记，自定义。
         config.userNick = null; //配置项：用户的昵称。
         config.application = this; //配置项：应用指针。
         config.context = getApplicationContext(); //配置项：应用上下文。
         config.isAliyunos = false; //配置项：是否为yunos。
         config.rsaPublicKey = "xxxxxxx"; //配置项：tlog公钥。
         AliHaAdapter.getInstance().addPlugin(Plugin.tlog);
         AliHaAdapter.getInstance().openDebug(true);
         AliHaAdapter.getInstance().start(config);
         TLogService.updateLogLevel(TLogLevel.XXXXXX); //配置项：控制台可拉取的日志级别。
    }
}
```

配置说明如下：

参数	说明
config.appKey	用于指定App的AppKey。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无

参数	说明
config.appVersion	用于设置App的版本号。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。
config.appSecret	用于指定App的AppSecret。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
config.channel	用于设置渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。
config.userNick	用于设置用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义

参数	说明
config.application	用于指定本应用。 **注意** 不能指向其他应用。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
config.context	用于指定App的上下文对象，设置 `getApplicationContext();` 即可。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
config.isAliyunos	用于判断App所在平台是否为YunOS。 【数据类型】布尔型 【取值范围】false或true 【是否必选】否 【是否可为空】是 【默认值】false
config.rsaPublicKey	用于指定远程日志公钥。 【数据类型】字符串 【如何获取】参见：准备 【是否必选】是 【是否可为空】否 【默认值】无
TLogLevel.XXXXXX	用于全局设置控制台可拉取的日志的级别。 【数据类型】枚举型 【取值范围】 ○ VERBOSE：可拉取所有级别的日志。 ○ DEBUG：可拉取DEBUG、INFO、WARN和ERROR级别的日志。 ○ INFO：可拉取INFO、WARN和ERROR级别的日志。 ○ WARN：可拉取WARN和ERROR级别的日志。 ○ ERROR：可拉取ERROR级别的日志。 【是否必选】是 【默认取值】ERROR 【配置说明】 ○ `TLogService.updateLogLevel()` 函数可选调用，如未调用，则全局默认可拉取的日志级别为ERROR。 ○ 日志级别说明参见：术语解释

2. 在 `AndroidManifest.xml` 中添加代码段注册 `Application` 。

```
<application
    android:name=".MyApplication"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme" >
    ...
</application>
```

打印日志

1. 如业务流程触发日志输出，需引入头文件：

```
import com.alibaba.ha.adapter.service.tlog.TLogService;
```

2. 在适当位置添加代码，打印日志信息。示例代码：

```
TLogService.logv("MODEL","TAG","MESSAGE");
TLogService.logd("MODEL","TAG","MESSAGE");
TLogService.logi("MODEL","TAG","MESSAGE");
TLogService.logw("MODEL","TAG","MESSAGE");
TLogService.loge("MODEL","TAG","MESSAGE");
```

函数： `TLogService.<LogLevel>(<MODEL>,<TAG,<MESSAGE>);`

说明： 用于输出指定级别的日志信息。

参数	说明
LogLevel	指定拉取的日志级别。日志级别说明参见：术语解释 【数据类型】枚举型 【取值范围】 • logv：输出VERBOSE（调试详情）级别的日志。 • logd：输出DEBUG（调试信息）级别的日志。 • logi：输出INFO（一般信息）级别的日志。 • logw：输出WARN（警告信息）级别的日志。 • loge：输出ERROR（错误信息）级别的日志。 【配置说明】输出的日志是否可以被控制台拉取，取决于 `TLogService.updateLogLevel()` 函数的参数设置。例如，当 `TLogService.updateLogLevel(TLogLevel.WARN)` 时，则控制台拉取不到通过logv、logd和logi输出的日志。
MODEL	用于设置输出日志内容的功能模块，便于后续根据来源筛选日志。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是 【是否大小写敏感】否 【示例】“推送功能模块Push”

参数	说明
TAG	用于设置日志的关键字，便于后续根据标签筛选日志。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是 【是否大小写敏感】否 【示例】“推送功能模块收到了推送Push.receive，推送功能模块点击了推送Push.click”
MESSAGE	用于输出日志信息。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
#keep class
-keep interface com.taobao.tao.log.ITLogController{*;}
-keep class com.taobao.tao.log.upload.*{*;}
-keep class com.taobao.tao.log.message.*{*;}
-keep class com.taobao.tao.log.LogLevel{*;}
-keep class com.taobao.tao.log.TLog{*;}
-keep class com.taobao.tao.log.TLogConstant{*;}
-keep class com.taobao.tao.log.TLogController{*;}
-keep class com.taobao.tao.log.TLogInitializer{public *;}
-keep class com.taobao.tao.log.TLogUtils{public *;}
-keep class com.taobao.tao.log.TLogNative{*;}
-keep class com.taobao.tao.log.TLogNative$*{*;}
-keep class com.taobao.tao.log.CommandDataCenter{*;}
-keep class com.taobao.tao.log.task.PullTask{*;}
-keep class com.taobao.tao.log.task.UploadFileTask{*;}
-keep class com.taobao.tao.log.upload.LogFileUploadManager{public *;}
-keep class com.taobao.tao.log.monitor.**{*;}
#兼容godeye
-keep class com.taobao.tao.log.godeye.core.module.*{*;}
-keep class com.taobao.tao.log.godeye.GodeyeInitializer{*;}
-keep class com.taobao.tao.log.godeye.GodeyeConfig{*;}
-keep class com.taobao.tao.log.godeye.core.control.Godeye{*;}
-keep interface com.taobao.tao.log.godeye.core.GodEyeAppListener{*;}
-keep interface com.taobao.tao.log.godeye.core.GodEyeReponse{*;}
-keep interface com.taobao.tao.log.godeye.api.file.FileUploadListener{*;}
-keep public class * extends com.taobao.android.tlog.protocol.model.request.base.FileInfo{*
;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable,*An
notation*,EnclosingMethod
```

编译

如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

样例代码

远程日志服务Android SDK接入工程样例参见：Demo工程

3.4.2. Android SDK接入（本地集成）

本文介绍如何通过本地集成方式添加依赖接入远程日志服务。

> 说明
>
> - 接入远程日志服务的Android SDK可采用Maven集成和本地集成2种方式添加依赖。推荐使用Maven集成方式添加依赖，可大幅简化接入操作。
> - 如需使用Maven集成方式添加依赖，操作方法参见：Android SDK接入（Maven集成）

前提条件

已下载Android配置文件，并在文件中获取AppKey、AppSecret和PackageName。具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持Android 4.2及以上版本。
- 仅支持armeabi、armeabi-v7a和x86架构。
- 日志在手机端最多存储7天。

接入概述

1. 准备：获取公钥；下载SDK包。
2. 添加依赖：采用本地集成方式。
3. 接入服务：添加自定义Application，以及初始化代码；配置ABI；设置日志拉入级别。
4. 打印日志：引入头文件；在代码中打印日志信息。
5. 混淆配置：如App对代码进行乱序混淆，则修改混淆配置文件。
6. 编译。

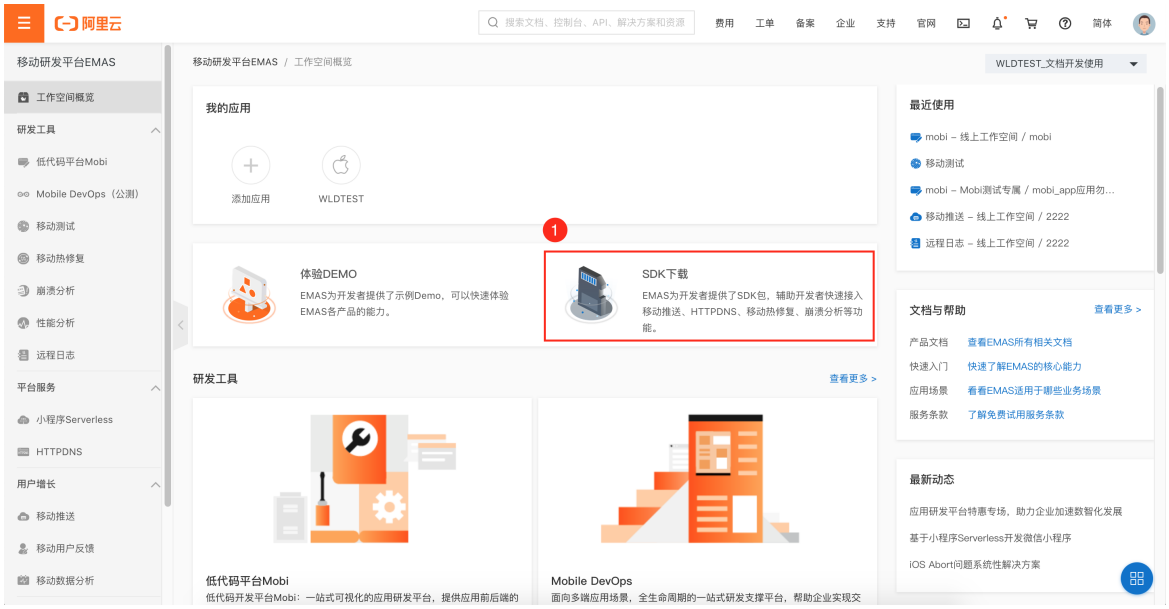
准备

获取性能分析公钥

打开配置文件，查询 `appmonitor.tlog.rsaSecret` 字段内容，即为远程日志公钥。

下载SDK包

1. 登录移动研发平台的管理控制台，默认打开**我的工作空间**页面。
2. 在**我的工作空间**页面，单击工作空间标签的**进入**按钮，打开指定工作空间的概览页面。
3. 在工作空间概览页面，单击**SDK下载**区域，打开**SDK下载**右侧栏。



4. 在**SDK下载**右侧栏，选中**远程日志**复选框，单击**下载Android版本**按钮，下载远程日志服务的SDK包。

> **说明**
> 在**远程日志**行的**Android版**列，单击版本号链接，可查看版本变更记录。

5. 检查SDK包，确保内容完整。

SDK包文件列表如下：

- alicloud-android-ha-adapter-1.1.3.8-open.aar
- alicloud-android-ha-core-1.1.0.6.1-open.aar
- alicloud-android-ha-protocol-1.1.1.0-open.aar
- alicloud-android-ha-tbrest-1.1.1.0-open.aar

- alicloud-android-ha-telescopebase-1.1.1-open.aar
- alicloud-android-ha-telescopesdk-1.1.3-open.aar
- alicloud-android-ha-tlog-message-rpc-1.1.3.1-open.aar
- alicloud-android-ha-tlog-native-1.1.0.7-open.aar
- alicloud-android-ha-tlog-protocol-1.1.0.8-open.aar
- alicloud-android-ha-tlog-uploader-oss-1.1.0.6.3-open.aar
- alicloud-android-tlog-1.1.3.1-open.aar
- alicloud-android-utdid-2.5.1-proguard.jar
- fastjson-1.1.54.android.jar
- okhttp-3.4.1.jar
- okio-1.9.0.jar
- oss-android-sdk-2.9.3.aar

> **说明**
> 如存在名称相同，版本号不同的包文件，添加最高版本的包文件即可。

添加依赖

1. 将SDK包内所有文件拷贝至项目的libs目录下。
2. 在项目级 `build.gradle` 项目文件中，添加本地SDK文件目录地址。

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

3. 在应用级 `build.gradle` 项目文件的 `dependencies{}` 代码段，添加SDK依赖。

```
//1、本地jar库引入。
compile fileTree(include:['*.jar'],dir:'libs')

//2、公共库。
compile(name:'alicloud-android-ha-adapter-1.1.3.8-open',ext:'aar')
compile(name:'alicloud-android-ha-core-1.1.0.6.1-open',ext:'aar')
compile(name:'alicloud-android-ha-protocol-1.1.1.0-open',ext:'aar')
compile(name:'alicloud-android-ha-tbrest-1.1.1.0-open',ext:'aar')
compile(name:'alicloud-android-utdid-2.5.1-proguard',ext:'jar')
compile(name:'fastjson-1.1.54.android',ext:'jar')


//3、移动日志，不接入可注释掉。
compile(name:'alicloud-android-tlog-1.1.3.1-open',ext:'aar')
compile(name:'alicloud-android-ha-tlog-message-rpc-1.1.3.1-open',ext:'aar')
compile(name:'alicloud-android-ha-telescopebase-1.1.1-open',ext:'aar')
compile(name:'alicloud-android-ha-tlog-uploader-oss-1.1.0.6.3-open',ext:'aar')
compile(name:'alicloud-android-ha-tlog-protocol-1.1.0.8-open',ext:'aar')
compile(name:'alicloud-android-ha-telescopesdk-1.1.3-open',ext:'aar')
compile(name:'alicloud-android-ha-tlog-native-1.1.0.7-open',ext:'aar')
compile(name:'okhttp-3.4.1',ext:'jar')
compile(name:'okio-1.9.0',ext:'jar')
compile(name:'oss-android-sdk-2.9.3',ext:'aar')
```

接入服务

1. 定义 `Application` 类，并编写 `onCreate` 方法启动服务：

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        initHa();
    }
    private void initHa() {
         AliHaConfig config = new AliHaConfig();
         config.appKey = "xxxxxxxx"; //配置项：appkey。
         config.appVersion = "x.xx"; //配置项：应用的版本号。
         config.appSecret = "xxxxxxxxxxxx"; //配置项：appsecret。
         config.channel = "mqc_test"; //配置项：应用的渠道号标记，自定义。
         config.userNick = null; //配置项：用户的昵称。
         config.application = this; //配置项：应用指针。
         config.context = getApplicationContext(); //配置项：应用上下文。
         config.isAliyunos = false; //配置项：是否为yunos。
         config.rsaPublicKey = "xxxxxxx"; //配置项：tlog公钥。
         AliHaAdapter.getInstance().addPlugin(Plugin.tlog);
         AliHaAdapter.getInstance().openDebug(true);
         AliHaAdapter.getInstance().start(config);
         TLogService.updateLogLevel(TLogLevel.XXXXXX); //配置项：控制台可拉取的日志级别。
    }
}
```

配置说明如下：

参数	说明
config.appKey	用于指定App的AppKey。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
config.appVersion	用于设置App的版本号。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。
config.appSecret	用于指定App的AppSecret。 【数据类型】字符串 【如何获取】参见：前提条件 【是否必选】是 【是否可为空】否 【默认值】无
config.channel	用于设置渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。

参数	说明
config.userNick	用于设置用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】否 【是否可为空】是 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义
config.application	用于指定本应用。 **注意** 不能指向其他应用。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
config.context	用于指定App的上下文对象，设置 `getApplicationContext();` 即可。 【数据类型】对象 【是否必选】是 【是否可为空】否 【默认值】无
config.isAliyunos	用于判断App所在平台是否为YunOS。 【数据类型】布尔型 【取值范围】false或true 【是否必选】否 【是否可为空】是 【默认值】false
config.rsaPublicKey	用于指定远程日志公钥。 【数据类型】字符串 【如何获取】参见：准备 【是否必选】是 【是否可为空】否 【默认值】无

参数	说明
TLogLevel.XXXXXX	用于全局设置控制台可拉取的日志的级别。 【数据类型】枚举型 【取值范围】 ○ VERBOSE：可拉取所有级别的日志。 ○ DEBUG：可拉取DEBUG、INFO、WARN和ERROR级别的日志。 ○ INFO：可拉取INFO、WARN和ERROR级别的日志。 ○ WARN：可拉取WARN和ERROR级别的日志。 ○ ERROR：可拉取ERROR级别的日志。 【是否必选】是 【默认取值】ERROR 【配置说明】 ○ `TLogService.updateLogLevel()` 函数可选调用，如未调用，则全局默认可拉取的日志级别为ERROR。 ○ 日志级别说明参见：术语解释

2. 在 `AndroidManifest.xml` 中添加代码段注册 `Application` 。

```
<application
    android:name=".MyApplication"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme" >
    ...
</application>
```

打印日志

1. 如业务流程触发日志输出，需引入头文件：

```
import com.alibaba.ha.adapter.service.tlog.TLogService;
```

2. 在适当位置添加代码，输出日志信息。示例代码：

```
TLogService.logv("MODEL","TAG","MESSAGE");
TLogService.logd("MODEL","TAG","MESSAGE");
TLogService.logi("MODEL","TAG","MESSAGE");
TLogService.logw("MODEL","TAG","MESSAGE");
TLogService.loge("MODEL","TAG","MESSAGE");
```

函数：`TLogService.<LogLevel>(<MODEL>,<TAG,<MESSAGE>);`

说明：用于输出指定级别的日志信息。

参数	说明
LogLevel	指定拉取的日志级别。日志级别说明参见：术语解释 【数据类型】枚举型 【取值范围】 • logv：输出VERBOSE（调试详情）级别的日志。 • logd：输出DEBUG（调试信息）级别的日志。 • logi：输出INFO（一般信息）级别的日志。 • logw：输出WARN（警告信息）级别的日志。 • loge：输出ERROR（错误信息）级别的日志。 【配置说明】输出的日志是否可以被控制台拉取，取决于`TLogService.updateLogLevel()`函数的参数设置。例如，当`TLogService.updateLogLevel(TLogLevel.WARN)`时，则控制台拉取不到通过logv、logd和logi输出的日志。
MODEL	用于设置输出日志内容的功能模块，便于后续根据来源筛选日志。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是 【是否大小写敏感】否 【示例】“推送功能模块Push”
TAG	用于设置日志的关键字，便于后续根据标签筛选日志。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是 【是否大小写敏感】否 【示例】“推送功能模块收到了推送Push.receive，推送功能模块点击了推送Push.click”
MESSAGE	用于输出日志信息。 【数据类型】字符串 【字符类型】英文大小写、中文、数字、特殊字符 【是否必选】是

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
#keep class
-keep interface com.taobao.tao.log.ITLogController{*;}
-keep class com.taobao.tao.log.upload.*{*;}
-keep class com.taobao.tao.log.message.*{*;}
-keep class com.taobao.tao.log.LogLevel{*;}
-keep class com.taobao.tao.log.TLog{*;}
-keep class com.taobao.tao.log.TLogConstant{*;}
-keep class com.taobao.tao.log.TLogController{*;}
-keep class com.taobao.tao.log.TLogInitializer{public *;}
-keep class com.taobao.tao.log.TLogUtils{public *;}
-keep class com.taobao.tao.log.TLogNative{*;}
-keep class com.taobao.tao.log.TLogNative$*{*;}
-keep class com.taobao.tao.log.CommandDataCenter{*;}
-keep class com.taobao.tao.log.task.PullTask{*;}
-keep class com.taobao.tao.log.task.UploadFileTask{*;}
-keep class com.taobao.tao.log.upload.LogFileUploadManager{public *;}
-keep class com.taobao.tao.log.monitor.**{*;}
#兼容godeye
-keep class com.taobao.tao.log.godeye.core.module.*{*;}
-keep class com.taobao.tao.log.godeye.GodeyeInitializer{*;}
-keep class com.taobao.tao.log.godeye.GodeyeConfig{*;}
-keep class com.taobao.tao.log.godeye.core.control.Godeye{*;}
-keep interface com.taobao.tao.log.godeye.core.GodEyeAppListener{*;}
-keep interface com.taobao.tao.log.godeye.core.GodEyeReponse{*;}
-keep interface com.taobao.tao.log.godeye.api.file.FileUploadListener{*;}
-keep public class * extends com.taobao.android.tlog.protocol.model.request.base.FileInfo{*
;}
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable,*An
notation*,EnclosingMethod
```

编译

如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

样例代码

远程日志服务Android SDK接入工程样例参见：Demo工程

3.4.3. iOS SDK接入（Pod集成）

本文介绍如何通过Pod集成方式添加依赖接入远程日志服务 。

> **说明**
>
> - 接入远程日志服务的iOS SDK可采用Pod集成和手动集成2种方式添加依赖。推荐使用Pod集成方式添加依赖，可大幅简化接入操作。
> - 如需使用手动集成方式添加依赖，操作方法参见：iOS SDK接入（手动集成）

前提条件

已下载iOS配置文件，具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持iOS 8.0及以上的App

- 仅支持iOS 8.0及以上的App。
- 推荐使用CocoaPods管理依赖的Xcode项目。
- 日志在手机端上最多存储7天 。

接入概述

1. 添加依赖：采用Pod集成方式添加依赖。
2. 接入服务：添加iOS配置文件；引入头文件；初始化SDK ；设置日志拉取级别。
3. 执行编译：添加编译设置。
4. 打印日志：在业务代码中打印日志信息。

添加依赖

1. 指定官方仓库和阿里云仓库。

```
source "https://github.com/CocoaPods/Specs.git"
source "https://github.com/aliyun/aliyun-specs.git"
```

2. 添加依赖。

```
pod 'AlicloudTLog', '~> 1.0.0.2'
```

> 说明
> 执行 `pod search AlicloudTLog` 命令，查询 `AlicloudTLog` 最新版本。

3. 执行 `pod install` 或者 `pod update` 命令，获取SDK到项目中。

接入服务

1. 将iOS配置文件 `AliyunEmasServices-Info.plist` 拷贝至项目根目录。
   iOS配置文件获取方式参见：前提条件
2. 在 `AppDelegate.m` 文件中引入头文件：

```
#import <AlicloudTLog/AlicloudTlogProvider.h>
#import <AlicloudHAUtil/AlicloudHAProvider.h>
#import <TRemoteDebugger/TRDManagerService.h>
```

3. 在 `AppDelegate.m` 文件的 `application:didFinishLaunchingWithOptions` 方法中，添加代码段，初始化SDK。

```
NSString *appVersion = @"x.x"; //配置项：App版本。
NSString *channel = @"xx"; //配置项：渠道标记。
NSString *nick = @"xx"; //配置项：用户昵称。
[[AlicloudTlogProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:ni
ck];
[AlicloudHAProvider start];
[TRDManagerService updateLogLevel:TLogLevelXXX]; //配置项：控制台可拉取的日志级别。
```

   参数说明：

参数	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *appVersion = @"1.0";`
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 不支持中文字符、特殊字符。 【示例】`NSString *channel = @"appstore";`
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义 【示例】`NSString *nick = @"wldtest";`

参数	说明
TLogLevelXXX	用于设置控制台可拉取的日志级别。日志级别说明参见：术语解释 【数据类型】枚举型 【取值范围】 ○ TLogLevelError：拉取Error级别的日志。 ○ TLogLevelWarn：拉取Warn和Error级别的日志。 ○ TLogLevelInfo：拉取Warn、Error和Info级别的日志。 ○ TLogLevelDebug：拉取Warn、Error、Info和Debug级别的日志。 【是否必选】否 【默认取值】TLogLevelInfo 【示例】`[TRDManagerService updateLogLevel:TLogLevelInfo];`

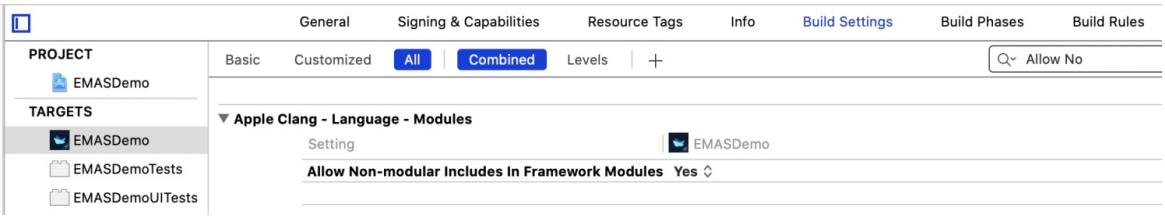
> 说明
> 推荐使用 `autoInitWithAppVersion` 接口接入服务。如需使用 `initWithAppKey` 接口接入服务，须手动配置 `appKey` / `secret` / `tlogRsaSecret` 参数。

打开iOS配置文件，查询参数取值：

参数	配置文件字段	说明
appKey	emas.appKey	App标识。
secret	emas.appSecret	App认证信息。
tlogRsaSecret	appmonitor.tlog.rsaSecret	远程日志公钥。

执行编译

1. 在项目的Build Setting中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。



2. 执行编译。

> 说明
>
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

打印日志

1. 如业务流程触发日志输出，需引入头文件：

```
#import <TRemoteDebugger/TLogBiz.h>
#import <TRemoteDebugger/TLogFactory.h>
#import <TRemoteDebugger/TRDManagerService.h>
```

2. 在适当位置添加代码，输出日志信息。示例代码：

```
TLogBiz *log = [TLogFactory createTLogForModuleName:@"YourModuleName"];
[log error:@"error message"];
[log warn:@"warn message"];
[log debug:@"debug message"];
[log info:@"info message"];
```

选项	说明
YourModuleName	指定保存日志信息的模块的名称。
error/warn/debug/info message	根据实际场景，区分级别输出日志信息，便于后续按照级别进行日志信息查询。日志级别说明参见：术语解释

样例代码

远程日志服务iOS SDK接入工程样例参见：Demo工程

3.4.4. iOS SDK接入（手动集成）

本文介绍如何通过手动集成方式添加依赖接入远程日志服务。

> 说明
>
> - 接入远程日志服务的iOS SDK可采用Pod集成和手动集成2种方式添加依赖。推荐使用Pod集成方式添加依赖，可大幅简化接入操作。
> - 如需使用Pod集成方式添加依赖，操作方法参见：iOS SDK接入（Pod集成）

前提条件

已下载iOS配置文件，具体操作，请参见快速入门：创建监控任务。

使用限制

- 仅支持iOS 8.0及以上的App。

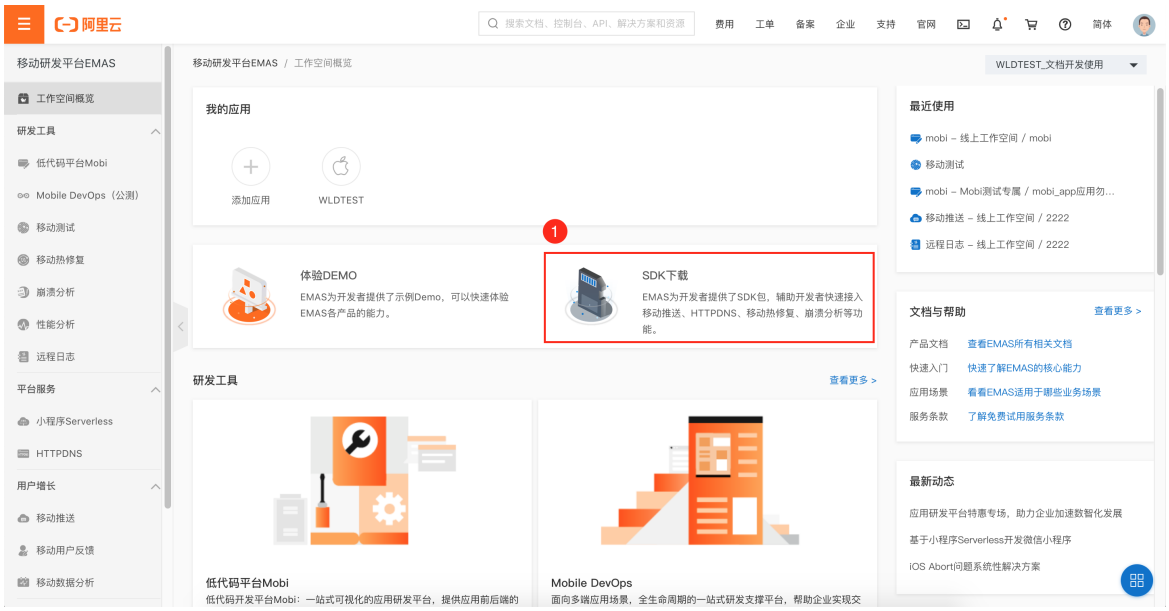
- 日志在手机端上最多存储7天。

接入概述

1. 准备：下载SDK包。
2. 添加依赖：采用手动集成方式。
3. 接入服务：添加iOS配置文件；引入头文件；初始化SDK；设置日志拉取级别。
4. 执行编译：添加编译设置。
5. 打印日志：在业务代码中输出日志信息。

准备

1. 登录移动研发平台的管理控制台，默认打开**我的工作空间**页面。
2. 在**我的工作空间**页面，单击工作空间标签的**进入**按钮，打开指定工作空间的概览页面。
3. 在工作空间概览页面，单击**SDK下载**区域，打开**SDK下载**右侧栏。



4. 在**SDK下载**右侧栏，选择**远程日志**前的复选框，单击**下载iOS版本**按钮，下载远程日志的SDK包。

> **说明**
> 在**远程日志**行的**iOS版**列，单击版本号链接，可查看版本变更记录。

5. 检查SDK包，确保内容完整无缺失。
   SDK包文件列表如下：

- AlicloudHAUtil.framework
- AlicloudTLog.framework
- AlicloudUtils.framework
- AliHACore.framework
- AliHALogEngine.framework
- AliHAMethodTrace.framework
- AliHAProtocol.framework
- AliHASecurity.framework
- AliyunOSSiOS.framework
- RemoteDebugChannel.framework
- TBJSONModel.framework
- TBRest.framework
- TRemoteDebugger.framework
- UTDID.framework
- UTMini.framework
- ZipArchive.framework

添加依赖

1. 在Xcode中，将SDK目录中的framework文件拖入Target目录，在弹出框选中 `Copy items if needed` 选项。
2. 选择**Build Phases** > **Link Binary With Libraries**，添加Xcode自带的公共包文件：

- libc++.tbd

- libresolv.tbd
- SystemConfiguration.framework

接入服务

1. 将iOS配置文件 `AliyunEmasServices-Info.plist` 拷贝至项目根目录。
   iOS配置文件获取方式参见：前提条件
2. 在 `AppDelegate.m` 文件中引入头文件：

```
#import <AlicloudTLog/AlicloudTlogProvider.h>
#import <AlicloudHAUtil/AlicloudHAProvider.h>
#import <TRemoteDebugger/TRDManagerService.h>
```

3. 在 `AppDelegate.m` 文件的 `application:didFinishLaunchingWithOptions` 方法中，添加代码段，初始化SDK。

```
NSString *appVersion = @"x.x"; //配置项：App版本
NSString *channel = @"xx"; //配置项：渠道标记
NSString *nick = @"xx"; //配置项：用户昵称
[[AlicloudTlogProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:nick];
[AlicloudHAProvider start];
[TRDManagerService updateLogLevel:TLogLevelXXX]; //配置项：控制台可拉取的日志级别
```

参数说明：

参数	说明
appVersion	用于指定App的版本，上报至服务端，进行版本区分。 【数据类型】字符串 【格式要求】自定义 【取值范围】任意长度。 **说明** 该参数值将在控制台显示为下拉列表选项，建议短小凝练。 【是否必选】是 【是否可为空】否 【默认值】无 【大小写敏感】是。例如，vx.x和Vx.x不是一个版本。 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】 `NSString *appVersion = @"1.0";`

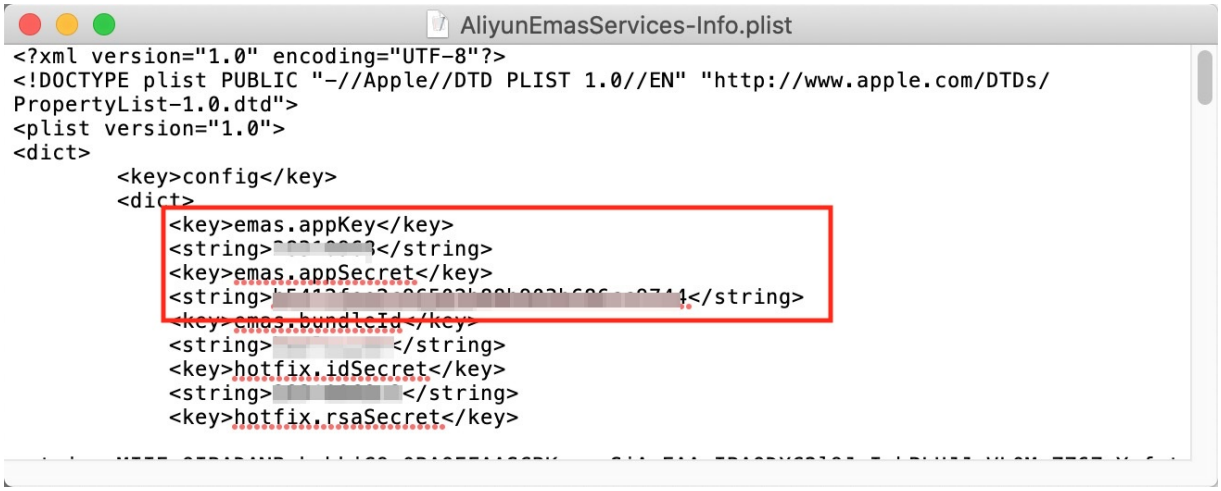
参数	说明
channel	用于指定渠道标识，上报至服务端，进行渠道区分。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【示例】`NSString *channel = @"appstore";`
nick	用于指定用户昵称，上报至服务端，进行用户区分。后续可能依据该参数，进行数据检索。 【数据类型】字符串 【取值范围】任意长度 【是否必选】是 【是否可为空】否 【默认值】无 【字符类型】英文大小写、数字。 **说明** 该参数不支持中文字符、特殊字符。 【命名规范】自定义 【示例】`NSString *nick = @"wldtest";`
TLogLevelXXX	用于设置控制台可拉取的日志级别。日志级别说明参见：术语解释 【数据类型】枚举型 【取值范围】 • TLogLevelError：拉取Error级别的日志。 • TLogLevelWarn：拉取Warn和Error级别的日志。 • TLogLevelInfo：拉取Warn、Error和Info级别的日志。 • TLogLevelDebug：拉取Warn、Error、Info和Debug级别的日志。 【是否必选】否 【默认取值】TLogLevelInfo 【示例】`[TRDManagerService updateLogLevel:TLogLevelInfo];`

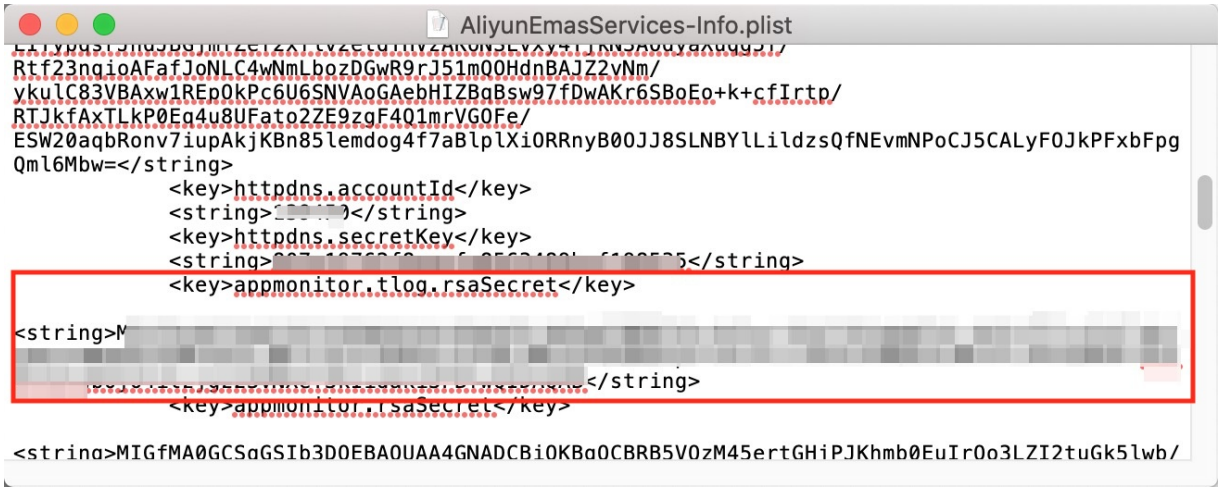
说明

推荐使用 `autoInitWithAppVersion` 接口接入服务。如需使用 `initWithAppKey` 接口接入服务，须手动配置 `appKey` 、 `secret` 和 `tlogRsaSecret` 参数。

打开iOS配置文件，查询参数取值：

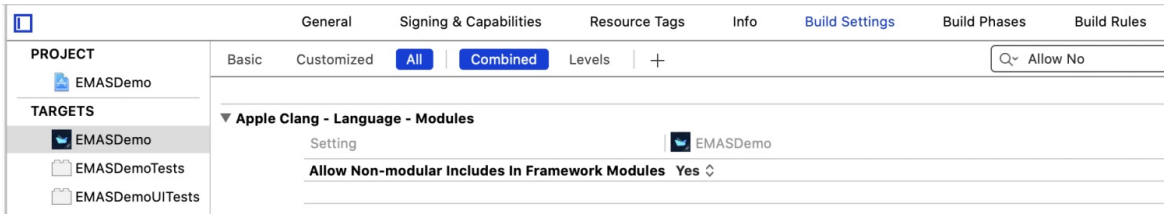
参数	配置文件字段	说明
appKey	emas.appKey	App标识。
secret	emas.appSecret	App认证信息。
tlogRsaSecret	appmonitor.tlog.rsaSecret	远程日志公钥。





执行编译

1. 在项目的Build Setting中，将 `Allow Non-modular Includes In Framework Modules` 设置为 `YES` 。



2. 执行编译。

> 说明
>
> - 编译过程中如出现 `duplicate symbol` 类型错误，确认本地依赖与CocoaPods管理的依赖是否重复；如是，则删除本地依赖。
> - 如同时使用其他阿里云产品，可能会因为依赖中存在UTDID冲突，造成编译失败。解决办法参见：SDK UTDID冲突解决方案

打印日志

1. 如业务流程触发日志输出，需引入头文件：

```
#import <TRemoteDebugger/TLogBiz.h>
#import <TRemoteDebugger/TLogFactory.h>
#import <TRemoteDebugger/TRDManagerService.h>
```

2. 在适当位置添加代码，输出日志信息。示例代码：

```
TLogBiz *log = [TLogFactory createTLogForModuleName:@"YourModuleName"];
[log error:@"error message"];
[log warn:@"warn message"];
[log debug:@"debug message"];
[log info:@"info message"];
```

选项	说明
YourModuleName	指定保存日志信息的模块的名称。
error/warn/debug/info message	根据实际场景，区分级别输出日志信息，便于后续按照级别进行日志信息查询。日志级别说明参见：术语解释

样例代码

远程日志服务iOS SDK接入工程样例参见：Demo工程

4.崩溃分析

4.1. 实时数据

本文介绍ARMS App监控崩溃分析的趋势概览和Top 10问题。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。

概览

在左侧导航栏选择**今日实时 > 趋势概览**。

趋势概览页面展示了崩溃、卡顿、异常3大类下各指标的详细数据。

Top 10问题

在左侧导航栏选择**今日实时 > Top 10问题**。

Top 10问题页面展示了各错误类型下发生次数最多的Top 10问题列表。单击列表内各错误类型标题进入聚合详情，聚合详情详细说明，请参见查看错误聚合详情。

查看错误聚合详情

基本信息

基本信息区域介绍了各种错误类型的聚合详情基本信息。

参数	说明
ID	此条错误的唯一标识。
发生次数	所选时间段内此错误发生的总次数。
影响用户数	所选时间段内此错误发生后影响的设备数。

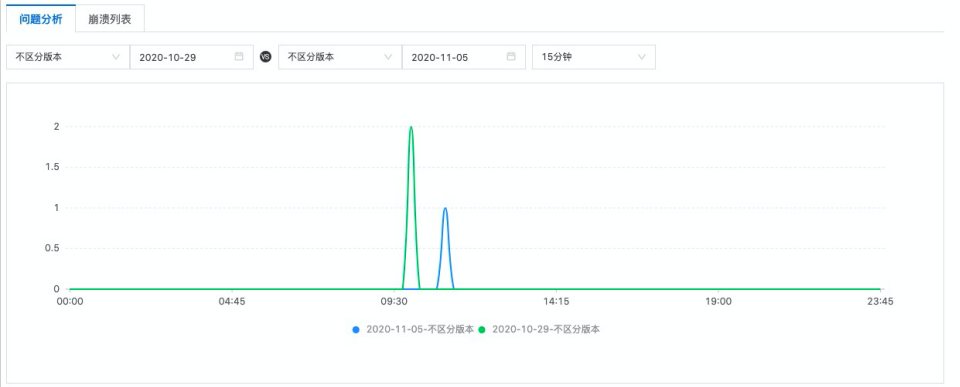
在**基本信息**区域您可以执行以下操作：

- 添加标签：单击**添加**，可为此条错误添加标签。
- 修改状态：单击错误详情右上角的下拉框，您可以更改此错误类型的状态为New、Open或Fixed。不同状态的含义如下：
  - New：新出现
  - Fixed：已被修复
  - Open：被修复后再次出现

问题分析

问题分析页签分为趋势对比、调用栈分析、特征分析3个功能模块：

- 趋势对比：您可以通过选择版本、时间，对两个独立天的数据进行对比。



- 调用栈分析：查看问题发生的堆栈。

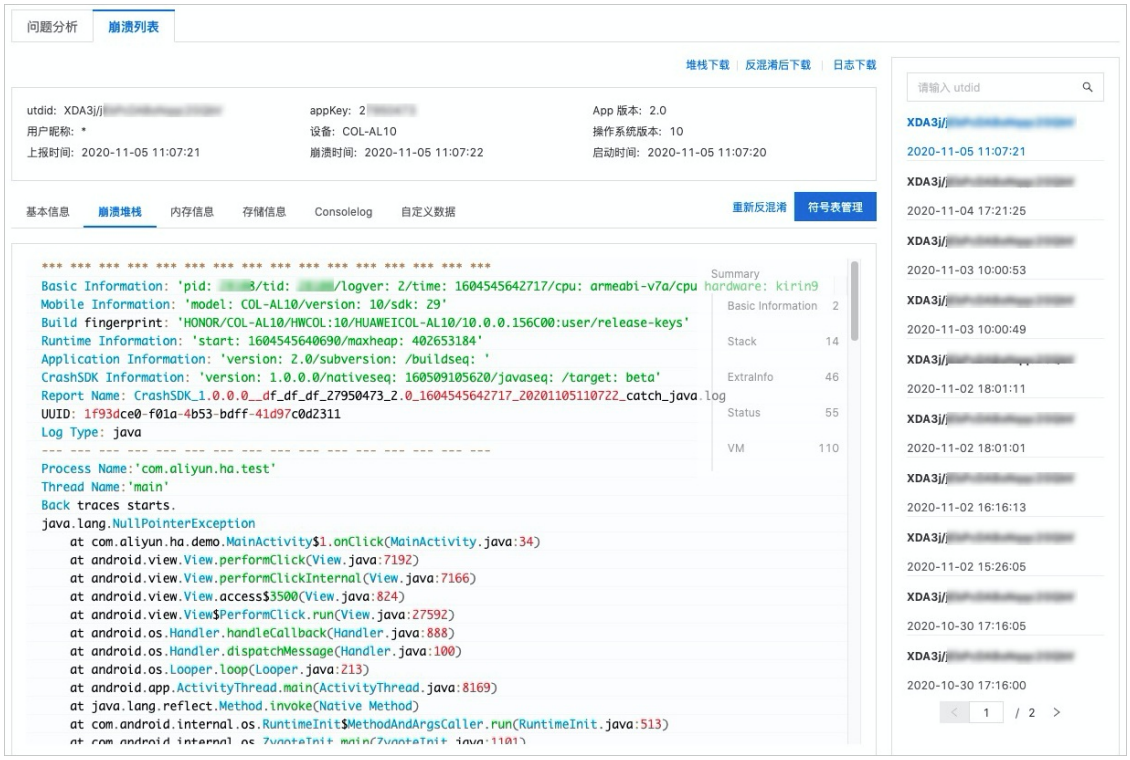
- 特征分析：展示此聚合详情中不同品牌、进程、页面、地市、机型、渠道、系统版本、View、应用版本维度下个值的占比情况。

> **说明** 单击单图右上角切换图标，可在饼图和列表的展示形式之间切换。

崩溃列表

崩溃列表中展示了聚合到当前错误中的所有错误实例，每条实例代表一个发生此错误的设备。

崩溃列表页签右侧展示所有的崩溃信息，单击某一条数据，左侧页面将会展示此条崩溃信息的基本信息、崩溃堆栈、内存信息、存储信息、ConsoleLog和自定义数据。



在**崩溃列表**页签，您可以执行以下操作：

- **堆栈下载**：下载原始堆栈信息。
- **反混淆后下载**：下载反混淆后的堆栈信息。
- **日志下载**：下载系统日志。
- **重新反混淆**：重新执行反混淆，执行后可单击反混淆下载，下载最新反混淆后的堆栈信息。
- **符号表管理**：配置符号表。

基本信息区域展示了此条崩溃信息的全部基本信息。

参数	说明
appKey	接入崩溃分析时使用的AppKey
应用版本	接入崩溃分析时传入的应用版本
构建号	暂无意义
渠道	接入崩溃分析时传入的渠道标识
记录ID	本次崩溃的ID
聚合ID	本次崩溃的同类崩溃聚合ID
聚合类型	崩溃类型
上报时间	上报时间，服务端时间
启动时间	App启用时间，客户端时间
崩溃时间	崩溃发生时间，客户端时间

参数	说明
品牌	崩溃发生设备的品牌
机型	崩溃发生设备的型号标识
操作系统/版本	崩溃发生时设备的操作系统及系统版本
运营商/网络	崩溃发生时设备接入的运营商和网络信息
国家/地区	崩溃发生时设备所处的地理位置，根据IP推算
用户IP	崩溃发生时设备的公网IP
分辨率	崩溃发生设备的屏幕分辨率
View	崩溃发生时App打开的视图
Page	崩溃发生时App打开的页面
是否是前台	崩溃是否发生在前台进程
是否Root	崩溃发生设备是否已被Root
是否是主线程	崩溃是否发生在主线程
异常类型	崩溃的异常类型
Native模块	崩溃相关的Native模块
所属模块	暂无意义
进程名	崩溃进程的名称
父进程名	崩溃进程的父进程名称

崩溃堆栈区域展示了崩溃发生时的堆栈及相关的线程、App所处状态等信息。

内存信息区域展示了崩溃繁盛时内存使用状态。

Android

参数	说明
MemTotal	系统可用总内存，不包括kernel占用的内存，这个值在系统运行期间一般是固定不变的
MemFree	系统尚未使用的内存
MemAvailable	kernel估算出的可用内存
Buffers	块设备（block device）所占用的缓存页

参数	说明
Cached	普通文件所占用的缓存页，包含已被进程解除关联
SwapCached	匿名页（anonymous pages）使用到的交换缓存区
Active	最近被访问过的内存页
Inactive	长时间未被访问过的内存页
Active(anon)	最近被访问过的匿名内存页
Inactive(anon)	长时间未被访问过的匿名内存页
Active(file)	最近被访问过的文件内存页
Inactive(file)	长时间未被访问过的文件内存页
Unevictable	不能pageout/swapout的内存页
Mlocked	被mlock()系统调用锁定的内存大小
SwapTotal	交换缓存区总大小
SwapFree	交换缓存区空闲大小
Dirty	等待被写回磁盘的缓存也页
Writeback	正准备回写硬盘的缓存页
AnonPages	匿名内存页
Mapped	普通文件所占用的缓存页
Shmem	共享内存，以及tmpfs和devtmpfs
Slab	通过slab分配的总内存
SReclaimable	slab中可回收的部分
SUnreclaim	slab中不可回收的部分
KernelStack	内核栈
PageTables	用于虚拟地址映射的内存页
NFS_Unstable	发给NFS server但尚未写入硬盘的缓存页
Bounce	内存跳转buffer使用的内存
WritebackTmp	FUSE使用的临时缓冲
CommitLimit	基于overcommit ratio计算得到的系统可分配内存

参数	说明
Committed_AS	完成当前负载预计需要的内存
VmallocTotal	vmalloc可分配的总虚拟内存
VmallocUsed	通过vmalloc分配的虚拟内存
VmallocChunk	可用的最大连续虚拟内存空间
AnonHugePages	匿名HugePages大小
HugePages_Total	系统HugePages的总大小
HugePages_Free	可用的HugePages大小
HugePages_Rsvd	申请后还未使用的HugePages大小
HugePages_Surp	剩余的HugePages大小
Hugepagesize	HugePages大小
DirectMap4k	映射为4 KB的内存数量
DirectMap2M	映射为2 MB的内存数量

说明 Android应用发生崩溃时的内存信息取自/proc/meminfo。

iOS

参数	说明
Memory Usage	内存使用量

存储信息区域展示了Android应用崩溃发生时存储使用状态。

参数	说明
hasSDCard	是否有SDCard
RootDirectory	*/system*系统目录
DataDirectory	*/data*数据目录
ExternalStorageDirectory	*/storage/emulated/0*外部储存目录
DownloadCacheDirectory	*/data/cache*下载缓存目录
TotalSize(Byte)	总大小
FreeSize(Byte)	空闲大小

参数	说明
AvailableSize(Byte)	可用大小

Consolelog区域展示了崩溃发生前后一段时间的系统日志。

自定义数据区域展示了崩溃发生时通过SDK回调来上报的自定义数据。

4.2. 崩溃/卡顿/异常数据

您可以在**崩溃**、**卡顿**或**异常**数据页面查看不同问题类型的明细数据，并查看所选异常类型下的聚合数据。

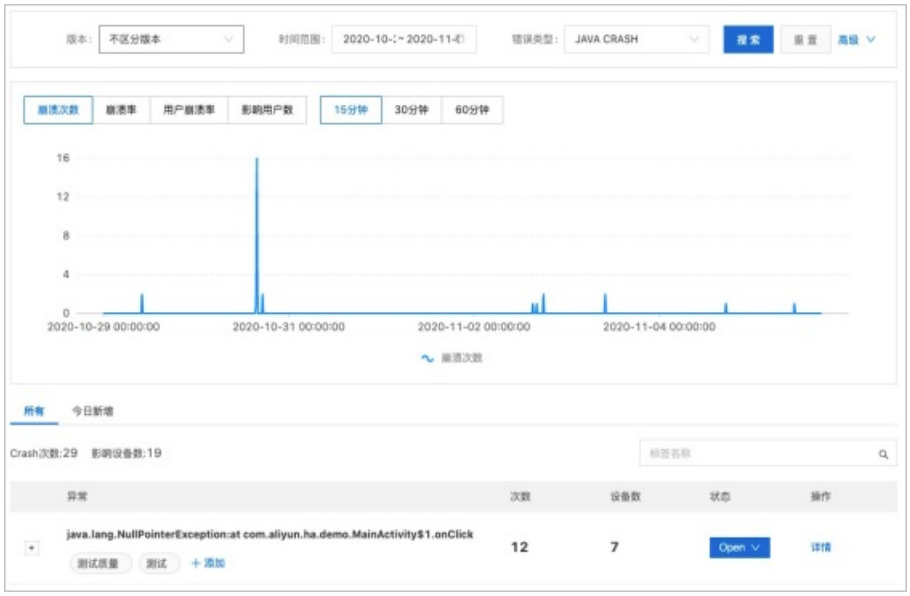
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。

查看问题数据明细

在左侧导航栏单击**崩溃**、**卡顿**或**异常**，然后在页面顶部选择不同版本、时间范围、错误类型，单击**搜索**。

> **说明** 单击**高级**，可以增加关键字搜索。



在页面下方的错误类型聚合数据右侧，单击**详情**，进入错误聚合详情页面。更多信息，请参见查看错误聚合详情。

查看错误聚合详情

基本信息

基本信息区域介绍了各种错误类型的聚合详情基本信息。

参数	说明
ID	此条错误的唯一标识。
发生次数	所选时间段内此错误发生的总次数。
影响用户数	所选时间段内此错误发生后影响的设备数。

在**基本信息**区域您可以执行以下操作：

- 添加标签：单击**添加**，可为此条错误添加标签。
- 修改状态：单击错误详情右上角的下拉框，您可以更改此错误类型的状态为New、Open或Fixed。不同状态的含义如下：
  - New：新出现
  - Fixed：已被修复
  - Open：被修复后再次出现

问题分析

问题分析页签分为趋势对比、调用栈分析、特征分析3个功能模块：

- 趋势对比：您可以通过选择版本、时间，对两个独立天的数据进行对比。

- 调用栈分析：查看问题发生的堆栈。

- 特征分析：展示此聚合详情中不同品牌、进程、页面、地市、机型、渠道、系统版本、View、应用版本维

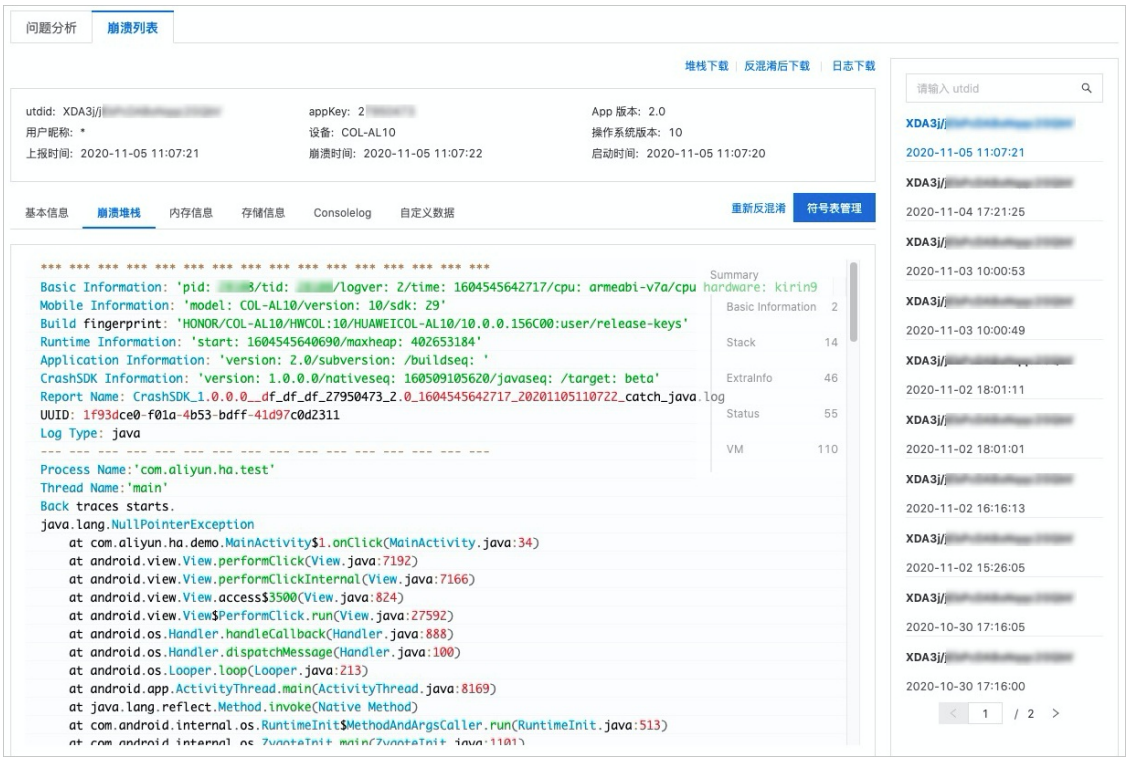
度下个值的占比情况。

> **说明** 单击单图右上角切换图标，可在饼图和列表的展示形式之间切换。

崩溃列表

崩溃列表中展示了聚合到当前错误中的所有错误实例，每条实例代表一个发生此错误的设备。

崩溃列表页签右侧展示所有的崩溃信息，单击某一条数据，左侧页面将会展示此条崩溃信息的基本信息、崩溃堆栈、内存信息、存储信息、ConsoleLog和自定义数据。



在**崩溃列表**页签，您可以执行以下操作：

- **堆栈下载**：下载原始堆栈信息。
- **反混淆后下载**：下载反混淆后的堆栈信息。
- **日志下载**：下载系统日志。
- **重新反混淆**：重新执行反混淆，执行后可单击反混淆下载，下载最新反混淆后的堆栈信息。
- **符号表管理**：配置符号表。

基本信息区域展示了此条崩溃信息的全部基本信息。

参数	说明
appKey	接入崩溃分析时使用的AppKey
应用版本	接入崩溃分析时传入的应用版本
构建号	暂无意义
渠道	接入崩溃分析时传入的渠道标识
记录ID	本次崩溃的ID
聚合ID	本次崩溃的同类崩溃聚合ID
聚合类型	崩溃类型
上报时间	上报时间，服务端时间
启动时间	App启用时间，客户端时间
崩溃时间	崩溃发生时间，客户端时间
品牌	崩溃发生设备的品牌
机型	崩溃发生设备的型号标识
操作系统/版本	崩溃发生时设备的操作系统及系统版本
运营商/网络	崩溃发生时设备接入的运营商和网络信息
国家/地区	崩溃发生时设备所处的地理位置，根据IP推算
用户IP	崩溃发生时设备的公网IP
分辨率	崩溃发生设备的屏幕分辨率
View	崩溃发生时App打开的视图
Page	崩溃发生时App打开的页面
是否是前台	崩溃是否发生在前台进程
是否Root	崩溃发生设备是否已被Root
是否是主线程	崩溃是否发生在主线程

参数	说明
异常类型	崩溃的异常类型
Native模块	崩溃相关的Native模块
所属模块	暂无意义
进程名	崩溃进程的名称
父进程名	崩溃进程的父进程名称

崩溃堆栈区域展示了崩溃发生时的堆栈及相关的线程、App所处状态等信息。

内存信息区域展示了崩溃繁盛时内存使用状态。

Android

参数	说明
MemTotal	系统可用总内存，不包括kernel占用的内存，这个值在系统运行期间一般是固定不变的
MemFree	系统尚未使用的内存
MemAvailable	kernel估算出的可用内存
Buffers	块设备（block device）所占用的缓存页
Cached	普通文件所占用的缓存页，包含已被进程解除关联
SwapCached	匿名页（anonymous pages）使用到的交换缓存区
Active	最近被访问过的内存页
Inactive	长时间未被访问过的内存页
Active(anon)	最近被访问过的匿名内存页
Inactive(anon)	长时间未被访问过的匿名内存页
Active(file)	最近被访问过的文件内存页
Inactive(file)	长时间未被访问过的文件内存页
Unevictable	不能pageout/swapout的内存页
Mlocked	被mlock()系统调用锁定的内存大小
SwapTotal	交换缓存区总大小
SwapFree	交换缓存区空闲大小
Dirty	等待被写回磁盘的缓存也页

参数	说明
Writeback	正准备回写硬盘的缓存页
AnonPages	匿名内存页
Mapped	普通文件所占用的缓存页
Shmem	共享内存，以及tmpfs和devtmpfs
Slab	通过slab分配的总内存
SReclaimable	slab中可回收的部分
SUnreclaim	slab中不可回收的部分
KernelStack	内核栈
PageTables	用于虚拟地址映射的内存页
NFS_Unstable	发给NFS server但尚未写入硬盘的缓存页
Bounce	内存跳转buffer使用的内存
WritebackTmp	FUSE使用的临时缓冲
CommitLimit	基于overcommit ratio计算得到的系统可分配内存
Committed_AS	完成当前负载预计需要的内存
VmallocTotal	vmalloc可分配的总虚拟内存
VmallocUsed	通过vmalloc分配的虚拟内存
VmallocChunk	可用的最大连续虚拟内存空间
AnonHugePages	匿名HugePages大小
HugePages_Total	系统HugePages的总大小
HugePages_Free	可用的HugePages大小
HugePages_Rsvd	申请后还未使用的HugePages大小
HugePages_Surp	剩余的HugePages大小
Hugepagesize	HugePages大小
DirectMap4k	映射为4 KB的内存数量
DirectMap2M	映射为2 MB的内存数量

说明 Android应用发生崩溃时的内存信息取自/proc/meminfo。

iOS

参数	说明
Memory Usage	内存使用量

存储信息区域展示了Android应用崩溃发生时存储使用状态。

参数	说明
hasSDCard	是否有SDCard
RootDirectory	*/system*系统目录
DataDirectory	*/data*数据目录
ExternalStorageDirectory	*/storage/emulated/0*外部储存目录
DownloadCacheDirectory	*/data/cache*下载缓存目录
TotalSize(Byte)	总大小
FreeSize(Byte)	空闲大小
AvailableSize(Byte)	可用大小

Consolelog区域展示了崩溃发生前后一段时间的系统日志。

自定义数据区域展示了崩溃发生时通过SDK回调来上报的自定义数据。

4.3. 高级搜索

在**高级搜索**页面您可以查看指定时间段内的Crash次数及影响设备数，以及指定时间段内最近5条、最近10条、最近15条的崩溃数据。

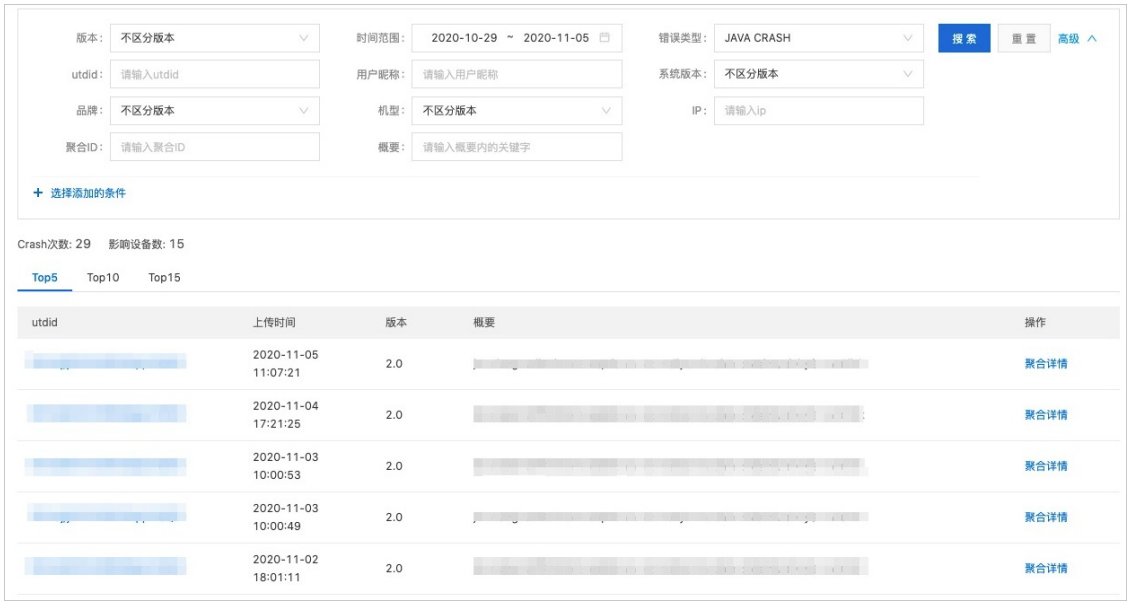
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。

功能说明

在左侧导航栏单击**高级搜索**，然后在右侧页面顶部选择不同版本、时间范围、错误类型，单击**搜索**。

> **说明** 您可以单击**高级**，打开高级搜索增加搜索项。



在查询到的崩溃数据右侧，单击**聚合详情**，进入错误聚合详情页面。更多信息，请参见查看错误聚合详情。

查看错误聚合详情

基本信息

基本信息区域介绍了各种错误类型的聚合详情基本信息。

参数	说明
ID	此条错误的唯一标识。
发生次数	所选时间段内此错误发生的总次数。
影响用户数	所选时间段内此错误发生后影响的设备数。

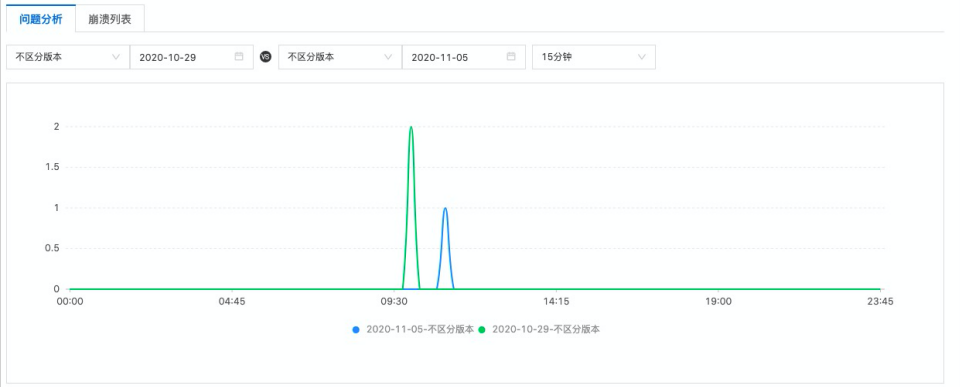
在**基本信息**区域您可以执行以下操作：

- 添加标签：单击**添加**，可为此条错误添加标签。
- 修改状态：单击错误详情右上角的下拉框，您可以更改此错误类型的状态为New、Open或Fixed。不同状态的含义如下：
  - New：新出现
  - Fixed：已被修复
  - Open：被修复后再次出现

问题分析

问题分析页签分为趋势对比、调用栈分析、特征分析3个功能模块：

- 趋势对比：您可以通过选择版本、时间，对两个独立天的数据进行对比。



- 调用栈分析：查看问题发生的堆栈。

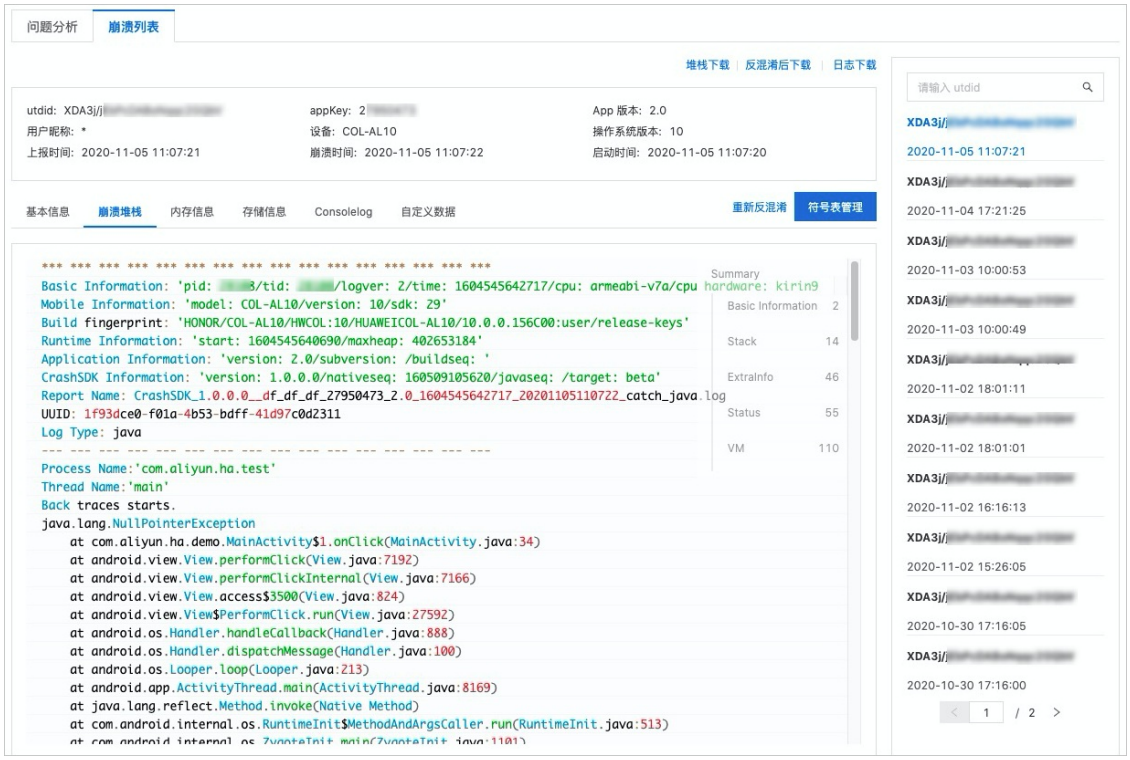
- 特征分析：展示此聚合详情中不同品牌、进程、页面、地市、机型、渠道、系统版本、View、应用版本维度下个值的占比情况。

> **说明** 单击单图右上角切换图标，可在饼图和列表的展示形式之间切换。

崩溃列表

崩溃列表中展示了聚合到当前错误中的所有错误实例，每条实例代表一个发生此错误的设备。

崩溃列表页签右侧展示所有的崩溃信息，单击某一条数据，左侧页面将会展示此条崩溃信息的基本信息、崩溃堆栈、内存信息、存储信息、ConsoleLog和自定义数据。



在**崩溃列表**页签，您可以执行以下操作：

- **堆栈下载**：下载原始堆栈信息。
- **反混淆后下载**：下载反混淆后的堆栈信息。
- **日志下载**：下载系统日志。
- **重新反混淆**：重新执行反混淆，执行后可单击反混淆下载，下载最新反混淆后的堆栈信息。
- **符号表管理**：配置符号表。

基本信息区域展示了此条崩溃信息的全部基本信息。

参数	说明
appKey	接入崩溃分析时使用的AppKey
应用版本	接入崩溃分析时传入的应用版本
构建号	暂无意义
渠道	接入崩溃分析时传入的渠道标识
记录ID	本次崩溃的ID
聚合ID	本次崩溃的同类崩溃聚合ID
聚合类型	崩溃类型
上报时间	上报时间，服务端时间
启动时间	App启用时间，客户端时间
崩溃时间	崩溃发生时间，客户端时间

参数	说明
品牌	崩溃发生设备的品牌
机型	崩溃发生设备的型号标识
操作系统/版本	崩溃发生时设备的操作系统及系统版本
运营商/网络	崩溃发生时设备接入的运营商和网络信息
国家/地区	崩溃发生时设备所处的地理位置，根据IP推算
用户IP	崩溃发生时设备的公网IP
分辨率	崩溃发生设备的屏幕分辨率
View	崩溃发生时App打开的视图
Page	崩溃发生时App打开的页面
是否是前台	崩溃是否发生在前台进程
是否Root	崩溃发生设备是否已被Root
是否是主线程	崩溃是否发生在主线程
异常类型	崩溃的异常类型
Native模块	崩溃相关的Native模块
所属模块	暂无意义
进程名	崩溃进程的名称
父进程名	崩溃进程的父进程名称

崩溃堆栈区域展示了崩溃发生时的堆栈及相关的线程、App所处状态等信息。

内存信息区域展示了崩溃繁盛时内存使用状态。

Android

参数	说明
MemTotal	系统可用总内存，不包括kernel占用的内存，这个值在系统运行期间一般是固定不变的
MemFree	系统尚未使用的内存
MemAvailable	kernel估算出的可用内存
Buffers	块设备（block device）所占用的缓存页

参数	说明
Cached	普通文件所占用的缓存页，包含已被进程解除关联
SwapCached	匿名页（anonymous pages）使用到的交换缓存区
Active	最近被访问过的内存页
Inactive	长时间未被访问过的内存页
Active(anon)	最近被访问过的匿名内存页
Inactive(anon)	长时间未被访问过的匿名内存页
Active(file)	最近被访问过的文件内存页
Inactive(file)	长时间未被访问过的文件内存页
Unevictable	不能pageout/swapout的内存页
Mlocked	被mlock()系统调用锁定的内存大小
SwapTotal	交换缓存区总大小
SwapFree	交换缓存区空闲大小
Dirty	等待被写回磁盘的缓存也页
Writeback	正准备回写硬盘的缓存页
AnonPages	匿名内存页
Mapped	普通文件所占用的缓存页
Shmem	共享内存，以及tmpfs和devtmpfs
Slab	通过slab分配的总内存
SReclaimable	slab中可回收的部分
SUnreclaim	slab中不可回收的部分
KernelStack	内核栈
PageTables	用于虚拟地址映射的内存页
NFS_Unstable	发给NFS server但尚未写入硬盘的缓存页
Bounce	内存跳转buffer使用的内存
WritebackTmp	FUSE使用的临时缓冲
CommitLimit	基于overcommit ratio计算得到的系统可分配内存

参数	说明
Committed_AS	完成当前负载预计需要的内存
VmallocTotal	vmalloc可分配的总虚拟内存
VmallocUsed	通过vmalloc分配的虚拟内存
VmallocChunk	可用的最大连续虚拟内存空间
AnonHugePages	匿名HugePages大小
HugePages_Total	系统HugePages的总大小
HugePages_Free	可用的HugePages大小
HugePages_Rsvd	申请后还未使用的HugePages大小
HugePages_Surp	剩余的HugePages大小
Hugepagesize	HugePages大小
DirectMap4k	映射为4 KB的内存数量
DirectMap2M	映射为2 MB的内存数量

说明 Android应用发生崩溃时的内存信息取自/proc/meminfo。

iOS

参数	说明
Memory Usage	内存使用量

存储信息区域展示了Android应用崩溃发生时存储使用状态。

参数	说明
hasSDCard	是否有SDCard
RootDirectory	*/system*系统目录
DataDirectory	*/data*数据目录
ExternalStorageDirectory	*/storage/emulated/0*外部储存目录
DownloadCacheDirectory	*/data/cache*下载缓存目录
TotalSize(Byte)	总大小
FreeSize(Byte)	空闲大小

参数	说明
AvailableSize(Byte)	可用大小

Consolelog区域展示了崩溃发生前后一段时间的系统日志。

自定义数据区域展示了崩溃发生时通过SDK回调来上报的自定义数据。

4.4. 多维分析

多维分析功能根据App版本、页面、地域、网络和设备维度聚合分析崩溃数据情况。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。

版本分析

在左侧导航栏单击**多维分析 > 版本**，然后在右侧页面顶部选择筛选条件，查看各版本的崩溃数据。

版本分析是根据App版本聚合分析崩溃数据情况。

筛选条件：

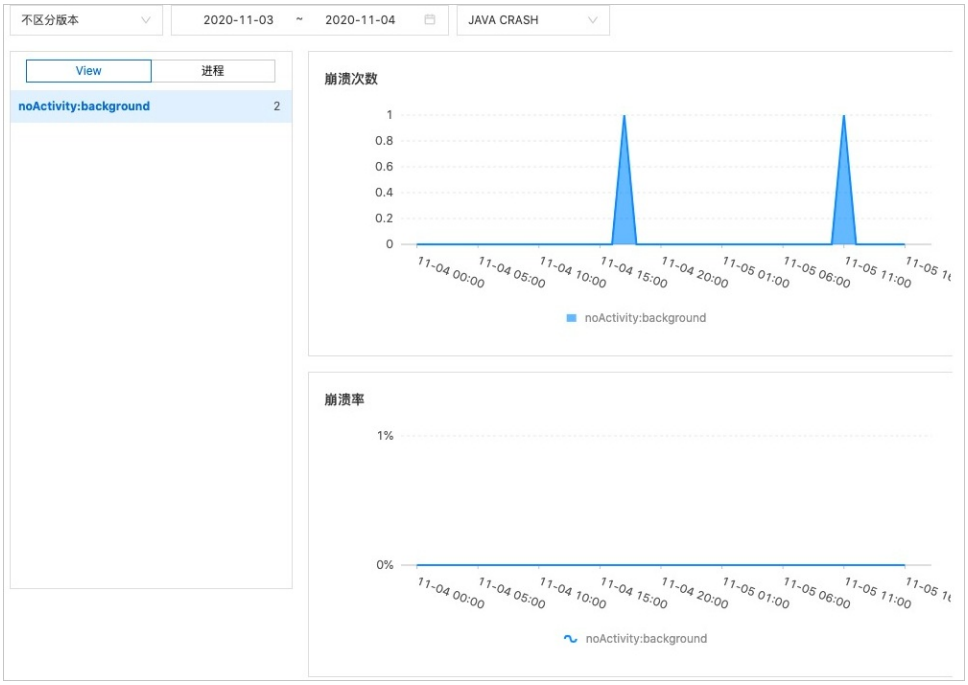
- UV最大日：历史上该版本UV最大的一天。
- 发版后第二日：该版本开始有上报数据的第二天。
- 中间平均：表示该版本开始有上报数据的第2天到第32天，取31天的平均值。

日期	版本	启动次数	Java Crash 次数	Java Crash 率	Native Crash 次数	Native Crash 率	ANR 次数	ANR 率	总体 Crash 率
2020-10-30	1.0	156	17	10.90%	0	0.00%	0	0.00%	10.90%
2020-09-22	2.0	43	6	13.95%	0	0.00%	0	0.00%	13.95%
2020-09-18	1.3.1	2	0	0.00%	0	0.00%	0	0.00%	0.00%
2020-10-27	1.0.0	8	0	0.00%	0	0.00%	0	0.00%	0.00%

页面分析

在左侧导航栏单击**多维分析 > 页面**，然后在右侧页面顶部选择不同版本、时间段、异常指标，查看不同应用页面（View）、应用进程的崩溃数据。

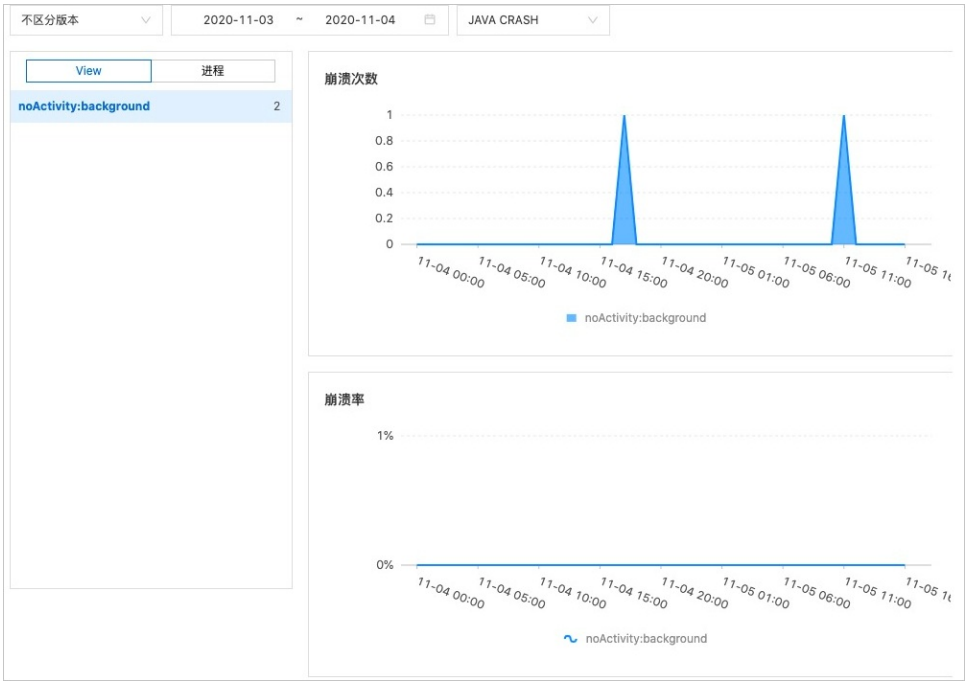
页面分析是按应用崩溃页面以及崩溃进程聚合分析不同应用页面和应用进程下应用的崩溃数据。



地域分析

在左侧导航栏单击**多维分析 > 地域**，然后在右侧页面顶部选择不同版本、时间段、异常指标，查看不同地域（国家/地区、省份、地市）的崩溃数据。

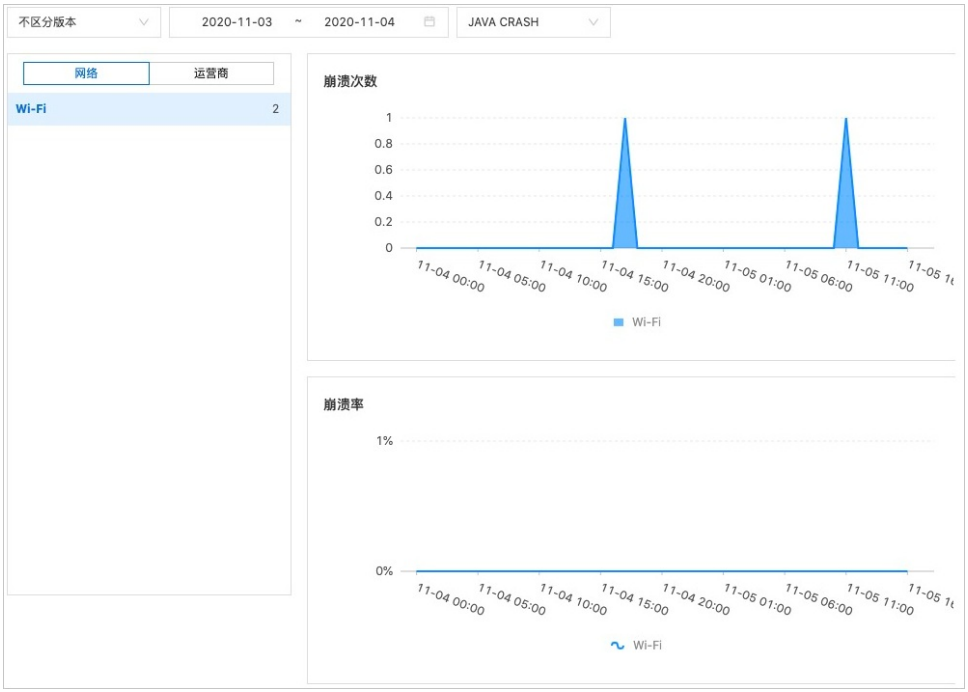
地域分析是按崩溃机型所在地域信息聚合分析不同地域的应用崩溃数据。



网络分析

在左侧导航栏单击**多维分析 > 网络**，然后在右侧页面顶部选择不同版本、时间段、异常指标，查看不同网络、运营商的崩溃数据。

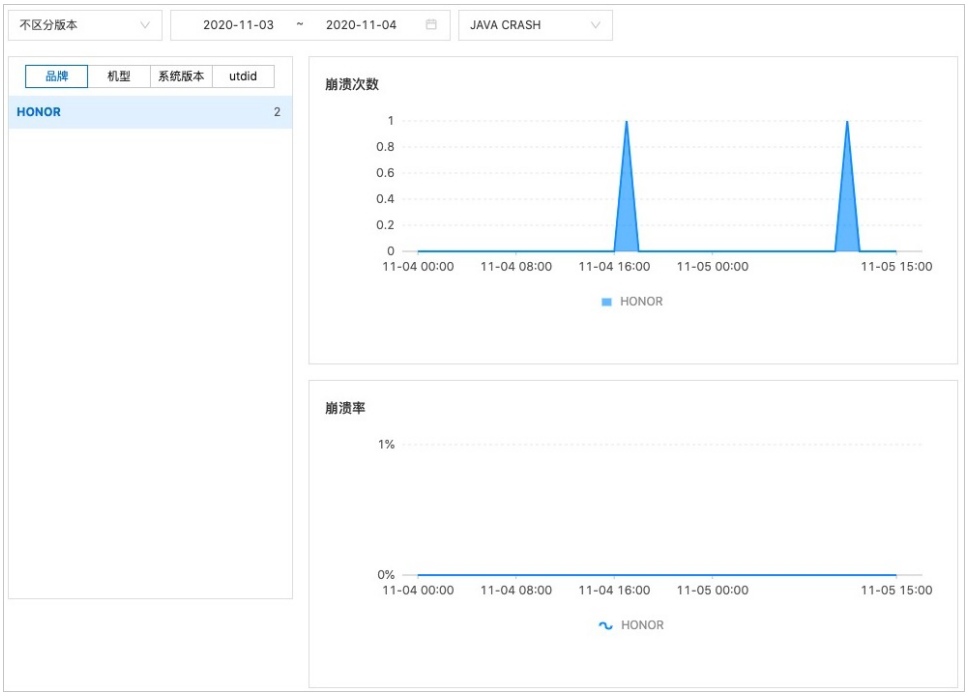
网络分析是按崩溃机型接入的网络和运营商聚合分析不同网络条件下的应用崩溃数据。



设备分析

在左侧导航栏单击**多维分析 > 设备**，然后在右侧页面顶部选择不同版本、时间段、异常指标，查看不同品牌、机型、系统版本、utdid的崩溃数据。

设备分析是将崩溃数据按品牌、机型、系统版本、设备ID（utdid）进行聚合分析。



4.5. 告警管理

4.5.1. 管理联系人

崩溃分析服务支持告警服务，您可以在崩溃分析控制台上管理告警联系人和联系人组，以便于接收告警信息。

功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。
4. 在左侧导航栏中选择**设置 > 告警管理**，然后单击**联系人管理**页签。

新建告警联系人

1. 在**联系人管理 > 联系人**页签，单击**新建联系人**。
2. 在**新增联系人**面板设置联系人姓名、手机号、Email、钉钉机器人，然后单击**确定**。

> **说明** 获取钉钉机器人Webhook地址的操作，请参见获取钉钉机器人Webhook地址。

新建联系人组

1. 在**联系人管理 > 联系人组**页签，单击**新建联系人组**。
2. 在**新增联系人组**面板设置联系人组名称，选择创建好的告警联系人，然后单击**确定**。

4.5.2. 新建告警规则

崩溃分析服务支持告警服务，您可以在崩溃分析控制台配置告警规则，当异常发生时，会自动发送告警。

前提条件

已创建告警联系人并将之添加到告警联系组，具体操作，请参见管理联系人。

操作步骤

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。
4. 在左侧导航栏中选择**设置 > 告警管理**，然后单击**告警规则**页签。
5. 单击**新增规则**。
6. 在**新增规则**面板，设置如下参数，然后单击**确定**。

参数	说明
规则名称	输入告警规则的名称。
应用版本	选择告警规则所覆盖的应用版本。
异常类型	选择需要配置告警的指标项。
条件之间的关系	选择告警条件的生效规则。 ○ 同时满足下述条件 ○ 满足下述一条规则即可

参数	说明
条件	可根据对选择的异常类型选择4种条件。 ○ Crash次数 ○ Crash率 ○ 影响用户数 ○ 用户Crash率
通知间隔	选择告警通知发送的频率。
允许通知时间段	选择发送告警通知的时间段。
告警途径	选择告警发送的方式，可选择邮件、短信、钉钉。 **说明** 需要告警联系人配置对应的联系方式后才能收到告警。
联系人组	选择配置好的联系人组。

告警规则创建完成后，您可以在**告警管理**页签查看所有已创建的告警规则。

说明 对于暂时不需要告警的规则，您可以单击规则右侧的**禁用**或**恢复**来控制是否生效。

4.5.3. 查看告警历史

您可以在**告警历史**页签查看历史触发的告警记录。

操作步骤

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App监控**。
3. 在**App监控**页面，单击目标App右侧的**崩溃分析**。
4. 在左侧导航栏中选择**设置 > 告警管理**，然后单击**告警历史**页签。
5. 在**告警历史**页签，查看所有告警信息，您也可以通过规则名称、时间范围来搜索告警信息。

6. （可选）单击告警右侧的分析处理，可以转跳至崩溃数据查看页面。

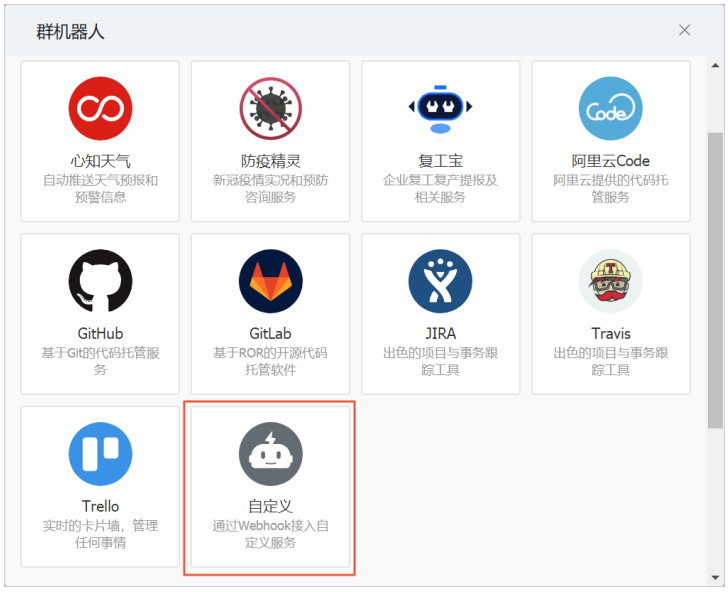
> **说明** 告警的分析处理页面，可以修改当前缺陷类型的状态，分为New、Open、Fixed。

4.5.4. 获取钉钉机器人Webhook地址

使用钉钉机器人，可将崩溃分析服务产生的告警信息，实时推送至指定钉钉群。

操作步骤

1. 在PC版钉钉上打开您想要添加报警机器人的钉钉群，并单击右上角的群设置图标。
2. 在**群设置**面板中单击**智能群助手**。
3. 在**智能群助手**面板单击**添加机器人**。
4. 在**群机器人**对话框单击**添加机器人**区域的 **+** 图标，然后选择添加**自定义**。



5. 在**机器人详情**对话框单击**添加**。
6. 在**添加机器人**对话框中执行以下操作。

i. 设置机器人头像和名字。

ii. **安全设置**选中**自定义关键词**，设置关键词为EMAS。

iii. 选中**我已阅读并同意《自定义机器人服务及免责条款》**。

iv. 单击**完成**。

> **说明** 更多关于钉钉机器人的操作，请参见自定义机器人接入。

7. 在**添加机器人**对话框中复制生成的机器人Webhook地址，然后单击**完成**。

5.性能分析
5.1. 概览
5.1.1. 启动速度

本文介绍指定条件下，移动应用的启动耗时分布统计、移动设备数和启动耗时随时间变化的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**概览 > 启动速度**。

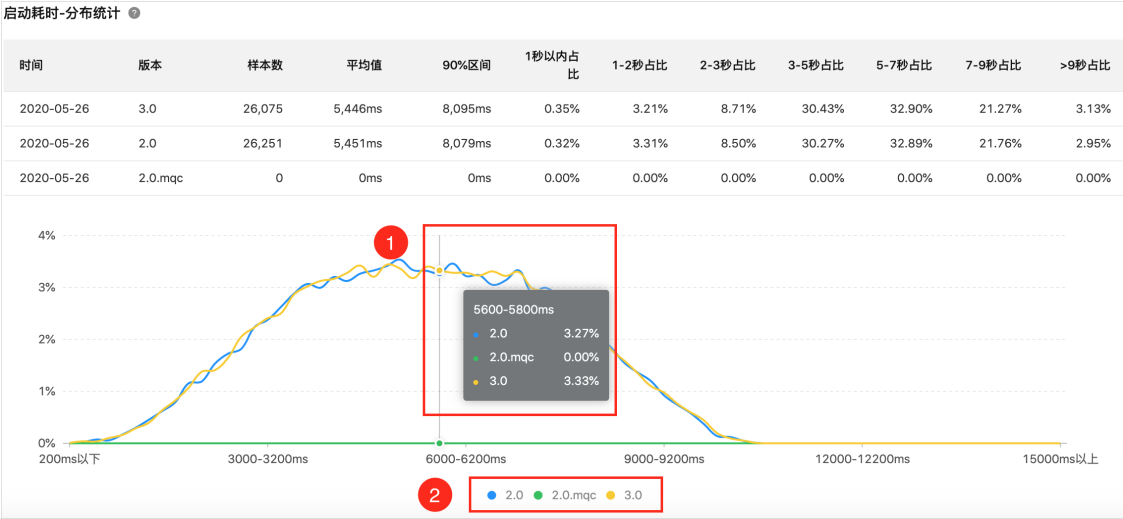
条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
日期	指定统计的日期。
启动类型	指定移动应用的启动类型。
机型	指定计入统计的设备类型。
地区	指定计入统计的设备所在地区，包括国内主要省市自治区。

显示数据说明

分布统计

显示指定条件下，移动应用启动耗时的分布情况。



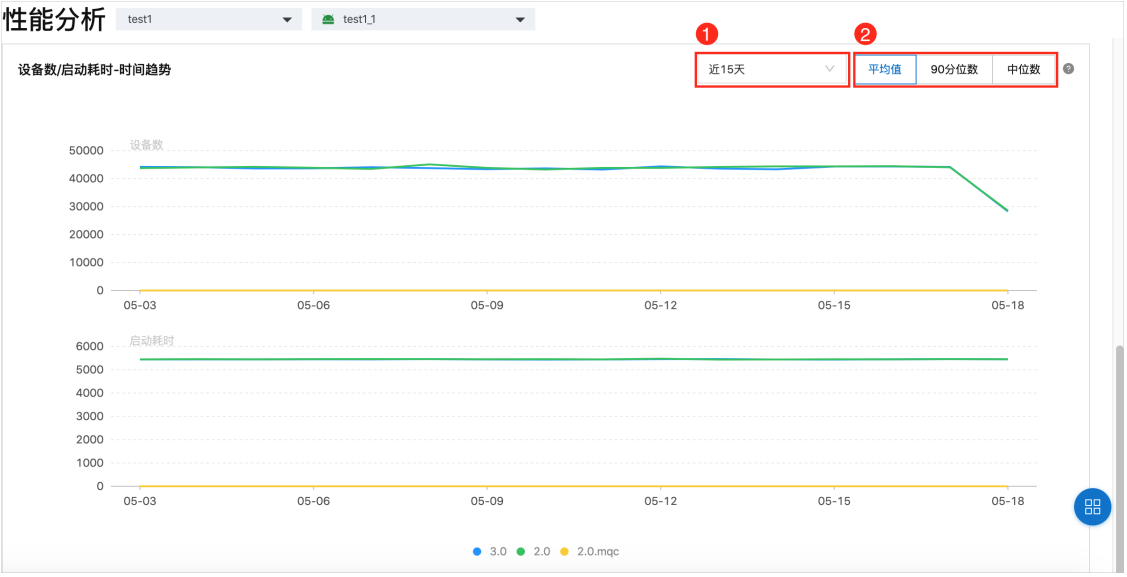
选项	说明
时间	统计的日期。

选项	说明
版本	移动应用的版本。
样本数	启动移动应用，并且计入统计的设备数量。
平均值	基于已有设备样本启动移动应用的平均时长。
90%区间	基于已有设备样本启动移动应用，其中占比90%的启动时长。
N秒占比	基于已有设备样本启动移动应用，时长为N秒的设备数占比。

> **说明**
>
> - 将鼠标移动至分布统计图上方，显示指定启动耗时的情况下，各版本的占比情况。
> - 在分布统计图中，以版本为单位，使用不同颜色的曲线分别显示。

时间趋势

显示指定条件下，移动设备数和启动耗时随时间变化的情况。



选项	说明
设备数	指定条件下，设备数随时间变化的统计数据。
启动耗时	指定条件下，移动应用启动耗时随时间变化的统计数据。

选项设置：

序号	选项	说明
1	时间区间	用于指定统计数据的时间区间。
2	统计类型	用于指定统计数据的类型。

> **说明**
> - 将鼠标移动至时间趋势上方，显示指定时间节点，各版本的统计数据。
> - 在时间趋势中，以版本为单位，使用不同颜色的曲线分别显示。

5.1.2. 页面性能

本文介绍指定条件下，页面加载耗时和滑动帧率平均值、加载耗时和滑动帧率的分布统计，以及设备数/滑动帧率随时间变化的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**概览 > 页面性能**。

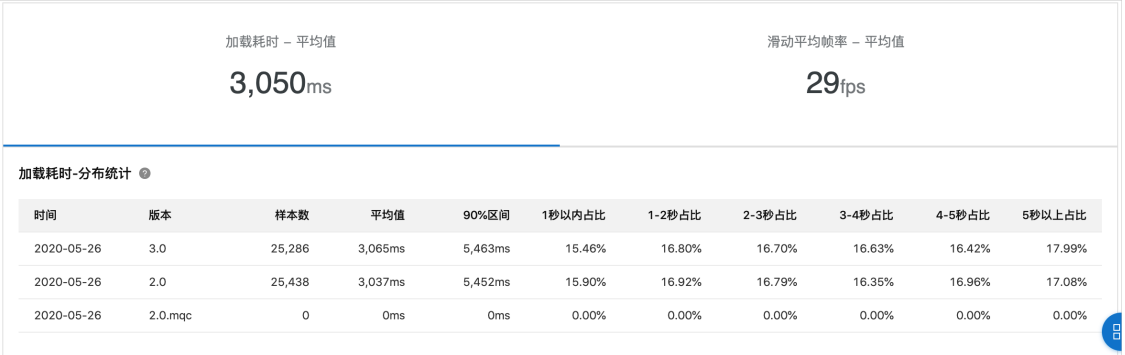
条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
日期	指定统计的日期。
机型	指定计入统计的设备类型。
地区	指定计入统计的设备所在地区，包括国内主要省、市、自治区。

显示数据说明

平均值

平均值包括加载耗时平均值和滑动帧率平均值。



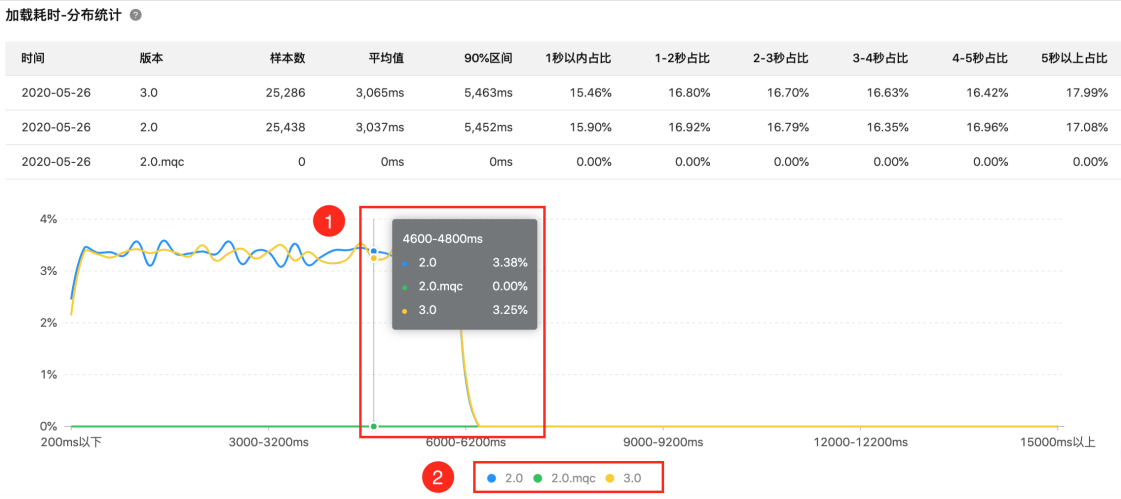
时间	版本	样本数	平均值	90%区间	1秒以内占比	1-2秒占比	2-3秒占比	3-4秒占比	4-5秒占比	5秒以上占比
2020-05-26	3.0	25,286	3,065ms	5,463ms	15.46%	16.80%	16.70%	16.63%	16.42%	17.99%
2020-05-26	2.0	25,438	3,037ms	5,452ms	15.90%	16.92%	16.79%	16.35%	16.96%	17.08%
2020-05-26	2.0.mqc	0	0ms	0ms	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

选项	说明
加载耗时平均值	基于已有设备样本加载页面的平均时长。
滑动平均帧率平均值	基于已有设备样本每秒可加载页面的数量。

> **说明** 单击加载耗时平均值或滑动帧率平均值区域，分布统计图表显示相应的分布统计数据。

分布统计

显示指定条件下，加载耗时和滑动平均帧率在各时间区间的占比情况。

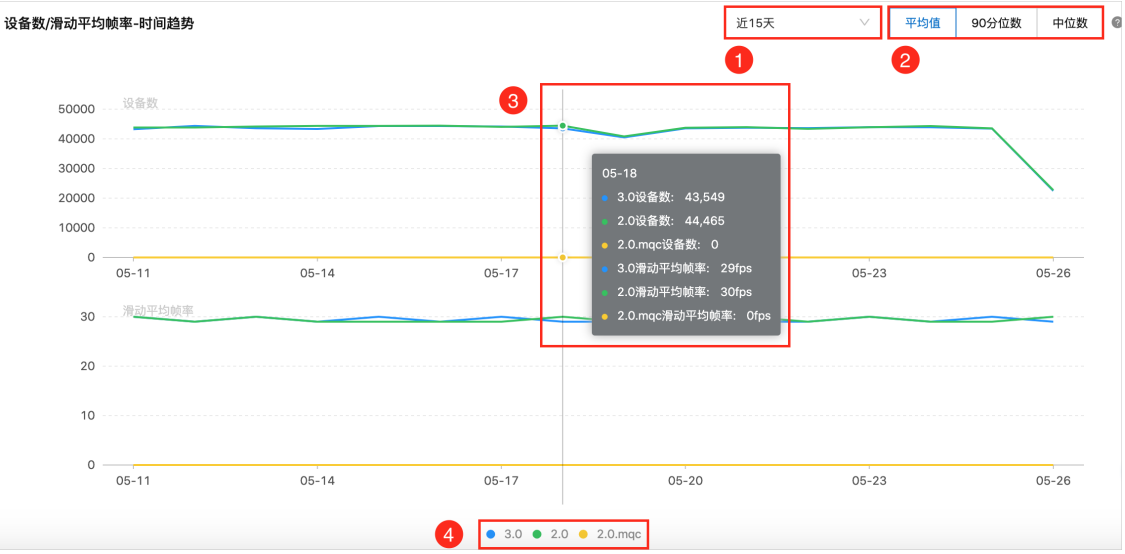


选项	说明
时间	统计的日期。
版本	移动应用的版本。
样本数	加载页面，并且计入统计的设备的数量。
平均值	基于已有设备样本加载页面的平均时长。
90%区间	基于已有设备样本加载页面，其中占比90%的加载时长。
N秒占比	基于已有设备样本加载页面，时长为N秒的设备数占比。

> **说明**
>
> - 将鼠标移动至分布统计图上方，显示指定数值场景下，各版本的占比情况。
> - 在分布统计图中，以版本为单位，使用不同颜色的曲线分别显示。

时间趋势

显示指定条件下，设备数量和滑动平均帧率随时间变化的情况。



选项设置：

序号	选项	说明
1	时间区间	指定统计数据的时间区间。
2	统计类型	指定统计数据的类型。

> **说明**
> - 将鼠标移动至时间趋势图上方，显示指定时间节点，各版本的统计值。
> - 在时间趋势图中，以版本为单位，使用不同颜色的曲线分别显示。

5.1.3. 网络请求

本文介绍移动应用网络性能的统计数据，主要包括平均响应时间、网络错误率和HTTP错误率等。

前提条件

已添加网络请求白名单。具体操作，请参见设置URL过滤。

> **说明** 如未预先设置URL过滤，在打开**网络请求**页面时，将会弹出**数据设置**对话框，单击**确定**跳转至**URL设置**页面。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**概览 > 网络请求**。

条件设置说明

选项	说明
版本	用于指定一个或多个移动应用的版本号。
时间	用于指定时间区间。
域名	用于指定域名类型。 说明 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

统计数据

显示网络请求的统计数据，如下图所示。

平均响应时间列表

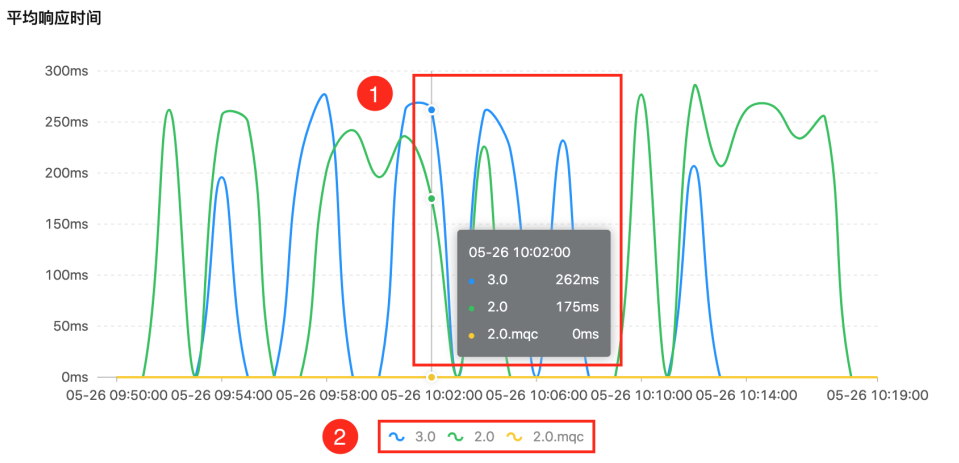
以接口或者IP地址为统计单位，显示平均响应时间。

选项设置：

序号	选项	说明
1	搜索域名	输入域名关键字，查询指定域名的平均响应时间。
2	平均响应时间	按照平均响应时间的升序或降序显示接口或IP。
3	接口/IP	以接口或IP为统计单位，显示其平均响应时间。

趋势图

显示其网络请求的平均响应时间、网络错误率和HTTP错误率随时间变化的情况。此处以平均响应时间为例：



说明

- 将鼠标移动至趋势图上方，显示指定时间点各版本的瞬时值。
- 在请求时间占比分布图中，以版本为单位，使用不同颜色的曲线分别显示。

5.2. 启动

本文基于机型和运营商维度，介绍其对移动应用启动时间的影响。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**启动 > Top100机型**或**启动 > 运营商**。

条件设置说明

Top 100机型和**运营商**页面的条件设置选项相同，包括：

选项	说明
版本	指定一个或多个移动应用的版本号。
日期	指定统计的日期。
启动类型	指定移动应用的启动类型。

显示数据说明

Top 100机型

选项	说明
机型	Top 100各机型的名称。
平均值	指定机型启动移动应用的平均时长。
90分位数	指定机型启动移动应用时，占比90%的启动时长。
该维度值样本数	指定机型启动移动应用时，计入统计的设备数量。

选项	说明
该维度值样本占比	指定机型启动移动应用时，计入统计的设备占设备总数的比例。
N秒占比	指定机型启动移动应用时，时长为N秒的设备数占比。

> **说明** 在列表中，单击平均值、90分位数、该维度值样本数、**该维度值样本占比**列的表头，可针对指定列进行排序。

运营商

- 启动耗时对比分析：运营商启动耗时对比分析用于对比各运营商的启动耗时情况。

选项	说明
运营商	运营商的名称。
平均值	指定运营商设备启动移动应用的平均时长。
90分位数	指定运营商设备启动移动应用时，占比90%的启动时长。
样本数	指定运营商设备启动移动应用时，计入统计的设备数量。
样本占比	指定运营商设备启动移动应用时，计入统计的设备占设备总数的比例。
N秒占比	指定运营商设备启动移动应用时，时长为N秒的设备数占比。

> **说明** 在列表中，单击平均值、90分位数、样本数、样本占比列的表头，可针对指定列进行排序。

- 启动耗时长尾分析：运营商启动耗时长尾分析用于对比各运营商启动时间大于8000ms的情况占比。

5.3. 页面

在**页面**页签您可以查询指定条件下，移动应用加载页面的加载耗时和滑动帧率。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**页面 > 加载耗时**或**页面 > 滑动帧率**。

条件设置说明

加载耗时和**滑动帧率**页面的条件设置选项相同，包括：

选项	说明
版本	指定一个或多个移动应用的版本号。
日期	指定计入统计的日期。
机型	指定计入统计的设备类型。

选项	说明
地区	指定计入统计的设备所在地区，包括国内主要省市自治区。

显示数据说明

加载耗时

选项	说明
页面	统计的页面。
时间	统计的日期。
版本	移动应用的版本。
样本数	计入统计的设备的数量。
平均值	基于已有设备样本启动移动应用的平均时长。
90分位数	基于已有设备样本启动移动应用，其中占比90%的启动时长。
N秒占比	基于已有设备样本启动移动应用，时长为N秒的设备数占比。

滑动帧率

选项	说明
页面	统计的页面。
时间	统计的日期。
版本	移动应用的版本。
样本数	计入统计的设备的数量。
平均值	基于已有设备样本加载页面的平均滑动帧率。
90分位数	基于已有设备样本加载页面，其中占比90%的滑动帧率。
N fps占比	基于已有设备样本加载页面，滑动帧率为N fps的设备数占比。

5.4. 网络

5.4.1. 响应时间

在**响应时间**页面您可以按照指定条件，查询移动应用的平均响应时间、分布统计，以及随时间变化的趋势。

前提条件

已添加网络请求白名单。具体操作，请参见设置URL过滤。

> **说明** 如未预先设置URL过滤，在打开**网络请求**页面时，将会弹出**数据设置**对话框，单击**确定**跳转至**URL设置**页面。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 响应时间**。

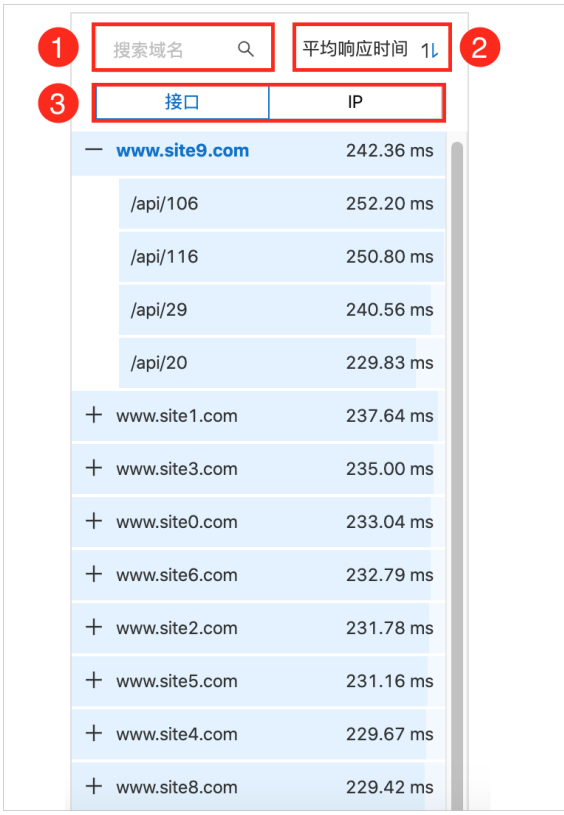
条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

平均响应时间

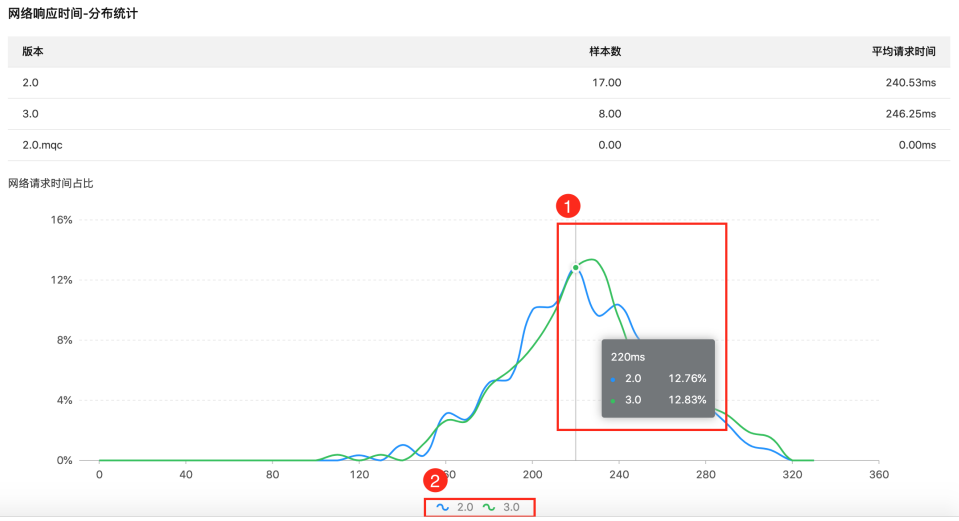
在左侧列表显示接口和IP的平均响应时间。



选项设置：

序号	选项	说明
1	搜索域名	输入域名关键字，查询指定域名的平均响应时间。
2	平均响应时间	按照平均响应时间的升序或降序显示接口或IP。
3	接口/IP	以接口或IP为统计单位，显示其平均响应时间。

分布统计

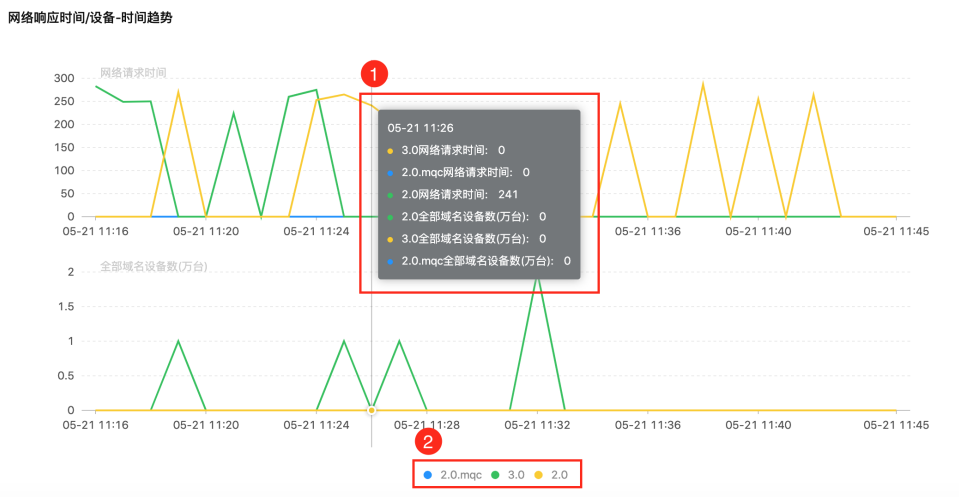


内容	说明
版本说明	以版本为单位，显示其样本数和平均请求时间。
网络请求时间占比图	以版本为单位，显示其网络请求时间占比分布。

说明

- 将鼠标移动至分布统计图上方，查看指定取值情况下，各版本的占比情况（图示中①）。
- 在请求时间占比分布图中，使用不同颜色的曲线分别显示不同的版本（图示中②）。

时间趋势



内容	说明
网络请求时间	网络请求时间的长度随时间变化的情况。
全部域名设备数	全部域名设备数随时间变化的情况。

> **说明**
> - 将鼠标移动至时间趋势图上方，查看指定时间点，各版本的网络请求时间和全部域名设备数。
> - 在时间趋势图中，使用不同颜色的曲线分别显示不同的版本。

5.4.2. 网络错误

在**网络错误**页面您可以按照指定条件，查询移动应用的错误率，以及错误率随时间分布的趋势。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 网络错误**。

条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

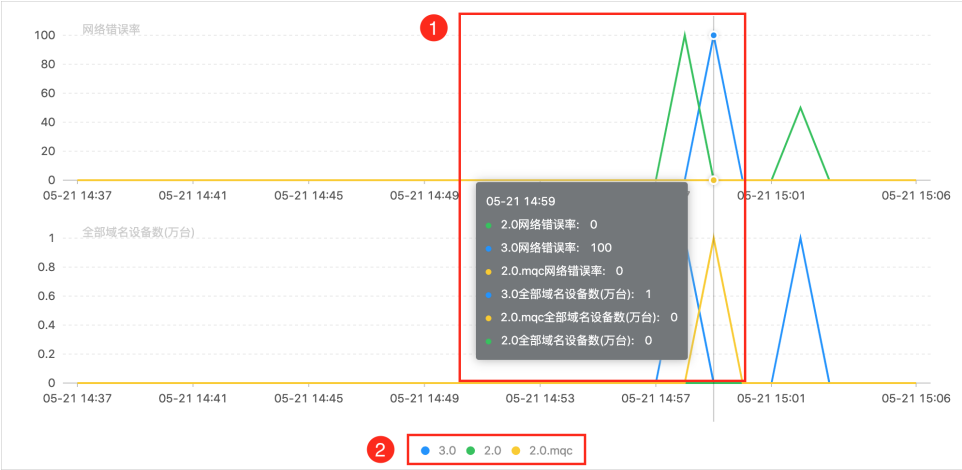
错误率

在左侧列表显示各域名的错误率。

选项设置：

序号	选项	操作说明
1	搜索域名	输入域名关键字，查询指定域名的错误率。
2	错误率	按照错误率的升序或降序显示域名。

错误率和设备数的时间趋势



内容	说明
版本说明	以版本为单位，显示其样本数、请求总数和错误率。
网络错误率	网络错误率随时间变化的情况。

内容	说明
全部域名设备数	全部域名设备数随时间变化的情况。单位为万台。

> **说明**
>
> - 将鼠标移动至时间趋势图上方，显示指定时间点，各版本的错误率和设备数情况（图示中①）。
> - 在时间趋势图中，以版本为单位，使用不同颜色的曲线，分别显示其错误率和设备数情况（图示中②）。

网络错误列表

网络错误列表显示网络错误的信息，如下图所示。

单击**操作**列的**查看详情**，进入**网络错误列表**页面，查询指定网络错误的详细信息。

在**网络错误列表**页面，设置过滤条件，单击**确定**，对错误信息进行筛选。过滤条件包括：

序号	选项	说明
1	UTDID	用于设置设备的ID。
2	机型	用于指定移动设备的品牌和型号。
3	系统	用于指定移动设备的系统和版本。
4	地域	用于指定移动设备所属行政地区。

单击**操作**列的**查看详情**，进入**错误详情**页面，查看指定错误的详细信息。

5.4.3. HTTP状态码

在**HTTP状态码**页面您可以按照指定条件，查询移动应用接受到的各类状态码的次数，和随时间变化的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > HTTP状态码**。

条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

状态码分3种类别进行显示，包括：

错误码类型	说明
1XX、2XX、3XX状态码	信息响应、成功响应、重定向
4XX状态码	客户端错误
5XX状态码	服务器错误

状态码次数

状态码次数以域名和接口为单位进行统计。

选项设置：

序号	选项	操作说明
1	搜索域名	输入域名关键字，查询指定域名的状态码次数。
2	状态码次数	按照状态码次数的升序或降序显示域名。

状态码时间趋势

状态码时间趋势，以4XX状态码为例，如下图所示。

内容	说明
版本说明	以版本为单位，显示其样本数和平均请求时间。

内容	说明
状态码或错误次数	以版本为单位，显示接收到的HTTP状态码次数随时间变化的情况。 说明 • 1XX、2XX、3XX状态码场景，显示**状态码次数**。 • 4XX、5XX状态码场景，显示**错误次数**。

说明

- 单击**错误（状态码）次数**或**错误（状态码）比例**，切换显示错误（状态码）次数或错误（状态码）比例随时间变化的情况（图示中①）。
- 将鼠标移动至趋势图上方，显示指定时间点的错误（状态码）次数或错误（状态码）比例（图示中②）。
- 在趋势图中，以版本为单位，使用不同颜色的曲线，分别显示其错误（状态码）次数或错误（状态码）比例随时间变化的情况（图示中③）。

HTTP错误详情

HTTP错误列表显示4XX、5XX状态码的错误信息，如下图所示。

URL	状态码类型	错误数	操作
/api/16	500	4	查看详情
/api/36	500	4	查看详情
/api/65	400	3	查看详情
/api/126	400	3	查看详情
/api/126	300	3	-
/api/71	400	2	查看详情
/api/71	500	2	查看详情
/api/87	100	2	-
/api/34	400	2	查看详情
/api/84	200	2	-

单击**操作**列的**查看详情**，进入**HTTP错误列表**页面。

单击**操作**列的**查看详情**，进入**错误详情**页面，查看指定错误的详细信息。

5.4.4. 网络劫持

在**网络劫持**页面您可以按照指定条件，查询移动应用被劫持的次数和劫持率，以及两者随时间分布的趋势。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 网络劫持**。

条件设置说明

网络劫持页面的条件设置选项包括：

选项	说明
IP	查看被劫持域名解析的IP地址维度信息。
地域	查看被劫持域名的地域分布信息，地域以省份维度展示。
版本	用于指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	用于指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

选项	说明
风险级别	域名被解析到某IP地址时，疑似发生网络劫持的风险程度，包括高风险和低风险。 注意 • 请重点关注高风险级别的疑似网络劫持，避免发生网络劫持事件。抵御网络劫持，请使用HTTPDNS。 • 低风险发生网络劫持的概率比高风险低，存在由于运营商网络策略、网络使用代理等被别为疑似劫持的场景。

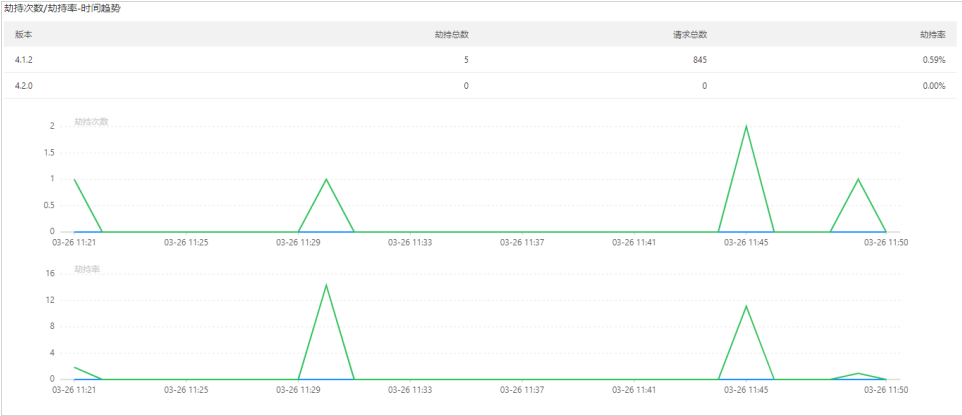
IP地址维度的分析数据

劫持次数

选项设置：

选项	操作说明
搜索域名	输入域名关键字，查询指定域名的劫持次数。
劫持次数	按照劫持次数的升序或降序显示查询结果。

时间趋势



参数	说明
版本	移动应用的版本号。
劫持总数	指定条件下的劫持总数。
请求总数	指定条件下的请求总数。
劫持率	劫持总数与请求总数的百分比。

被劫持域名下的IP地址列表

显示被劫持风险、劫持次数、劫持率、域名请求总数等。

参数	说明
IP	被劫持域名解析到的IP地址。
被劫持风险	域名被解析到某IP地址时，疑似发生网络劫持的风险程度，包括高风险和低风险。 **注意** • 请重点关注高风险级别的疑似网络劫持，避免发生网络劫持事件。抵御网络劫持，请使用HTTPDNS。 • 低风险发生网络劫持的概率比高风险低，存在由于运营商网络策略、网络使用代理等被别为疑似劫持的场景。
劫持次数	被劫持域名解析到某IP地址的总数。
劫持率	被劫持域名解析到某IP地址的总数与请求总数的占比。
域名请求总数	被劫持域名接收请求的总数。
操作	可在操作列豁免被劫持的IP地址。

豁免IP设置

- 豁免：对于被劫持的IP地址，若确认非DNS劫持，可在IP页签下对待豁免IP地址进行豁免操作。
  i. 在**IP**页签，单击待豁免IP地址**操作**列的**豁免**。
  ii. 单击**确定**。
- 取消豁免：可通过“豁免IP设置”调整豁免策略。
  i. 在**IP**页签，单击页面右上角的**豁免IP设置**。
  ii. 单击某域名前的**+**图标，展开豁免IP列表。
  iii. 单击IP地址**操作**列的**删除**。
  iv. 单击**确定**。

地域维度的分析数据

显示被劫持域名的地域分布情况。页面左侧的地域和域名列表与右侧地图呈对应关系，光标滑动至地图上指定区域，即可显示该区域指定域名的劫持总数。

> **说明** 单击 > 图标可展开域名列表。

5.4.5. 地域

在**地域**页面您可以按照指定条件，以移动设备的归属地为维度，查询网络状况，包括响应时间、网络错误和HTTP错误的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 地域**。

条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

地域列表

以地域为单位，显示网络的平均响应时间、网络错误和HTTP错误。

序号	选项	操作说明
1	搜索地域	输入关键字，查询指定地域的统计数据。
2	排序	按照平均响应时间、网络错误数或HTTP错误数的升序或降序显示统计数据。
3	切换	单击响应时间、网络错误或HTTP错误，下方列表以地域为统计单位，显示相应统计数据，并在右侧页面显示相应热力图。

热力图

显示各地域平均响应时间、网络错误数和HTTP错误数比较。颜色越深表示数值越大，颜色越浅表示数值越小。

5.4.6. 运营商

在**运营商**页面您可以按照指定条件，以移动设备的归属运营商为维度，查询网络状况，包括响应时间、网络错误和HTTP错误的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 运营商**。

条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

运营商列表

以运营商为单位，显示网络的平均响应时间、网络错误和HTTP错误。

序号	选项	操作说明
1	搜索运营商	输入关键字，查询指定运营商的统计数据。
2	排序	按照平均响应时间、网络错误数或HTTP错误数的升序或降序显示统计数据。
3	切换	单击响应时间、网络错误或HTTP错误，下方列表以运营商为统计单位，显示相应统计数据，并在右侧页面显示相应的时间趋势图和数据占比图。

时间趋势图

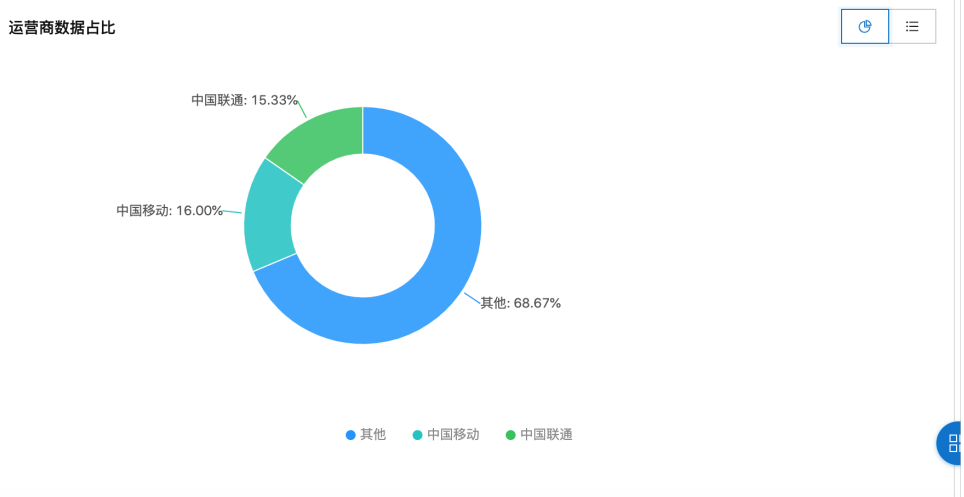
在运营商列表选择响应时间、网络错误或HTTP错误，显示相应的指标数据。

内容	说明
版本	以版本为单位，显示各项指标数据的基本信息。
平均响应时间	在运营商列表选择响应时间，则时间趋势图显示平均响应时间随时间变化的情况。
网络错误	在运营商列表选择网络错误，则时间趋势图显示网络错误数随时间变化的情况。
HTTP错误	在运营商列表选择HTTP错误，则时间趋势图显示HTTP错误数随时间变化的情况。

> **说明**
>
> - 将鼠标移动至趋势图上方，显示指定时间点的各项指标数据。
> - 在趋势图中，以版本为单位，使用不同颜色的曲线，分别各项指标数据随时间变化的情况。

运营商数据占比

在运营商列表选择响应时间、网络错误或HTTP错误，显示相应的指标数据。



单击右上角的饼图或列表图标，可采用饼图或列表形式查看数据占比。

5.4.7. 版本

在**版本**页面您可以照指定条件，以移动应用的版本为维度，查询网络状况，包括响应时间、网络错误和HTTP错误的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 版本**。

条件设置说明

选项	说明
时间	指定时间区间。
域名	指定域名类型。 **说明** 三方域名和自有域名的说明，请参见术语解释。

显示数据说明

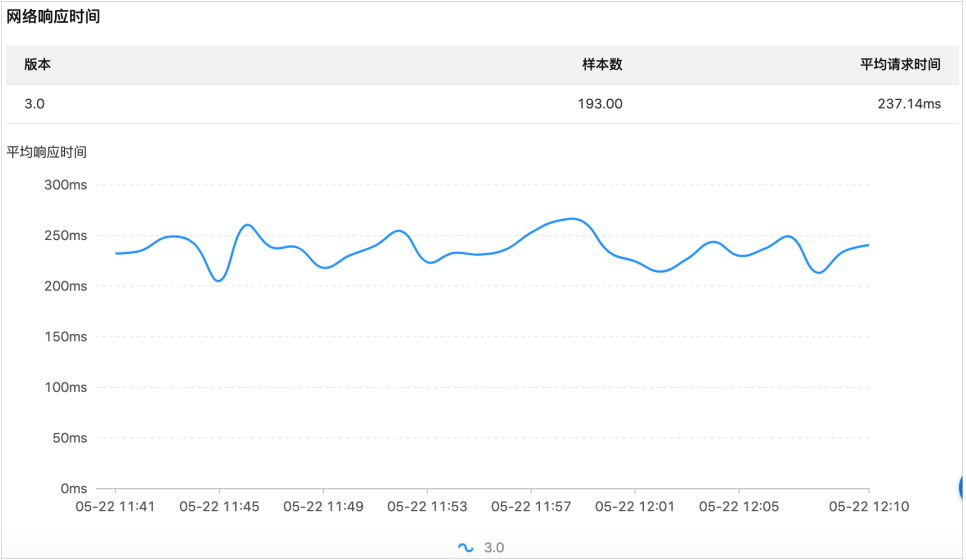
版本列表

以版本为单位，显示网络的平均响应时间、网络错误和HTTP错误。

序号	选项	操作说明
1	搜索版本	输入关键字，查询指定版本的统计数据。
2	排序	按照平均响应时间、网络错误数或HTTP错误数的升序或降序显示统计数据。
3	切换	单击响应时间、网络错误或HTTP错误，下方列表以版本为统计单位，显示相应统计数据，并在右侧页面显示相应的时间趋势图和数据占比图。
4	版本号	单击版本号，在右侧页面显示指定版本的时间趋势图和数据占比图。

时间趋势图

显示指定版本指定指标随时间变化的情况。



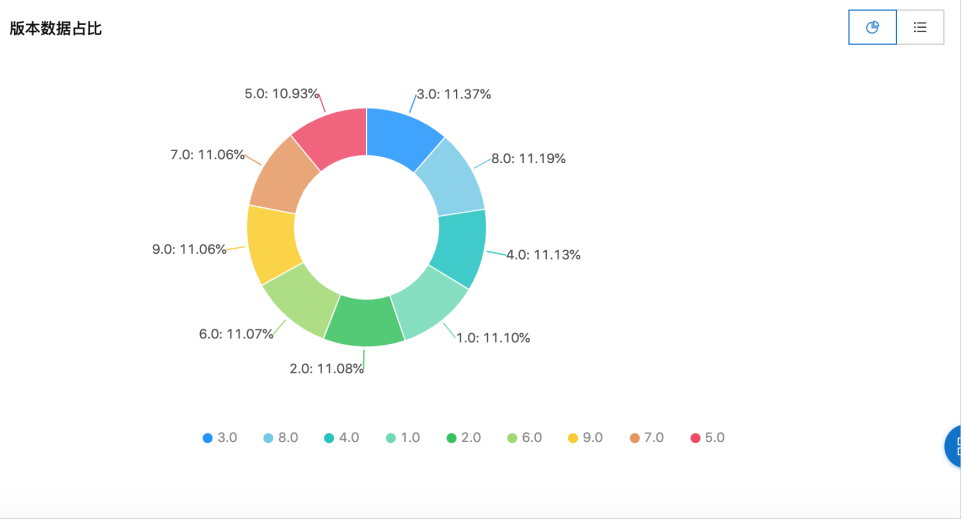
内容	说明
版本	以版本为单位，显示各项指标数据的基本信息。
平均响应时间	在版本列表选择响应时间，则时间趋势图显示平均响应时间随时间变化的情况。

内容	说明
网络错误	在版本列表选择网络错误，则时间趋势图显示网络错误数随时间变化的情况。
HTTP错误	在版本列表选择HTTP错误，则时间趋势图显示HTTP错误数随时间变化的情况。

> **说明** 将鼠标移动至趋势图上方，显示指定时间点的各项指标数据。

版本数据占比

显示各项指标数据的版本占比情况。



单击右上角的饼图或列表图标，可采用饼图或列表形式查看数据占比。

5.4.8. 设备

在**设备**页面您可以照指定条件，以移动设备的机型和操作系统为维度，查询网络状况，包括响应时间、网络错误和HTTP错误的情况。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**网络 > 设备**。

条件设置说明

选项	说明
版本	指定一个或多个移动应用的版本号。
时间	指定时间区间。

显示数据说明

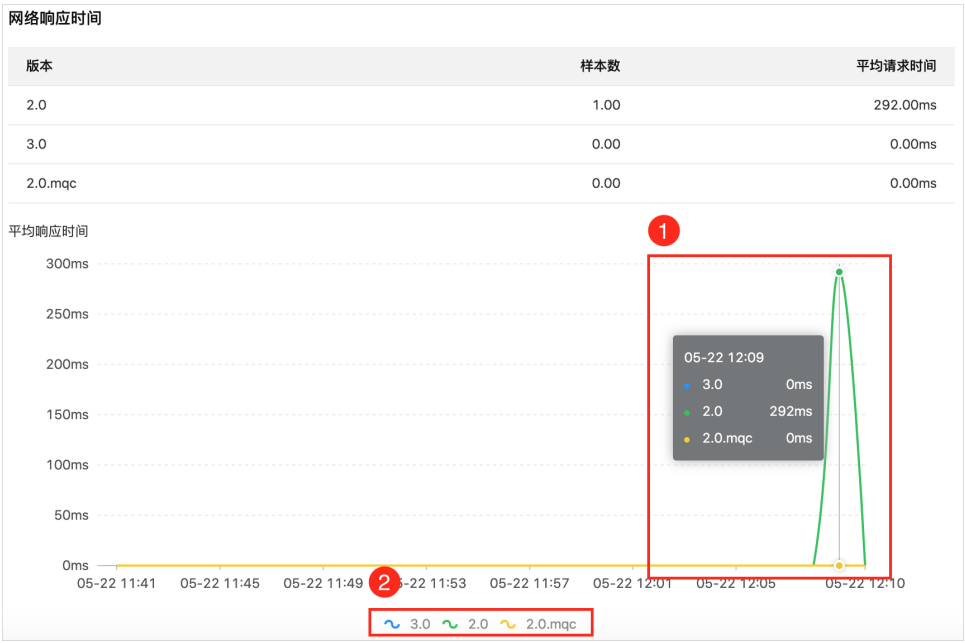
机型和操作系统列表

以移动设备的机型和操作系统为单位，显示网络的平均响应时间、网络错误和HTTP错误。

序号	选项	操作说明
1	切换页签	单击**机型**或**操作系统**，显示机型或操作系统列表。
2	搜索框	输入关键字，查询指定机型或操作系统的统计数据。
3	排序	按照平均响应时间、网络错误数或HTTP错误数的升序或降序显示统计数据。
4	切换	单击响应时间、网络错误或HTTP错误，下方列表以机型或操作系统为统计单位，显示相应统计数据，并在右侧页面显示相应的时间趋势图和数据占比图。
5	列表项	单击指定机型或操作系统，在右侧页面显示对应的时间趋势图。

时间趋势图

显示指定机型或操作系统的指定指标随时间变化的情况。



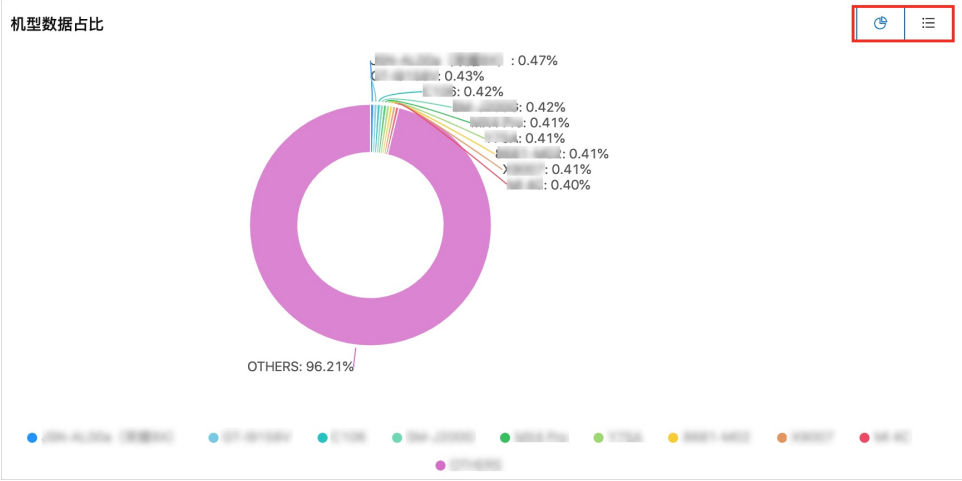
内容	说明
版本	以版本为单位，显示各项指标数据的基本信息。
平均响应时间	在左侧列表选择**响应时间**，则时间趋势图显示平均响应时间随时间变化的情况。
网络错误	在左侧列表选择网络错误，则时间趋势图显示网络错误数随时间变化的情况。
HTTP错误	在左侧列表选择HTTP错误，则时间趋势图显示HTTP错误数随时间变化的情况。

说明

- 将鼠标移动至趋势图上方，显示指定时间点的各项指标数据（图示中①）。
- 在趋势图中，以版本为单位，使用不同颜色的曲线，分别各项指标数据随时间变化的情况（图示中②）。

数据占比

显示各项指标数据的机型或操作系统占比情况。



单击右上角的饼图或列表图标，可采用饼图或列表形式查看数据占比。

5.5. 地域

在**地域**页面您可以按照指定条件，查询移动应用的平均启动时间、平均页面加载时间，以及设备分布和占比。

功能入口

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**地域 > 启动时间**或**地域 > 页面加载**。

条件设置说明

选项	说明
版本	指定一个移动应用的版本号。
时间	以天为单位，指定统计时间。
启动类型	指定移动应用的启动类型。 **说明** **页面加载**功能不涉及该选项。
机型	指定移动设备的机型。
域名	指定移动设备的归属地。

显示数据说明

地域列表

以地域为单位，显示平均启动时间和页面加载时间。

启动时间热力图

在**启动时间**页面，显示各地域平均启动时间对比。

颜色越深表示时间较长，颜色越浅表示时间较短。

参考时间：2秒内优秀；2~3秒良好；3~5秒可接受。

页面加载时间热力图

在**页面加载**页签，显示各地域平均页面加载时间对比。

颜色越深表示时间较长，颜色越浅表示时间较短。

参考时间：2秒内优秀；2~3秒良好；3~5秒可接受。

5.6. 设置URL过滤

通过设置白名单，终端SDK在上报网络数据时，对网络请求的URL/域名实施过滤。如网络请求的URL/域名已添加至白名单，则上报至性能分析服务进行后续分析和统计；否则忽略。

添加域名

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**数据设置 > URL设置**，然后在右侧页面单击**+添加白名单**。
3. 在**添加白名单**对话框输入需要添加为白名单的URL或域名，然后单击**确定**。

5.7. 设置域名及分组

性能分析服务支持对接入数据的域名进行分组管理，便于区分三方域名或自有域名，后续基于分组进行数据分析。

添加域名

1. 登录ARMS控制台，在左侧导航栏单击**App监控**，然后单击目标App右侧的**性能分析**。
2. 在左侧导航栏选择**数据设置 > 域名设置**，然后在右侧页面单击**+添加自有域名**。
3. 在**添加自有域名**面板输入域名，然后单击**确定**。

> **说明** 接入服务的用户数据中涉及的域名默认在列表中显示，分组为三方域名。

管理域名分组

域名分为自有域名和三方域名两类，后续基于不同分组进行数据分析。

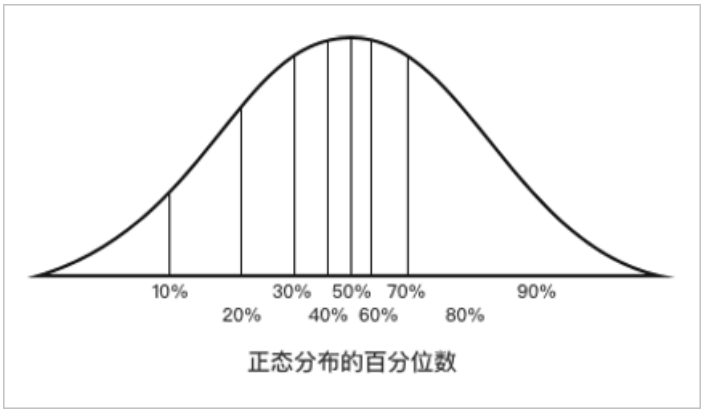
1. 在**自有域名**或**三方域名**列表中选中一个或多个域名。
2. 单击列表框之间的左右箭头，修改域名分组。

> **说明**
> - 选中域名列表表头中的复选框，可全选该分组下所有域名。
> - 在域名列表的搜索框中输入关键字，可模糊查询指定域名。

5.8. 术语解释

性能分析服务相关术语的解释。

术语	说明
冷启动	指启动一个全新的移动应用进程。

术语	说明
热启动	指进程仍在后台运行的移动应用，被重新唤醒。
机型	指设备的品牌、型号。
运营商	指设备所属的网络服务供应商。
地区	指设备的所属地。
启动耗时	指设备启动移动应用所需时间。
加载耗时	指设备加载页面所需时间。
滑动帧率	指单位时间内，设备加载页面的帧数。单位：帧/秒（fps）
分布统计	指关键指标数据在各个分段的分布情况。
时间趋势	指关键指标数据随时间变化的情况。
平均值	指对样本数据，取平均数值。
分位数	指将一个随机变量的概率分布范围分为几个等份的数值点，常用的有中位数（即二分位数）、四分位数、百分位数等。正态分布的百分位数情况如下图所示： 
90分位数	指将样本数据切分为100份，存在某阈值，其间数据占全部样本数据的90%。
中位数	指将样本数据按大小顺序排列起来，形成一个数列，居于数列中间位置的那个数据。
长尾分布	指启动时间大于8000ms的情况。
响应时间	指用户提出服务请求，至收到响应所需的时间。
网络错误	指使用网络库发起连接的过程中，各阶段可能出现的所有错误，包括但不限于DNS解析、SSL校验、建立连接、等待服务超时等。
错误率	指未得到正确响应的服务请求，占全部请求的比例。
自有域名	指用户方注册持有的域名。

术语	说明
三方域名	指非用户方注册持有的域名。
HTTP状态码	指示特定HTTP请求是否已成功完成。响应分为五类：信息响应（1XX）、成功响应（2XX）、重定向（3XX）、客户端错误（4XX）和服务器错误（5XX）。
网络劫持	指通过攻击域名解析服务器（DNS），或伪造域名解析服务器（DNS）的方法，把目标网站域名解析到错误的IP地址，从而实现用户无法访问目标网站的目的，或蓄意或恶意要求用户访问指定IP地址的目的。
劫持白名单	对于特定域名，解析在白名单之外的IP地址，将被视为劫持发生。

6.参考信息

6.1. 崩溃指标说明

崩溃分析，是将Android和iOS平台常见的App崩溃问题进行归类分析，帮助企业根据崩溃指标快速发现、定位问题。

Android崩溃指标

崩溃指标	数据定义	公式定义
总体Crash率	Crash基础指标，表示Native+Java类型的崩溃率。	今天实时Crash率=今天0点到当前时间Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。
Native Crash率	Crash基础指标，表示Native的崩溃率。	今天实时Native Crash率=今天0点到当前时间Native类型Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。
Native用户Crash率	Crash基础指标，表示Native的用户崩溃率。	今天实时Native用户Crash率 =今天0点到当前时间Native类型Crash影响的设备去重数量/今天累计UV。
Native Crash次数	Crash基础指标，表示Native的崩溃次数。	今天实时Native Crash次数=今天0点到当前时间Native类型Crash发生累计次数。
Java Crash率	Crash基础指标，表示Java的崩溃率。	今天实时Java Crash率=今天0点到当前时间Java类型Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。
Java用户Crash率	Crash基础指标，表示Java的用户崩溃率。	今天实时Java用户Crash率=今天0点到当前时间Java类型Crash影响的设备去重数量/今天累计UV。
Java Crash次数	Crash基础指标，表示Java的崩溃次数。	今天实时Java Crash次数=今天0点到当前时间Java类型Crash发生累计次数。

iOS崩溃指标

崩溃指标	数据定义	公式定义
总体Crash率	Crash基础指标，表示全堆栈+Abort类型的崩溃率。	今天实时Crash率=今天0点到当前时间Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。

崩溃指标	数据定义	公式定义
全堆栈Crash率	Crash基础指标，表示全堆栈的崩溃率。	今天实时全堆栈Crash率=今天0点到当前时间全堆栈类型Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。
全堆栈用户Crash率	Crash基础指标，表示全堆栈的用户崩溃率。	今天实时全堆栈用户Crash率=今天0点到当前时间全堆栈类型Crash影响的设备去重数量/今天累计UV。
全堆栈Crash次数	Crash基础指标，表示全堆栈的崩溃次数。	今天实时全堆栈Crash次数=今天0点到当前时间全堆栈类型Crash发生累计次数。
Abort异常未捕获Crash率	Crash基础指标，表示Abort的崩溃率。	今天实时Abort Crash率=今天0点到当前时间Abort类型Crash发生累计次数/今天0点到当前时间应用（版本）总启动次数。
Abort异常未捕获用户Crash率	Crash基础指标，表示Abort的用户崩溃率。	今天实时Abort用户Crash率=今天0点到当前时间Abort类型Crash影响的设备去重数量/今天累计UV。
Abort异常未捕获Crash次数	Crash基础指标，表示Abort的崩溃次数。	今天实时Abort Crash次数=今天0点到当前时间Abort类型Crash发生累计次数。

7.故障排除

7.1. 为什么RAM用户查看应用详情时出现错误？

Condition

RAM用户在ARMS控制台的App监控页面查看应用详情时，提示错误：**对不起，你没有权限执行操作**。

Cause

默认情况下，阿里云账号（主账号）拥有自己所创建的App资源的完整操作权限，但是新创建的RAM用户没有权限操作阿里云账号的资源。

Remedy

需要通过RAM授权的方式，给予RAM用户操作阿里云账号资源的权限。

操作步骤

1. 登录RAM控制台，在左侧导航栏中选择**权限管理 > 权限策略管理**。
2. 在**权限策略管理**页面，单击**创建权限策略**。
3. 在**新建自定义权限策略**页面，完成以下配置后，单击**确定**：
   i. 在**策略名称**文本框输入自定义的策略名称，例如：ARMSAppReadAccess。
   ii. 在**配置模式**区域选择**脚本配置**。
   iii. 在**策略内容**区域按需输入以下策略内容：
   - 策略内容：只读权限

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emasha:View*",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

   - 策略内容：管理权限

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emasha:*",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

4. 在左侧导航栏中选择**人员管理 > 用户**。

5.

6. 在**添加权限**面板中，单击**自定义策略**，在**选择权限**区域框中，通过关键字搜索需要添加的权限策略，并单击权限策略将其添加至右侧的**已选择**列表中，然后单击**确定**。
   添加权限面板提示**授权成功**。

8.App监控常见问题

本章节汇总了使用ARMS App监控的常见问题。

为什么无法创建App监控任务？

此情况一般是由于App监控任务的数量超过底层限制导致的，请联系我们的钉钉账号解决：arms160804。