阿里云
应用实时监控服务 ARMS

App 监控

文档版本：20200701

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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。

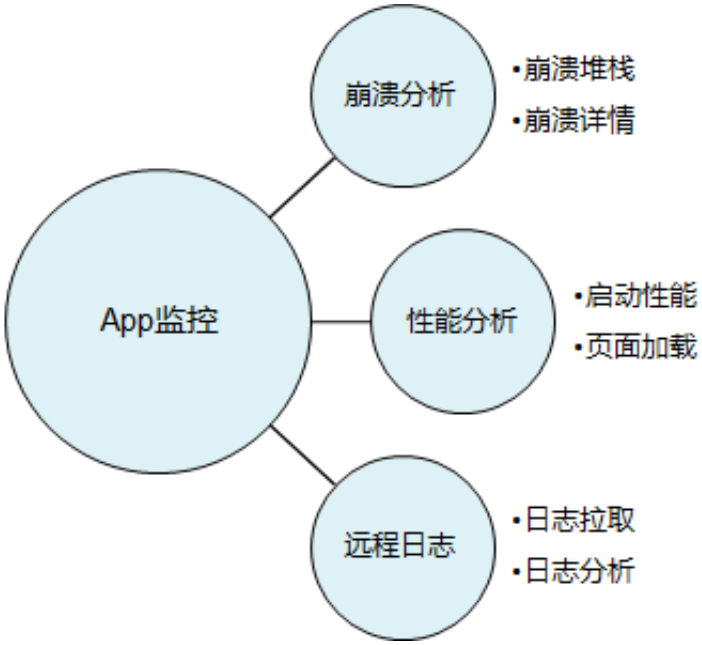
通用约定

格式	说明	样例
	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。	 **禁止：** 重置操作将丢失用户配置数据。
	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。
	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。
	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。	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

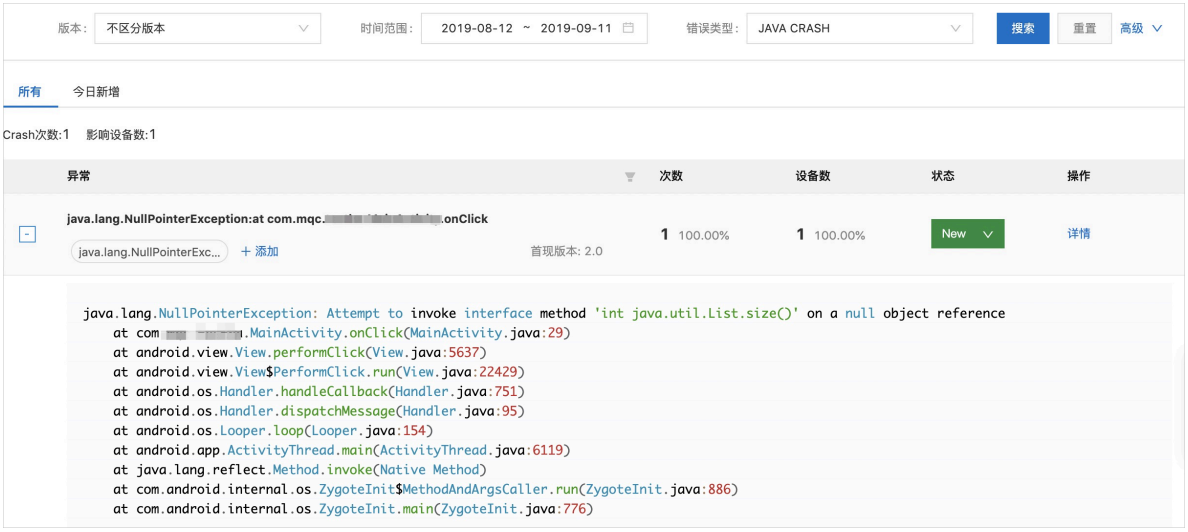
ARMS App监控是专注于监控移动设备上的应用性能和用户体验的工具，分别从崩溃分析、性能分析和远程日志这三个方面来帮助您精确衡量App应用的性能，并且能够实时监控、快速定位性能和可用性问题，帮助您以低成本、高效率发现App应用中的各类隐患。



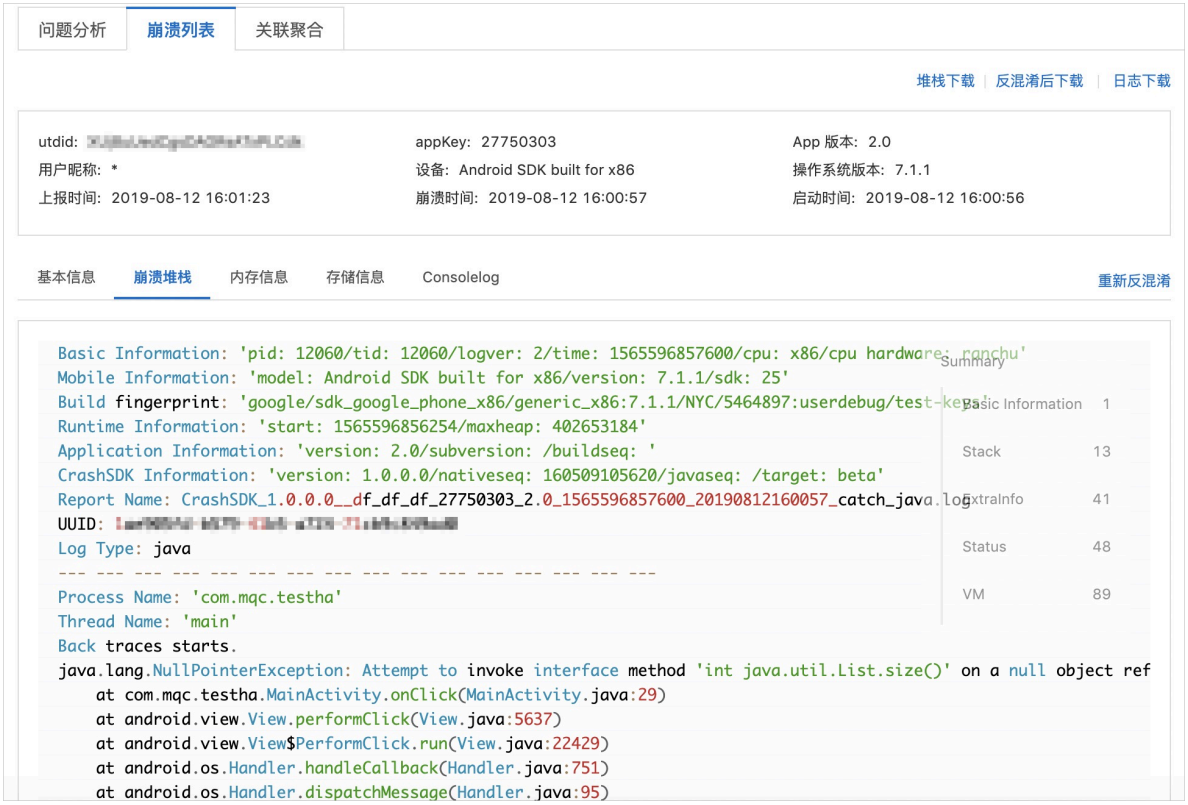
崩溃分析

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



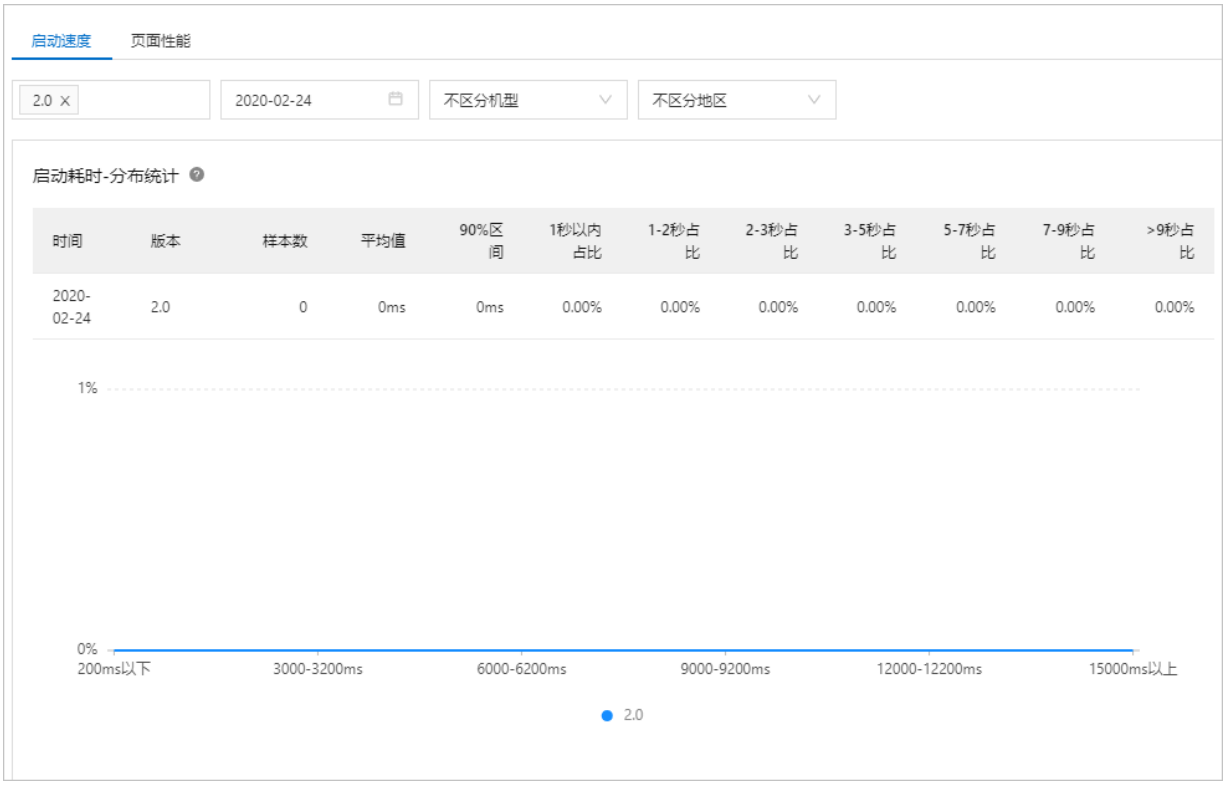
- 崩溃详情



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 创建监控任务

在您使用ARMS监控移动应用之前，需要先创建监控任务，将您的应用接入到ARMS。

操作步骤

1. 登录ARMS控制台，在左侧导航栏中选择**App监控**。
2. 在**App监控**页面，单击右上角的**创建App监控任务**。
3. 在弹出的**创建App监控任务**对话框的**填写App信息**页签中输入App名称和PackageName，并选择App的相应平台后，单击**创建应用**。

4. 在**查看配置**页签，按照说明进行操作，然后单击**下一步**。

   下图以Android平台为例。

5. 在**添加SDK**页签，按照说明进行操作，然后单击**完成**。

   下图以Android平台为例。

预期结果

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 崩溃分析

3.1.1 接入Android SDK进行崩溃分析

借助App监控的SDK，您可以获取完备的崩溃分析，具体包括Java Crash监控、Native Crash监控、ANR监控、JavaScript错误等。

前提条件

已在ARMS中创建App监控任务，请参见创建监控任务。

背景信息

本文档适用于使用gradle管理依赖的Android Studio项目，请参见Demo工程ha_android_demo。

步骤一：添加依赖

Maven仓库依赖接入：

1. 在项目build.gradle中添加阿里云Maven仓库地址。

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

2. app模块中的build.gradle的dependencies节点内添加以下代码。

```
compile('com.aliyun.ams:alicloud-android-ha-adapter:1.1.3.2-open@aar') {
    transitive=true
}
compile('com.aliyun.ams:alicloud-android-ha-crashreporter:1.2.1-open@aar') {
    transitive=true
}
```

步骤二：初始化实例

1. 在自定义Application类的**onCreate**里面启动服务。

```
public class MyApplication extends Application {
@Override
public void onCreate() {
   initHa();
}

private void initHa() {
  Log.e("ha", "init");
  //这里必须启动，否则服务端收不到数据。
   AliHaAdapter.getInstance().openPublishEmasHa();
```

```
    AliHaConfig config = new AliHaConfig();
    config.appKey = "<yourAppKeyId>";         //appKey
     config.appVersion = "x.xx";         //应用的版本号信箱。
     config.appSecret = "<yourAppKeySecret>";  //appSecret
     config.channel = "<yourChannelId>";        //应用的渠道号标记，自定义。
     config.userNick = null;
     config.application = this;
     config.context = getApplicationContext();
     config.isAliyunos = false;          //是否为yunos。
     AliHaAdapter.getInstance().startCrashReport(config);     //启动CrashReport。
   }
}
```

说明：

appKey和appSecret，可在创建监控任务中步骤4下载的配置文件中获得。

2. AndroidManifest.xml里面指定自定义Application。

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

步骤三：更新渠道标记

```
AliHaAdapter.getInstance().updateChannel("600000");
```

步骤四：更新自定义标记

```
AliHaAdapter.getInstance().updateUserNick("aliyun");
```

步骤五：添加混淆规则

```
-keep class com.alibaba.ha.**{*;}
-keep class com.taobao.tlog.**{*;}
-keep class com.ut.device.**{*;}
-keep class com.ta.utdid2.device.**{*;}
-keep public class com.alibaba.mtl.** { *;}
-keep public class com.ut.mini.** { *;}
-keep class com.alibaba.motu.crashreporter.**{ *;}
-keep class com.uc.crashsdk.**{*;}
-keep class com.ali.telescope.**{ *;}
-keep class libcore.io.**{*;}
-keep class android.app.**{*;}
-keep class dalvik.system.**{*;}
-keep class com.taobao.tao.log.**{*;}
-keep class com.taobao.android.tlog.**{*;}
-keep class com.alibaba.motu.**{*;}
-dontwarn com.taobao.orange.**
-dontwarn com.taobao.android.**
-dontwarn com.alibaba.ha.adapter.**
-dontwarn com.taobao.monitor.adapter.**
```

```
-dontwarn com.alibaba.fastjson.**
-dontwarn com.ali.alihadeviceevaluator.**
-dontwarn java.nio.file.**
-dontwarn org.codehaus.mojo.**
```

结果验证

在完成上述步骤之后，您可以对您的App进行测试，并登录ARMS控制台查看数据报表。

3.1.2 接入Android SDK进行性能分析

借助App监控的SDK，您可以获取完备的移动应用性能分析。

前提条件

已在ARMS中创建App监控任务，请参见创建监控任务。

背景信息

本文档适用于使用gradle管理依赖的Android Studio项目，请参见Demo工程ha_android_demo。

步骤一：添加依赖

Maven仓库依赖接入：

1. 在项目build.gradle中添加阿里云Maven仓库地址。

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

2. app模块中的build.gradle的dependencies节点内添加以下代码。

```
compile('com.aliyun.ams:alicloud-android-ha-adapter:1.1.3.2-open@aar') {
    transitive=true
}
compile('com.aliyun.ams:alicloud-android-apm:1.0.7.7-open@aar') {
    transitive=true
}
```

步骤二：接入服务

1. 在自定义Application类的**onCreate**里面启动服务。

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        initHa();
    }
    private void initHa() {
        Log.e("ha", "init");
        AliHaConfig config = new AliHaConfig();
        config.appKey = "xxxxxxxx";         //appKey
        config.appVersion = "x.xx";         //应用的版本号信息。
        config.appSecret = "xxxxxxxxxxxx";  //appSecret
        config.channel = "mqc_test";        //应用的渠道号标记，自定义。
        config.userNick = null;
        config.application = this;
```

```
        config.context = getApplicationContext();
        config.isAliyunos = false;          //是否为yunos。
        AliHaAdapter.getInstance().addPlugin(Plugin.apm);
        AliHaAdapter.getInstance().start(config);
    }
}
```

说明：

appKey和appSecret，可在创建监控任务中，步骤4下载的配置文件中获得。

2. AndroidManifest.xml里面指定自定义Application。

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

结果验证

在完成上述步骤之后，您可以对您的App进行测试，并登录ARMS控制台查看数据报表。

3.1.3 接入Android SDK获取远程日志

借助App监控的SDK，您可以获取完备的远程日志。

前提条件

已在ARMS中创建App监控任务。请参见创建监控任务。

背景信息

本文档适用于使用gradle管理依赖的Android Studio项目，请参见demo工程ha_android_demo。

说明：

日志在移动端最多存储7天。

步骤一：添加依赖

Maven仓库依赖接入。

1. 在项目build.gradle中添加阿里云Maven仓库地址。

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

2. App模块中的build.gradle的dependencies节点内添加以下代码。

```
compile('com.aliyun.ams:alicloud-android-ha-adapter:1.1.3.2-open@aar') {
    transitive=true
```

```
    }
compile('com.aliyun.ams:alicloud-android-ha-tlog:1.1.2.2-open@aar') {
        transitive=true
    }
```

步骤二：接入服务

1. 在自定义Application类的**onCreate**里面启动服务。

```
    public class MyApplication extends Application {
        @Override
        public void onCreate() {
            initHa();
        }
        private void initHa() {
            Log.e("ha", "init");
            AliHaConfig config = new AliHaConfig();
            config.appKey = "xxxxxxxx";         //appkey
            config.appVersion = "x.xx";         //应用的版本号信箱。
            config.appSecret = "xxxxxxxxxxxx";  //appsecret
            config.channel = "mqc_test";        //应用的渠道号标记，自定义。
            config.userNick = null;
            config.application = this;
            config.context = getApplicationContext();
            config.isAliyunos = false;          //是否为yunos。
            config.rsaPublicKey = "xxxxxxx";    //tlog公钥，在控制台下载aliyun-emas-services.json文件，文件内的appmonitor.tlog.rsaSecret字段即为公钥信息（文件下载方式：在EMAS控制台-> 应用管理 找到对应的应用，点击应用所在区块右上角菜单内的“配置下载”），必填。
            AliHaAdapter.getInstance().addPlugin(Plugin.tlog);
            AliHaAdapter.getInstance().openDebug(true);
            AliHaAdapter.getInstance().start(config);
        }
    }
```

说明：

appKey和appSecret，可在创建监控任务中，步骤4下载的配置文件中获得。

2. AndroidManifest.xml里面指定自定义Application。

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

3. App模块的build.gradle的defaultConfig节点内添加以下代码，根据需要配置abi：

```
ndk {
        abiFilters 'armeabi-v7a'
```

```
    }
```

步骤三：获取远程日志

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

示例代码：

```
String TAG = "xxx";
String MODEL = "xxxxx";

TLogService.logv(MODEL,TAG,"test tlog1");
TLogService.logd(MODEL,TAG,"test tlog2");
TLogService.logi(MODEL,TAG,"test tlog3");
TLogService.logw(MODEL,TAG,"test tlog4");
TLogService.loge(MODEL,TAG,"test tlog5");
```

结果验证

在完成上述步骤之后，您可以对您的App进行测试，并登录ARMS控制台查看数据报表。

3.2 开始监控iOS APP

3.2.1 接入iOS SDK进行性能分析

借助App监控的SDK，您可以获取完备的移动应用性能分析。

前提条件

已将应用接入App监控。请参见创建监控任务。

背景信息

本文档适用于使用cocoaPods管理依赖的Xcode项目，以及支持iOS 8.0或以上版本的App。

步骤一：添加依赖

Pod依赖接入：

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', '~> 1.0.0'
```

~>为模糊指定版本号方式，~> 1.0.0表明版本位于1.0.0 <= version < 1.1.0之间的最新版本SDK，SDK版本请参见Podfile Syntax Reference。

3. 执行pod update。

> **说明：**
> 如果在Xcode 9上出现报错：**RuntimeError - [Xcodeproj] Unknown object version.**，请将Project Format改成Xcode 8.0-compatible。

步骤二：接入服务

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>
```

示例代码：

```
    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";       //昵称，自定义，会上报

    [[AlicloudAPMProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick
:nick];
    [AlicloudHAProvider start];
```

步骤三：编译

如果编译报错，请在项目的build setting里设置Allow Non-modular Includes In Framework Modules为YES。

如果出现包含duplicate symbol的错误，请确认其他本地依赖和CocoaPods管理的依赖是否有重复。如有重复，请删除本地依赖。强烈建议所有依赖都通过CocoaPods管理。

结果验证

在完成上述步骤之后，您可以对您的APP进行测试，并登录ARMS控制台查看数据报表。

3.2.2 接入iOS SDK进行崩溃分析

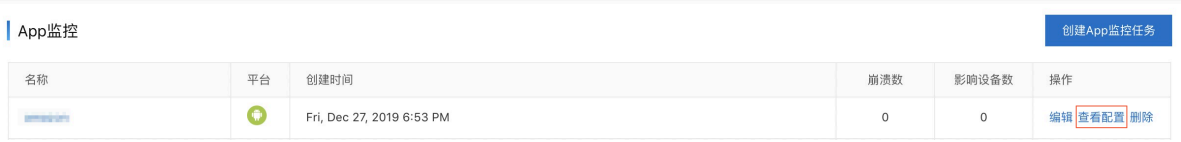
借助App监控的SDK，您可以获取完备的移动端崩溃分析监控，具体包括Crash监控、Abort监控、崩溃问题聚合、影响面分析等。

前提条件

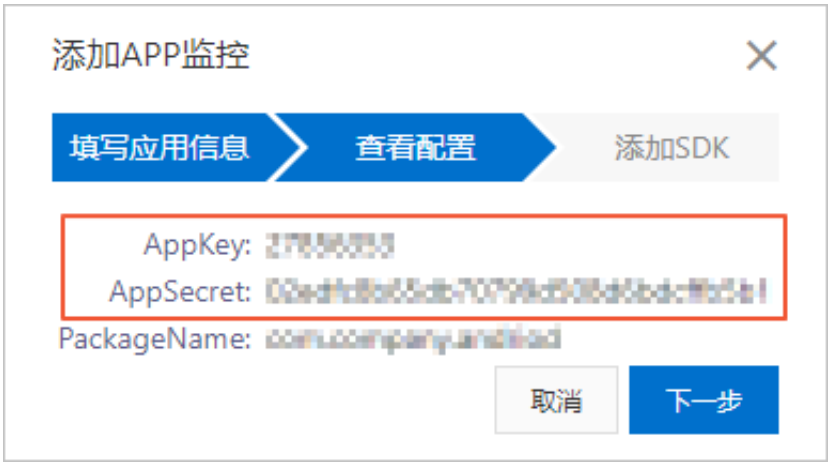
- 已将应用接入App监控。请参见创建监控任务。

- 获取应用的AppKey和AppSecret。

  在**App监控**页面中单击目标应用所在行最右侧的**查看配置**。



  在弹出的对话框中复制AppKey和AppSecret。



背景信息

本文档适用于使用cocoaPods管理依赖的Xcode项目，以及支持iOS 8.0或以上版本的App。

步骤一：添加依赖

您可以选择Pod依赖接入和本地依赖接入两种方式：

- Pod依赖接入

  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 'AlicloudCrash' , '~> 1.1.0'
```

  ~>为模糊指定版本号方式，~> 1.1.0表明引用位于1.1.0 <= version < 1.2.0之间的最新版本SDK，请参见Podfile Syntax Reference。

  3. 执行pod update。

  说明：

如果在Xcode 9上出现报错：**RuntimeError - [Xcodeproj] Unknown object version.**，请将Project Format改成Xcode 8.0-compatible。

- 本地依赖接入

1. 单击**App监控**页面右上角的**SDK下载**，下载**崩溃分析**、**iOS版**对应的SDK，并复制下载包内所有库文件放在项目工程中。
2. 引入Framework。

   a. 在Xcode中，把下载的SDK目录中的framework拖入对应Target下，在弹出的对话框中勾选**Copy items if needed**。framework如下：

   - AlicloudCrash.framework
   - AliHACore.framework
   - AliHALogEngine.framework
   - AliHAProtocol.framework
   - AlicloudHAUtil.framework
   - AlicloudUtils.framework
   - AlicloudUT.framework
   - CrashReporter.framework
   - JDYThreadTrace.framework
   - TBCrashReporter.framework
   - TBJSONModel.framework
   - TBRest.framework
   - UTDID.framework
   - ZipArchive.framework

   b. 在**Build Phases** > **Link Binary With Libraries**中，添加以下公共包：

   - libc++.tbd
   - SystemConfiguration.framework

步骤二：接入服务

在ARMS控制台下载AliyunEmasServices-Info.plist并复制至项目根目录。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>
```

示例代码：

```
    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";       //昵称，自定义，会上报

    [[AlicloudCrashProvider alloc] autoInitWithAppVersion:appVersion channel:channel
nick:nick];
    [AlicloudHAProvider start];
```

步骤三：编译

如果编译报错，请在项目的build setting里设置Allow Non-modular Includes In Framework Modules为YES。

如果出现包含duplicate symbol的错误，请确认其他本地依赖和CocoaPods管理的依赖是否有重复。如有重复，请删除本地依赖。强烈建议所有依赖都通过CocoaPods管理。

结果验证

在完成上述步骤之后，您可以对您的App进行测试，并登录ARMS控制台查看数据报表。

3.2.3 接入iOS SDK获取远程日志

借助App监控的SDK，您可以获取完备的远程日志。

前提条件

已将应用接入App监控。请参见创建监控任务。

背景信息

本文档适用于使用cocoaPods管理依赖的Xcode项目，以及支持iOS 8.0或以上版本的App。

说明：
日志在移动端最多存储7天。

步骤一：添加依赖

Pod依赖接入：

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'
```

~>为模糊指定版本号方式，~> 1.0.0表明引用位于1.0.0 <= version < 1.1.0之间的最新版本SDK。

SDK版本请参见Podfile Syntax Reference

3. 执行pod update。

> **说明：**
> 如果在Xcode 9上出现报错：**RuntimeError - [Xcodeproj] Unknown object version.**，请将Project Format改成Xcode 8.0-compatible。

步骤二：接入服务

在AppDelegate.m文件的application:didFinishLaunchingWithOptions:方法中初始化SDK。引入头文件：

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

示例代码：

```
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";
//昵称，自定义，会上报

[[AlicloudTlogProvider alloc] autoInitWithAppVersion:appVersion channel:channel nick:
nick];
[AlicloudHAProvider start];
```

步骤三：编译

如果编译报错，请在项目的build setting里设置Allow Non-modular Includes In Framework Modules为YES。

如果出现包含duplicate symbol的错误，请确认其他本地依赖和CocoaPods管理的依赖是否有重复。如有重复，请删除本地依赖。强烈建议所有依赖都通过CocoaPods管理。

步骤四获取远程日志

引入头文件：

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

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"];
```

结果验证

在完成上述步骤之后，您可以对您的App应用进行测试，并登录ARMS控制台查看数据报表。

4 控制台功能

4.1 崩溃分析

本文介绍ARMS App监控的崩溃分析页面所包含的功能。您可以借助控制台对崩溃数据的分类统计和展示，快速发现并定位问题。

前提条件

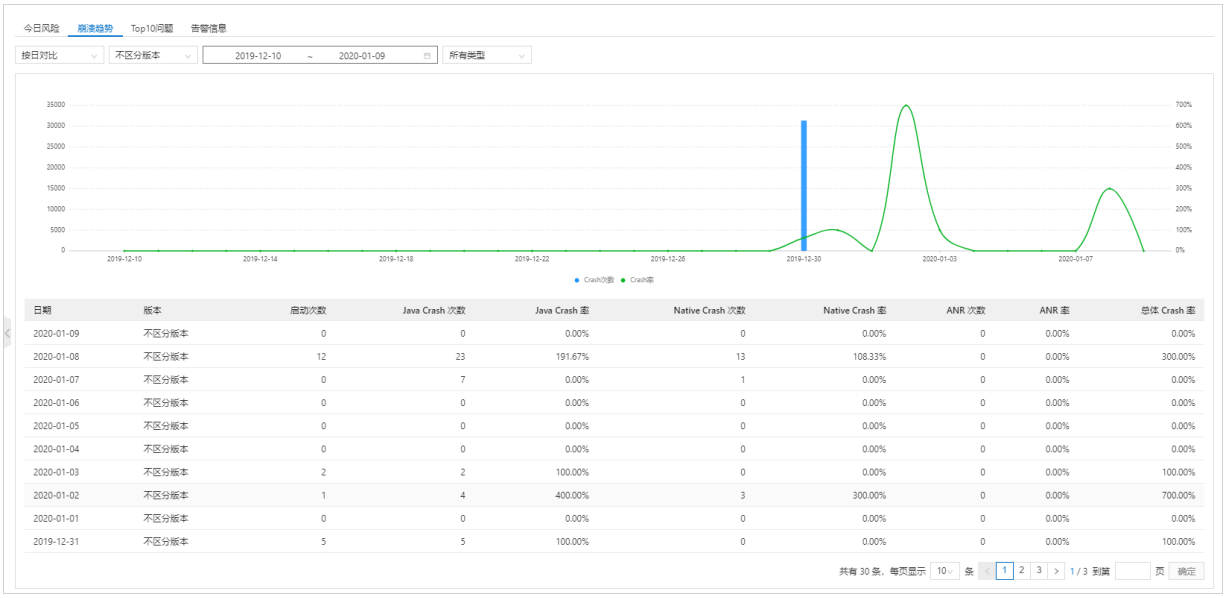
创建监控任务

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的名称。
4. 在接下来的页面的导航栏中可以看到**崩溃分析**、**性能分析**、**远程日志**。单击可选择每个模块中的子功能。

概览

概览页面是对崩溃信息进行概览性展示和分析。



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
2020-01-09	不区分版本	0	0	0.00%	0	0.00%	0	0.00%	0.00%
2020-01-08	不区分版本	12	23	191.67%	13	108.33%	0	0.00%	300.00%
2020-01-07	不区分版本	0	7	0.00%	1	0.00%	0	0.00%	0.00%
2020-01-06	不区分版本	0	0	0.00%	0	0.00%	0	0.00%	0.00%
2020-01-05	不区分版本	0	0	0.00%	0	0.00%	0	0.00%	0.00%
2020-01-04	不区分版本	0	0	0.00%	0	0.00%	0	0.00%	0.00%
2020-01-03	不区分版本	2	2	100.00%	0	0.00%	0	0.00%	100.00%
2020-01-02	不区分版本	1	4	400.00%	3	300.00%	0	0.00%	700.00%
2020-01-01	不区分版本	0	0	0.00%	0	0.00%	0	0.00%	0.00%
2019-12-31	不区分版本	5	5	100.00%	0	0.00%	0	0.00%	100.00%

概览包括今日风险、崩溃趋势、Top10问题和告警信息：

- **今日风险**：包含4个统计维度，分别为今日总新增异常、占比超过50% 的异常、占比较昨天上涨20% 的异常、占比较昨天上涨20% 的指标。
  - 以列表形式展示各统计维度的具体数据，包括版本、异常类型、类型条数、影响用户百分比、影响设备数，以及可执行的操作。
- **崩溃趋势**：以折线图、柱状图的形式，为用户展示以下4个维度的数据。
  - **今日实时**：按时间段展示崩溃率的变化趋势。

    可通过折线图上方的下拉列表筛选数据，筛选维度包括App版本、崩溃指标、时间区间。
  - **按日对比**：选定期间内，每日的崩溃次数（柱状图）和比率（折线图）。

    可通过柱状图或折线图上方的下拉列表筛选数据，筛选维度包括App版本、Crash类型、日期区间。

    以列表形式列出每日的各项崩溃指标。
  - **按版本对比**：选定日期内，App不同版本的崩溃次数（柱状图）和比率（折线图）。

    可通过柱状图或折线图上方的下拉列表筛选数据，筛选维度包括App版本、Crash类型、日期。

    以列表形式列出选定日期内各版本的各项崩溃指标。
  - **按地域对比**：App部署的各地域的崩溃次数（柱状图）和比率（折线图）。

    可通过柱状图或折线图上方的下拉列表筛选数据，筛选维度包括App版本、Crash类型、日期。

    以列表形式列出选定日期内各地域的各项崩溃指标。
- **Top10问题**：列出各崩溃类型在选定日内，崩溃次数最多的前10项崩溃信息，以及相对前一天的变化。

  单击目标Crash信息可进入详情页，查看问题分析、崩溃列表、关联聚合：
  - **问题分析**

    以折线图形式展示不同日期的崩溃次数对比。

    折线图下方列出调用栈代码，以及各项特征的数据，可切换列表或饼图展示。
  - **崩溃列表**

    列出崩溃的基本信息、崩溃堆栈、内存信息、存储信息和Consolelog。
  - **关联聚合**

    列出关联异常、异常状态和可执行的操作。

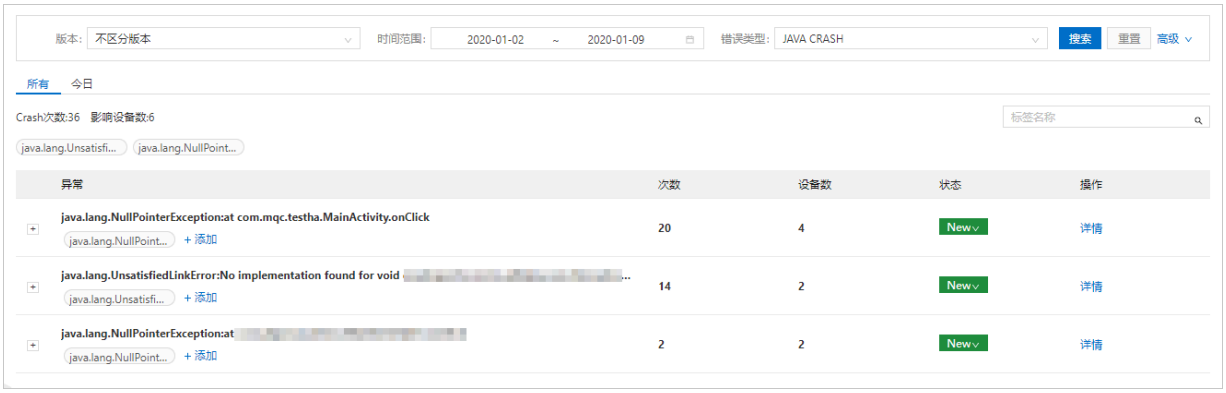
- **告警信息**

  列出选定时间段内的告警信息，包括触发时间、App版本、告警信息、规则名称、订阅人、告警方式，以及可执行的操作。

  可通过列表上方的下拉列表筛选数据，筛选维度包括规则类型、规则名称和日期区间。

聚合分析

以列表的形式展示选定时间段内的所有异常的详情及当日发生的异常的详情。



可通过列表上方的下拉列表筛选数据，筛选维度包括App版本、日期区间和错误类型。单击目标异常操作列的**详情**，展示该条异常的崩溃分析、崩溃列表和关联聚合：

- **问题分析**

  以折线图展示该条异常在不同日期内发生次数的对比。

  折线图下方为调用栈分析和特征分析。

- **崩溃列表**

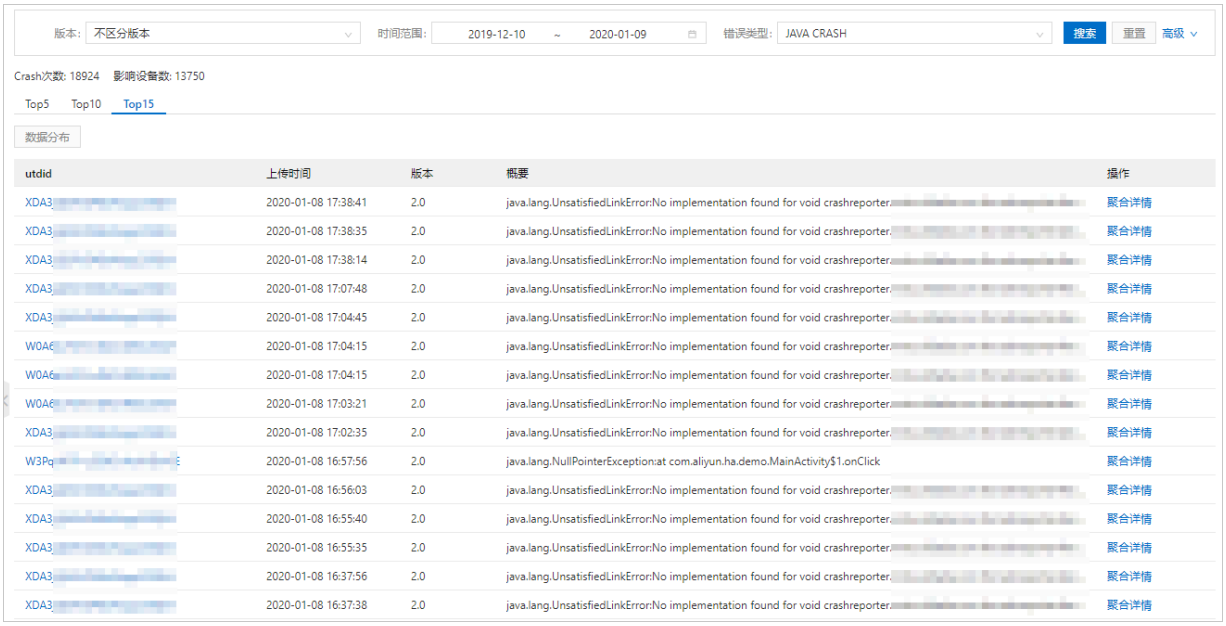
  展示异常的基本信息、崩溃堆栈、内存信息、存储信息和Consolelog。

- **关联聚合**

  列出关联异常、异常状态和可执行的操作。

定位分析

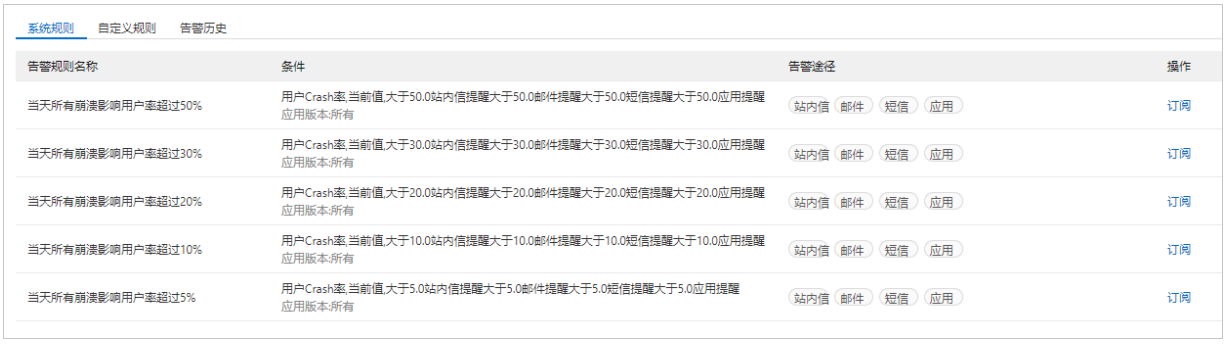
以列表形式，展示选定时间段内的发生异常次数最多的Top5、Top10、Top15设备的utdid（服务端生成的设备唯一标识符）、异常上报时间、App版本、异常概要，以及可执行的操作。



可通过列表上方的下拉列表筛选数据，筛选维度包括App版本、日期区间、错误类型。单击utdid或其对应**操作**列的聚合详情，页面右侧会弹出utdid对应的异常的聚合分析页。

报警设置

当规则被触发时，系统会以您指定的报警方式向报警联系人分组发送报警信息，以提醒您采取必要的问题解决措施。



报警设置包括系统规则、自定义规则和告警历史：

- **系统规则**：以列表展示系统规则名称、告警条件、告警途径、订阅操作。您可以自定义告警途径。
- **自定义规则**：可新增、编辑、订阅自定义规则。
- **告警历史**：以列表展示告警的详情，及可执行的操作。 可筛选规则类型、规则名称和时间范围。

还原配置

由于移动App的代码使用了混淆机制，导致难以从异常发生的日志中直接进行堆栈跟踪。您可以在**Mapping.txt文件**（对应Java代码）和**So文件**（对应Native代码）中提供混淆前后的内容对照

表，在**上传配置**中上传文件，App监控会自动使用文件将混淆后的崩溃堆栈追踪信息还原成正常信息。

4.2 性能分析

ARMS的App应用监控中的性能分析功能可以对移动App应用的性能进行统计与分析。

前提条件

创建监控任务

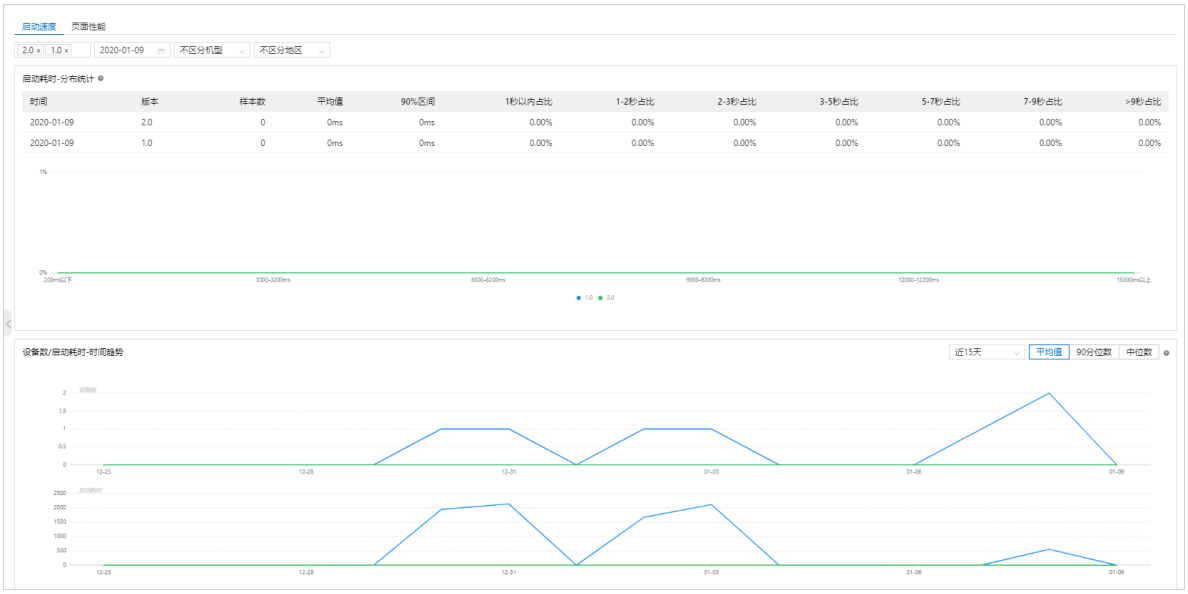
功能入口

1. 登录ARMS控制台。
2. 在左侧导航栏中单击**App应用监控**。
3. 在**App应用监控**页面，单击目标App应用的名称。
4. 在接下来的页面的导航栏中可以看到**崩溃分析**、**性能分析**、**远程日志**。单击可选择每个模块中的子功能。

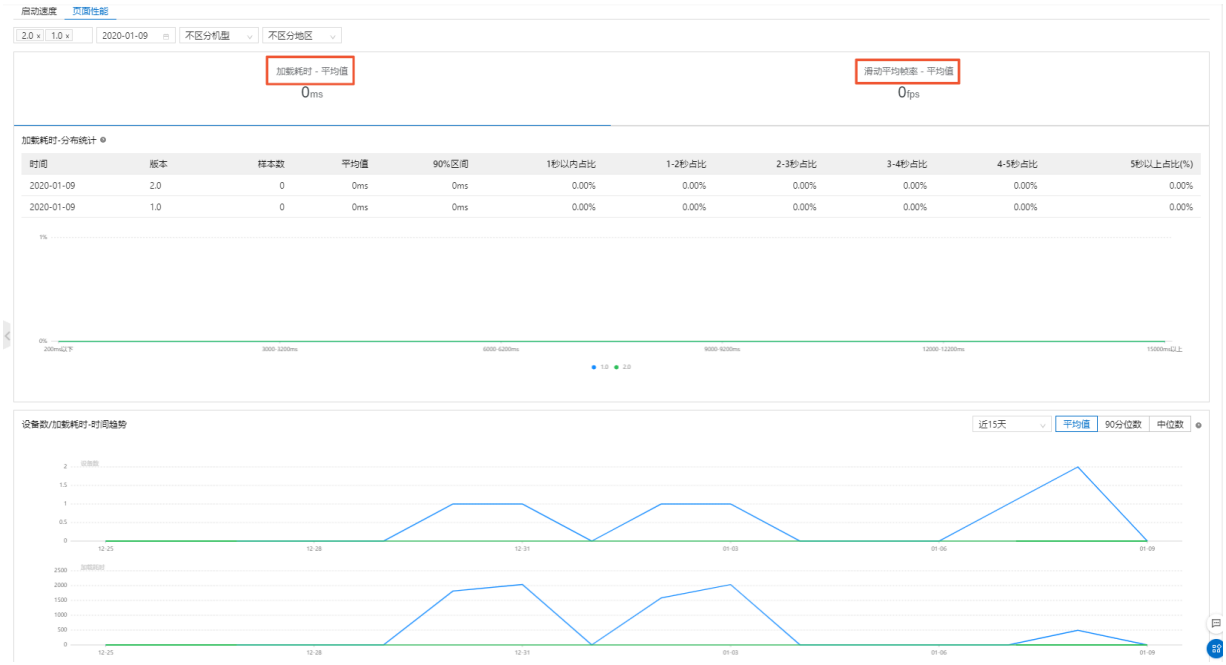
概览

对移动App应用的启动速度与页面性能进行概览性的展示和分析：

- **启动速度**
  - 统计各终端上App应用启动耗时的分布情况，并展示为表格。
  - 分析终端数与App应用启动耗时的时间趋势，并展示为图表。
  - 可以通过**App应用版本**、**时间**、**设备**与**地区**对呈现的数据进行筛选。



- **页面性能**
  - 单击**加载耗时 - 平均值**页签，可以统计各终端上App应用页面加载耗时的分布情况，并以表格形式展示。
  - 单击**滑动平均帧率 - 平均值**页签，可以统计各终端上App应用页面滑动平均帧率的分布情况，并以表格形式展示。
  - 可以通过**App应用版本**、**时间**、**设备**与**地区**对呈现的数据进行筛选。

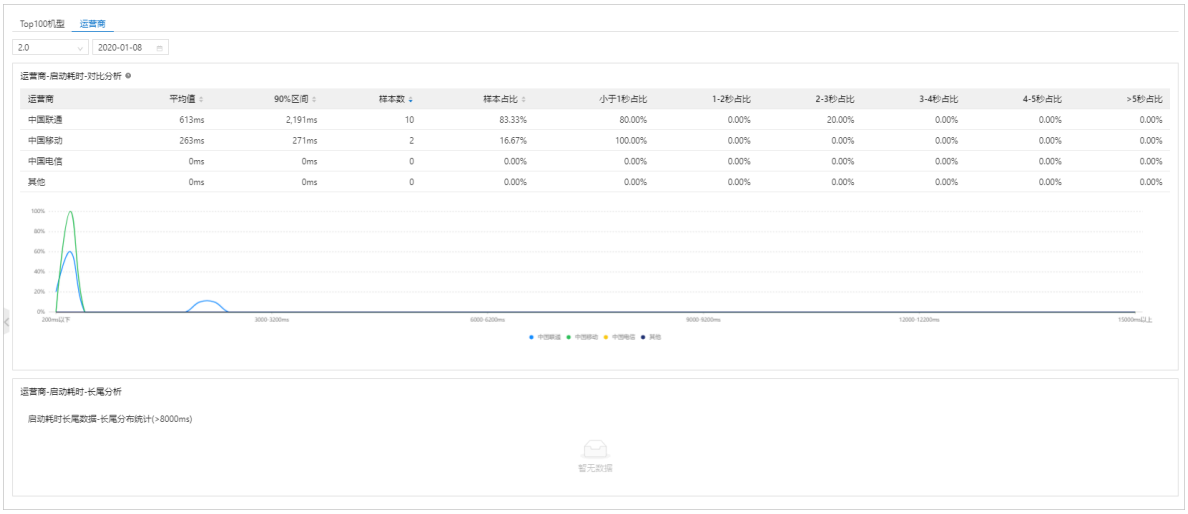


启动

对移动App应用的启动速度进行详细的分析：

- **Top 100机型**
  - 展示100个不同机型上的App应用启动性能统计数据，可以切换多种维度的排序，可用于分析机型对App应用启动速度的影响。
  - 可以通过**App应用版本**、**时间**对数据进行筛选。

- **运营商**
  - 展示不同运营商环境下App应用启动性能的统计数据，可用于分析运营商对App应用启动速度的影响。
  - 对出现耗时过长（超过8000ms）情况的运营商进行长尾分析。
  - 可以通过**App应用版本**、**时间**对数据进行筛选。



页面

对移动App应用的页面性能进行详细的分析：

- 在**加载耗时**页签中，可以统计App应用中每个页面加载耗时的分布情况，并展示为表格。

  单击目标页面名称，进入该App应用页面的启动分析页面。
- 在**滑动帧率**页签中，可以统计App应用中每个页面滑动帧率的分布情况，并展示为表格。

  单击目标页面名称，进入该App应用页面的启动分析页面。
- 可以通过**App应用版本**、**时间**、**设备**与**地区**对呈现的数据进行筛选。

地域

- 在**启动时间**页签，可以统计各个地区的App应用启动耗时情况，并展示为图表。
- 在**页面加载**页签，可以统计各个地区的App应用页面加载耗时情况，并展示为图表。
- 可以通过**App应用版本**、**时间**、**设备**与**地区**对呈现的数据进行筛选。

4.3 远程日志

ARMS的App监控模块中的远程日志功能可以拉取移动终端的异常日志，并进行统计与分析。

远程日志主要针对App上线后的问题分析和定位：

- 全量拉取移动终端的异常日志，还原现场，快速定位复杂问题。
- 单独客户异常问题，快速拉取日志进行排查。

前提条件

创建监控任务

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的名称。
4. 在接下来的页面的导航栏中可以看到**崩溃分析**、**性能分析**、**远程日志**。单击可选择每个模块中的子功能。

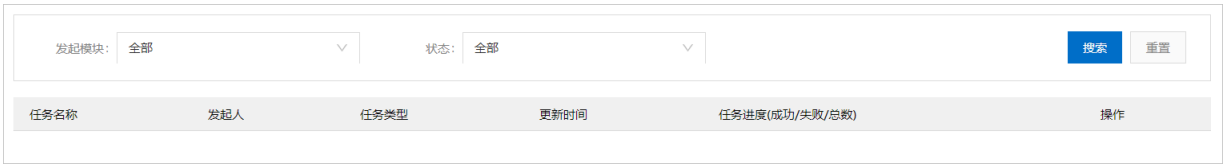
概览

各终端发生异常的概览，以列表形式展示。可以通过App版本、时间、设备与地区等对呈现的数据进行筛选，还可以拉取所需异常日志。

任务管理

展示拉取日志的任务的列表。可以通过**发起模块**和任务**状态**对呈现的数据进行筛选：

- **发起模块**：分为**用户拉取**和**崩溃分析**。**用户拉取**是指用户通过控制台操作，主动拉取的日志；**崩溃分析**是指崩溃分析模块发起的日志拉取。
- **状态**：分为**正常任务**和**已终止任务**，即成功拉取日志的任务和拉取失败的任务。



拉取设备列表

展示从各终端拉取日志的列表。可以通过设备唯一标识、发起人、用户更新时间和状态对呈现的数据进行筛选。

5 参考信息

5.1 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。
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。

6 故障排除

6.1 为什么RAM用户查看应用详情时出现错误？

问题现象

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**。

可能原因

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。

解决方案

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。

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**：

   a) 在**策略名称**文本框输入自定义的策略名称，例如：ARMSAppReadAccess。

   b) 在**配置模式**区域选择**脚本配置**。

   c) 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. 在**用户**页面上找到需要授权的用户，单击**操作**列中的**添加权限**。
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**。

7 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。