

业务实时监控服务 ARMS

常见问题

常见问题

一般常见问题

使用常见问题

ARMS 免费版和付费版有何区别？

关于 ARMS 各产品免费版和付费版的区别，请参见 ARMS 产品功能列表。

ARMS 应用监控目前支持哪些语言？将来的计划是什么样的？

ARMS 应用监控目前支持 Java 语言，计划支持 PHP 和 C# 语言，且即将提供 OpenAPI 供其他语言开发者定制接入。

ARMS 前端监控是否支持除杭州和新加坡以外的其他地域？

目前，前端监控在中国大陆范围内只支持杭州地域，在国际范围内只支持新加坡地域。由于前端监控的统计数据来自公网，地域化的需求很小，所以暂无支持除杭州和新加坡以外地域的计划。在其他地域开通阿里云服务的客户，可直接使用上述两个区域的前端监控服务，性能和功能不会受到影响。

ARMS 自定义监控如何扩容？

如果通过界面提示发现 ARMS 自定义监控由于计算资源不足而需要扩容，请联系 ARMS 在线客服进行手动扩容。理论上，扩容仅需数分钟即可完成。

ARMS 自定义监控的实时计算引擎和列存储能直接暴露给用户使用吗？

不能。用户仅可通过 ARMS 提供的用户界面和 OpenAPI 的方式来定制监控数据和获取结果。主要原因是 ARMS 的实时计算和列存储在组装上进行过高度定制优化，所以并不适合直接暴露某个单独模块给用户使用。如果用户有实时计算和列存储的特别需求，请开通其他阿里云产品服务，例如流计

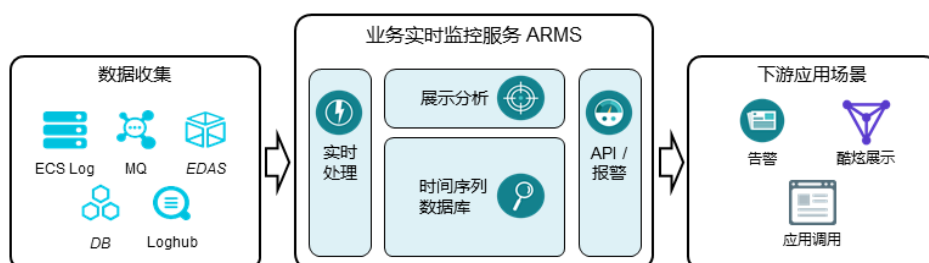
算、表格存储等。

云产品对接

本文阐述了 ARMS 在云产品的对接问题。

ARMS 产品对接总览

ARMS 在阿里云上可以和各产品进行对接，如下图所示。



其中：

- ECS Log、MQ、Loghub 等产品作为数据源可与 ARMS 无缝对接，详情可查看 ARMS。
- 其他下游产品，例如 DataV，其对接方式比较特殊，详述如下。

DataV 对接

前提条件：

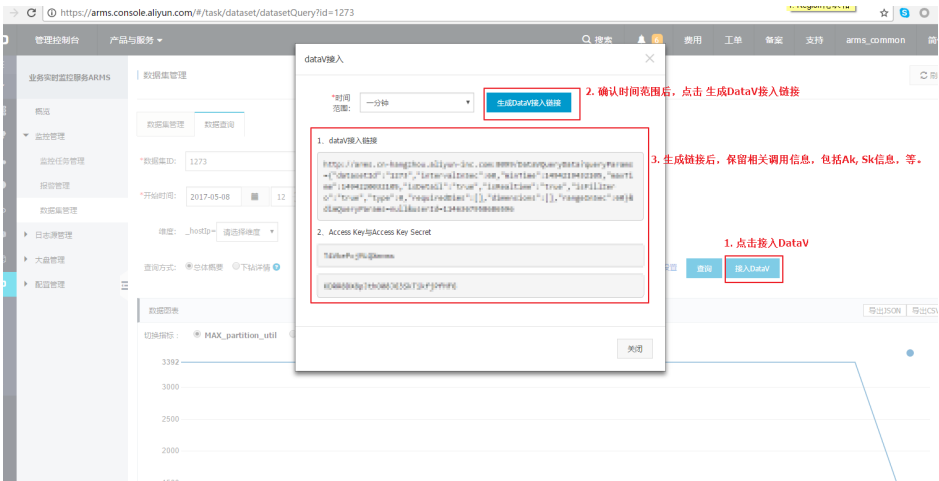
- 开通 ARMS
- 开通 DataV 基础版

接入步骤：

确定要进行查询的数据

在 ARMS 控制台的左侧菜单栏中，选择**自定义监控** > **数据集管理**。选择要查询的数据集，补全相关参数，并单击**查询数据**按钮。

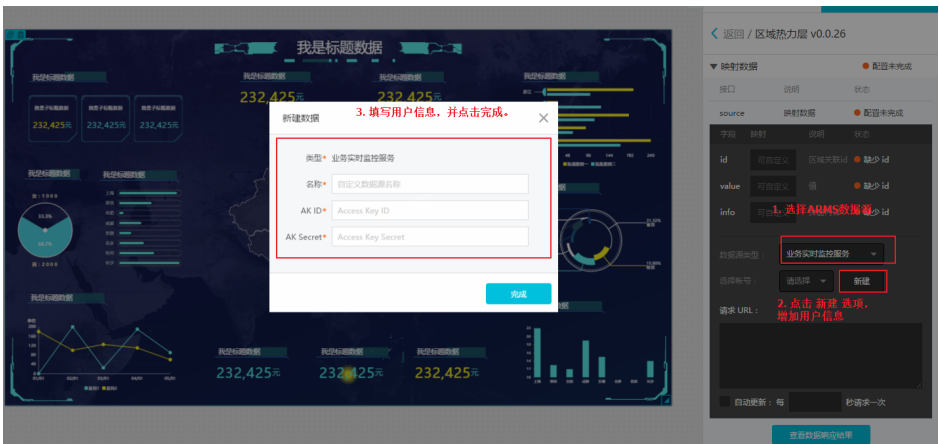
验证查询结果。确认无误后，单击“接入 DataV”按钮，并确认生成信息。如下图所示：



在 DataV 中选择 ARMS 数据源进行展示

在 DataV 中某个具体要展示的数据层, 选择 ARMS 数据源。

如果第一次使用, 则需要在界面中创建账号。如下图所示, 单击**新建账号**, 将第一步中在 ARMS 上生成的 Ak、Sk 填入, 并填写一个方便识别的用户名称。



创建用户以后, 将在第一步中 ARMS 生成函数 URL 填入到 DataV 中, 单击验证, 并根据结果填写字段映射表。



至此，ARMS 接入完成。

常见问题

问：我在 ARMS 中下钻维度采用的是省市区的中文名字，如四川省、上海市。在 API 输出时，API 调用采用下钻详情输出，也输出了所有省市的正确结果。但是如何在 DataV 的热力图中进行展示？

答：DataV 的地图热力图在地区 ID 方面仅支持区域码。如果要在 DataV 中进行展示，需要将省市转换成区域码。可以打开省市地区码映射，将其内容拷贝下来，在 ARMS 中建立一个维度的映射表，并在 ARMS 结果输出时，将结果关联到映射表做转义。详情请参考映射表管理。

RAM 子账号访问控制台

ARMS 支持阿里云主子账号（RAM）访问体系，本文介绍如何授权子账号访问 ARMS。

前提条件

使用 RAM 子账号访问 ARMS 控制台，子账号需要满足以下两个条件：

- 需要为该子账号创建 AccessKeyId/AccessSecretKey。
- 必须启用该子账号的控制台登录功能，为该子账号设置登录用户名、密码。

操作方法请参考 RAM 文档。

授权子账号访问 ARMS 服务控制台

满足登录前提条件的子账号，还需要授权才能使用 ARMS 服务控制台。目前 ARMS 仅支持授予子账号 ARMS 服务资源完全访问权限。

注意：Open API 暂时不支持子账号访问。

操作步骤如下：

1. 登录 RAM 控制台，在子账号管理界面，选择要授权子账号。操作方法参考 RAM 文档给用户授权。
2. 在授权策略页面选择 AliyunARMSFullAccess。

授权完成后，即可用子账号登录 ARMS 控制台。

应用监控常见问题

应用监控接入常见问题

本文介绍了导致用户创建应用时无法接入 ARMS 应用监控的常见问题及其排查方法。

检查网络连通性

使用 Telnet 命令测试目标主机与 ARMS 服务器网络是否连通。

各地域 ARMS 应用监控服务器地址：

地域	地址
杭州	arms-dc-hz.aliyuncs.com
北京	arms-dc-bj.aliyuncs.com
上海	arms-dc-sh.aliyuncs.com
青岛	arms-dc-qd.aliyuncs.com
深圳	arms-dc-sz.aliyuncs.com

假设用户在 ARMS 的深圳地域创建应用，则测试深圳地域环境与 ARMS 服务器网络是否连通。

以下结果表明网络已连通：

```
telnet arms-dc-sz.aliyuncs.com 8443

Trying 119.23.169.12...
Connected to arms-dc-sz.aliyuncs.com.
Escape character is '^J'.
```

注意：必须根据地域替换服务器地址，端口不变。

如何检查 ArmsAgent 是否加载成功？

使用 ps 命令查看命令行启动参数中是否成功加载 ArmsAgent。

```
ps -ef | grep 'arms-bootstrap'
```

成功加载时，如下图所示：

```
➜ ~ ps -ef | grep arms-agent
501 39784 18536 0 10:48下午 ?? 0:28.29 /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java
-Djava.util.logging.config.file=/Users/yangpeng/Library/Caches/IntelliJ IDEA2017.1/tomcat/Unnamed_apm-demo/conf/logging.properties
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:
56081,suspend=y,server=n -server -Xms512M -Xmx2048M -XX:PermSize=128m -XX:MaxPermSize=256m -Xmn50m -Dspas.identity=/Users/y
anpeng/key.txt -javaagent:/Users/yanpeng/debug/arms-apm-agent/target/arms-agent-1.7.0-SNAPSHOT/arms-bootstrap-1.7.0-SNAPSHOT.
jar -Darms.licenseKey=aokc50 -Darms.appId=aokc50 -Darms.appId=aokc50 -Darms.appId=aokc50 -Dcom.sun.management.jmxremote=-Dc
om.sun.management.jmxremote.port=1077 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=f
```

其命令行中的 `arms.licenseKey` 及 `arms.appId` 属性必须与 ARMS 应用设置界面中显示的内容保持一致。



ArmsAgent 加载成功，但是界面仍无监控数据？

1. 确认您的应用是否有持续的外部请求访问，包括 HTTP 请求、HSF 请求和 Dubbo 请求。
2. 确认选择的查询时间范围是否正确。请您将查询时间条件设为最近 15 分钟，然后再次确认是否有监控数据。

如果的通过 `-jar` 命令行启动的，请检查命令行设置，确保 `-javaagent` 参数在 `-jar` 之前。

```
java -javaagent:{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=xxx
-Darms.appId=xxx -jar demoApp.jar
```

如果仍无监控数据，请打包 Java 探针日志（路径：ArmsAgent/log），并联系钉钉服务账号 @ARMS-服务解决问题。

检查 JDK 版本。如果 JDK 版本为 1.8.0_25或者1.8.0_31，可能会出现无法安装探针的情况，建议您升级对应的 JDK 版本，或联系钉钉服务账号 @ARMS-服务咨询。

自定义监控常见问题

关于自定义清洗的常见问题

本文回答了关于自定义清洗的一些常见问题。

如何将特殊字符（不可见字符）作为切分的分隔符？

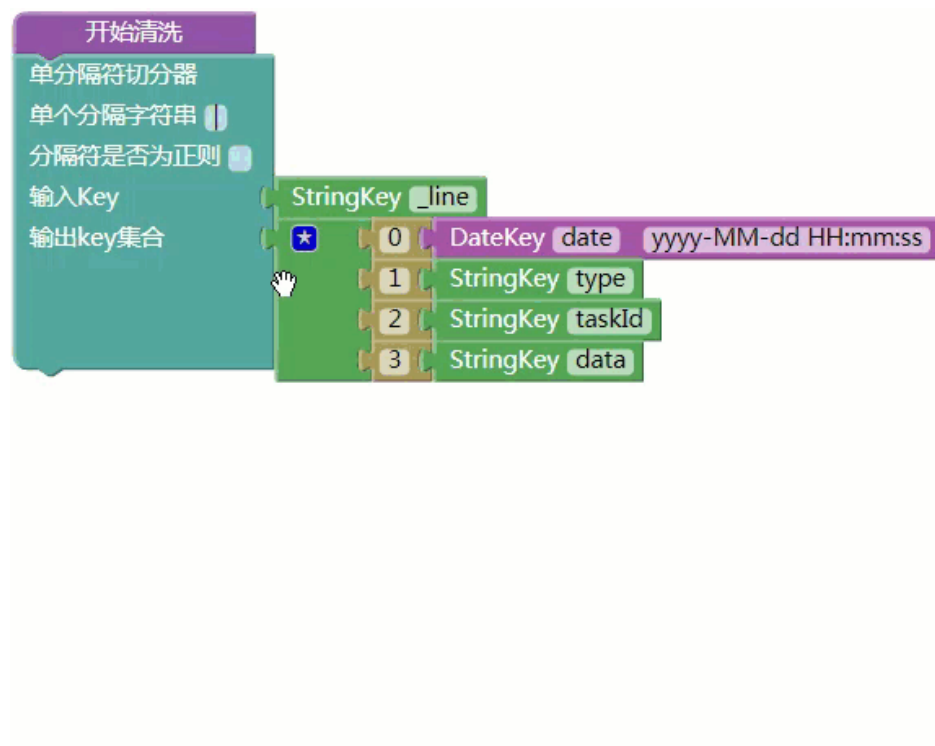
回答：如果不可见字符是 ASCII 码的“1”，则对应的分隔符为 \u0001。如果是“2”，则为 \u0002.....依此类推。如果要用“tab”作为分隔符，请使用 \t。

如何测试流程配置页面上的特殊字符？

回答：因为无法在浏览器里粘贴不可见字符，所以您只能暂时将测试文本的分隔符替换为可见字符进行测试。全部流程测试通过后，再将分隔符替换为不可见字符进行部署。

如何在积木块的 Key 列表中添加更多 Key？

回答：请单击积木块上的蓝底白色五角星，并拖入更多的项目。



JSON 格式的日志可以切分吗？

回答：可以，请使用 Json 切分器。详情请参考内置切分器的“Json 切分器”部分。

支持跨行日志（如异常堆栈）吗？

回答：暂不支持。

任务运行错误处理

TLogParseException 类型转换异常

异常原因：通常用户刚开始使用 ARMS 的时候，会使用 ARMS 的智能切分功能对样本数据进行切分。对于样本数据中是数字的字段，智能切分器会将其置为 LongKey，而实际数据过来时，发现数据有部分是字符串型的，这个时候就会报类型转换异常。

解决方案：

1. 重新进入监控任务编辑页面。
2. 在第 2 步**日志清洗**页面，选择**自定义切分**。
3. 根据错误提示，调整相应的类型转换异常字段。比如 String 无法转为 Long 的状况下，可以点开 Keys，找到 StringKey 替换 LongKey 即可。

FlowControlException 维度限流异常

异常原因：数据集大维度状况下，计算存储资源消耗会非常严重。在有限资源的状况下，ARMS 为了保证大部分监控任务能正常运行，对数据集维度个数进行了一定的限制，当前维度个数限制为 1000 个。当数据集的维度大小超出 1000 个时，ARMS 就会提示维度限流异常。

解决方案：

1. 根据错误提示中提到的数据集 ID，在**监控管理**>**数据集管理**页面搜索确定出现异常的数据集。
2. 查看维度的设置，看是否是自己真正需要观测的维度。如果不是自己需要关注的维度，请调整数据集，将维度配置的地方设置为**无**。
3. 回到监控任务页面，点击**暂停**按钮，再点击**恢复**按钮，新的配置即可生效。

如果确定配置的维度是自己需要关注的维度，请在 ARMS 控制台首页点击**联系我们**，与我们的客服人员联系，申请 VIP。我们会为您开通独立的计算存储资源，解决维度限流异常。

ExpressionRuntimeException 表达式异常

异常原因：通常是由于实际的数据与配置的切分器出现了不匹配。

解决方案：

1. 暂停监控任务，进入监控任务编辑页面。
2. 进入第 2 步**日志清洗**页面。
3. 将出现异常的日志，贴入日志抓取结果下的文本框。
4. 逐步拉开降低切分器的复杂度，点击**日志切分预览**，查看切分是否正常，最终找到不符合预期的部分。
5. 通过调整切分器或者调整输出数据解决该异常。

JSONException JSON 切分异常

异常原因：通常出现这个异常的原因是出现了异常的 JSON 数据，导致 JSON 切分器无法正常工作。

解决方案：根据错误提示，调整自己的日志输出为符合切分规则的 JSON。

前端监控常见问题

前端监控常见问题

本文解答了关于前端监控的一些常见问题。

1. 为什么有些监控页面或 API 名称中出现了星号（*）？

ARMS 前端监控的页面统计是以实际打开的页面 URL 名称为基础进行各维度统计的。监控页面或 API 名称中的星号（*）并不是真实页面 URL 的一部分，而是表示这是经过 URL 收敛后的结果。换言之，它表示这不是一个具体的 URL，而是一些相似 URL 的集合。



URL 收敛算法说明

- 问题：“变量”会导致同类 URL 发散成多个，因而难以监控和分析。
- 目标：合并同类 URL，用星号（*）代替不断变化的“变量”。
- 方案：采用我们自研的 URL 收敛算法，在尽可能保留语义信息的前提下，合并同类 URL，减少 URL 总数。主要分为以下两步：
 - 聚类：将相似 URL 归纳为一组。
 - “变量”识别：提取同组 URL 中不断变化的“变量”，并以星号（*）代替。

收敛过程如下图所示：



2. 为什么页面访问量列表和页面访问速度列表不一致？

这是因为您的应用是单页面应用（即 Single-Page Application，SPA），且开启了 SPA 自动解析。在 SPA 的应用场景下，页面访问量和页面访问速度的统计方法如下：

- 页面访问量：触发 hashchange 事件后会自动上报 PV，以统计该 hash 值对应页面的 PV 情况。所以在查看 SPA 应用的页面访问量列表时，可以查看对应的 hash 页面的具体 PV。
- 页面访问速度：因为 SPA 应用切换 hash 值后，页面的访问速度不会变化，所以访问速度的统计不以 hash 值为维度，这样不仅可以减少不必要的上报量，而且仍然可以清晰了解页面的性能情况。

Script Error 的产生原因和解决方法

“Script error” 是一个常见错误，但由于该错误不提供完整的报错信息（错误堆栈），问题排查往往无从下手。本文分析了该错误的产生原因，并提出了可行的解决方法。

“Script error” 的产生原因

“Script error” 有时也被称为跨域错误。当网站请求并执行一个托管在第三方域名下的脚本时，就可能遇到“Script error”。最常见的情形是使用 CDN 托管 JS 资源。

为了更好地理解，假设以下 HTML 页面部署在 `http://test.com` 域名下：

```
<!doctype html>
<html>
<head>
<title>Test page in http://test.com</title>
</head>
<body>
<script src="http://another-domain.com/app.js"></script>
<script>
window.onerror = function (message, url, line, column, error) {
console.log(message, url, line, column, error);
}
foo(); // 调用app.js中定义的foo方法
</script>
</body>
</html>
```

假设 `foo` 方法调用了一个未定义的 `bar` 方法：

```
// another-domain.com/app.js
function foo() {
bar(); // ReferenceError: bar is not a function
}
```

页面运行之后，捕获到的异常信息如下：

```
"Script error.", "", 0, 0, undefined
```

其实这并不是一个 JavaScript Bug。出于安全考虑，浏览器会刻意隐藏其他域的 JS 文件抛出的具体错误信息，这样做可以有效避免敏感信息无意中被不受控制的第三方脚本捕获。因此，浏览器只允许同域下的脚本捕获具体错误信息，而其他脚本只知道发生了一个错误，但无法获知错误的具体内容。

请参考 Webkit 源码：

```
bool ScriptExecutionContext::sanitizeScriptError(String& errorMessage, int& lineNumber, String& sourceURL)
{
    KURL targetURL = completeURL(sourceURL);
    if (securityOrigin()->canRequest(targetURL))
        return false;
    errorMessage = "Script error.";
    sourceURL = String();
    lineNumber = 0;
    return true;
}
```

了解了“Script error”的产生原因之后，接下来看看如何解决这个问题。

解法 1：开启 CORS (Cross Origin Resource Sharing , 跨域资源共享)

为了跨域捕获 JavaScript 异常，可执行以下两个步骤：

添加 `crossorigin="anonymous"` 属性。

```
<script src="http://another-domain.com/app.js" crossorigin="anonymous"></script>
```

此步骤的作用是告知浏览器以匿名方式获取目标脚本。这意味着请求脚本时不会向服务端发送潜在的用户身份信息（例如 Cookies、HTTP 证书等）。

添加跨域 HTTP 响应头。

```
Access-Control-Allow-Origin: *
```

或者

```
Access-Control-Allow-Origin: http://test.com
```

注意：大部分主流 CDN 默认添加了 `Access-Control-Allow-Origin` 属性。以下是阿里 CDN 的示例：

```
$ curl --head https://retcode.alicdn.com/retcode/bl.js | grep -i "access-control-allow-origin"

=> access-control-allow-origin: *
```

完成上述两步之后，即可通过 `window.onerror` 捕获跨域脚本的报错信息。回到之前的案例，页面重新运行后，捕获到的结果是：

```
=> "ReferenceError: bar is not defined", "http://another-domain.com/app.js", 2, 1, [Object Error]
```

解法 2（可选）：try catch

难以在 HTTP 请求响应头中添加跨域属性时，还可以考虑 try catch 这个备选方案。

在之前的示例 HTML 页面中加入 try catch：

```
<!doctype html>
<html>
<head>
<title>Test page in http://test.com</title>
</head>
<body>
<script src="http://another-domain.com/app.js"></script>
<script>
window.onerror = function (message, url, line, column, error) {
console.log(message, url, line, column, error);
}
try {
foo(); // 调用app.js中定义的foo方法
} catch (e) {
console.log(e);
throw e;
}
</script>
</body>
</html>
```

再次运行，输出结果如下：

```
=> ReferenceError: bar is not defined
at foo (http://another-domain.com/app.js:2:3)
at http://test.com/:15:3

=> "Script error.", "", 0, 0, undefined
```

可见 try catch 中的 Console 语句输出了完整的信息，但 window.onerror 中只能捕获 “Script error”。根据这个特点，可以在 catch 语句中手动上报捕获的异常。

```
// 参考本文末尾的相关文档 “前端监控 API 使用指南”
__bl.error(error, pos);
```

注意：尽管 try catch 方法可以捕获部分异常，但推荐采用解法 1。

相关文档

- 前端监控 API 使用指南