阿里云
物联网数据分析服务

云端开发指南

文档版本：20190916

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
	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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 云端SDK参考

1.1 下载云端SDK

物联网平台云端SDK用于调用云端API，以实现物联网平台的云端能力，如产品管理、设备管理、Topic管理、数据流转规则管理、消息通信等。

> 说明:
> 本章节仅介绍云端SDK的使用。设备端SDK开发，请参见设备端SDK。

云端SDK下载地址

物联网平台提供的云端SDK语言版本有：Java、Python、PHP和.NET。

单击以下链接，进入相应的云端SDK源码下载地址。

- IoT Java SDK
- IoT Python SDK
- IoT PHP SDK
- IoT .NET SDK

下载云端SDK Demo

阿里云物联网平台提供云端SDK使用Demo。Demo中包含Java、Python、PHP、.NET版本SDK。

单击这里下载云端SDK Demo。

SDK使用说明

云端SDK使用帮助说明，请参见以下链接文档。

- #unique_6
- #unique_7
- #unique_8
- #unique_9

1.2 Java SDK使用说明

物联网平台的Java SDK让开发人员可以方便地使用Java程序操作物联网平台。开发者可以使用Maven依赖添加SDK，也可以下载安装包到本地直接安装。

安装 SDK

1. 安装Java开发环境。

   您可以从 Java 官方网站 下载，并按说明安装Java开发环境。

2. 安装IoT Java SDK。

   a. 访问 Apache Maven 官网下载Maven软件。

   b. 添加Maven项目依赖。

   IoT Java SDK的Maven依赖坐标

```
<!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk
-iot -->
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-iot</artifactId>
    <version>7.0.0</version>
</dependency>
```

   依赖公共包

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>3.5.1</version>
</dependency>
```

初始化SDK

> 说明:
> 以下示例以华东2地域及其服务接入地址为例。您在设置时，需使用您的物联网平台地域和对应的服务接入地址。

```
String accessKey = "<your accessKey>";
String accessSecret = "<your accessSecret>";
DefaultProfile.addEndpoint("cn-shanghai", "cn-shanghai", "Iot", "iot.
cn-shanghai.aliyuncs.com");
IClientProfile profile = DefaultProfile.getProfile("cn-shanghai",
accessKey, accessSecret);
DefaultAcsClient client = new DefaultAcsClient(profile); //初始化SDK客户
端
```

accessKey即您的账号的AccessKeyId， accessSecret即AccessKeyId对应的AccessKeySecret。您可在阿里云官网控制台AccessKey管理中创建或查看您的AccessKey。

发起调用

物联网平台云端API，请参见#unique_11。

以调用Pub接口发布消息到Topic为例。

```
PubRequest request = new PubRequest();
request.setProductKey("productKey");
request.setMessageContent(Base64.encodeBase64String("hello world".
getBytes()));
request.setTopicFullName("/productKey/deviceName/get");
request.setQos(0); //目前支持QoS0和QoS1
try
{
   PubResponse response = client.getAcsResponse(request);
   System.out.println(response.getSuccess());
   System.out.println(response.getErrorMessage());
}
catch (ServerException e)
{
   e.printStackTrace();
}
 catch (ClientException e)
{
   e.printStackTrace();
}
```

附录：Demo

单击下载云端SDK Demo。Demo中包含Java、Python、PHP、.NET版本SDK示例。

另外，阿里云提供API在线调试工具 OpenAPI Explorer。在OpenAPI Explorer页，您可以快速检索和试验调用API。系统会根据您输入的参数同步生成各语言SDK的Demo代码。各语言SDK Demo显示在页面右侧示例代码页签下。在调试结果页签下，查看API调用的真实请求URL和JSON格式的返回结果。

1.3 Python SDK使用说明

物联网平台提供Python语言的云端SDK供开发人员使用。本文介绍云端Python SDK的安装和配置，及使用Python SDK调用云端API的示例。

安装Python SDK

1. 安装Python开发环境。

   访问Python官网下载Python安装包，并完成安装。目前，支持2.6.5及以上版本。

2. 安装Python的包管理工具pip。（如果您已安装pip，请忽略此步骤。）

   访问 pip 官网下载pip安装包，并完成安装。

3. 安装IoT Python SDK。

   以管理员权限执以下命令，安装IoT Python SDK。请参见最新版aliyun-python-sdk-iot信息。

```
sudo pip install aliyun-python-sdk-core
sudo pip install aliyun-python-sdk-iot
```

4. 将IoT Python SDK相关文件引入Python文件。

```
from aliyunsdkcore import client
from aliyunsdkiot.request.v20180120 import RegisterDeviceRequest
from aliyunsdkiot.request.v20180120 import PubRequest
...
```

初始化SDK

```
accessKeyId = '<your accessKey>'
 accessKeySecret = '<your accessSecret>'
 clt = client.AcsClient(accessKeyId, accessKeySecret, 'cn-shanghai')
```

accessKeyId即您的账号的AccessKeyId， accessKeySecret即AccessKeyId对应的AccessKeySecret。您可在阿里云官网控制台AccessKey管理中创建或查看您的AccessKey。

发起调用

物联网平台云端API，请参见#unique_11。

以调用Pub接口发布消息到设备为例。

```
request = PubRequest.PubRequest()
request.set_accept_format('json')  #设置返回数据格式，默认为XML，此例中设置为JSON
request.set_ProductKey('productKey')
request.set_TopicFullName('/productKey/deviceName/get')  #消息发送到的Topic全名
request.set_MessageContent('aGVsbG8gd29ybGQ=')  #hello world Base64 String
request.set_Qos(0)
result = clt.do_action_with_exception(request)
print 'result : ' + result
```

附录：Demo

单击下载云端SDK Demo。Demo中包含Java、Python、PHP、.NET版本SDK示例。

另外，阿里云提供API在线调试工具 OpenAPI Explorer。在OpenAPI Explorer页，您可以快速检索和试验调用API。系统会根据您输入的参数同步生成各语言SDK的Demo代码。各语言SDK Demo显示在页面右侧示例代码页签下。在调试结果页签下，查看API调用的真实请求URL和JSON格式的返回结果。

1.4 PHP SDK使用说明

物联网平台提供PHP语言的云端SDK供开发人员使用。本文介绍云端PHP SDK的安装和配置，及使用PHP SDK调用云端API的示例。

安装IoT PHP SDK

IoT PHP SDK是Alibaba Cloud SDK for PHP的一部分。如果您已安装*Alibaba Cloud SDK for PHP*，则无需安装IoT PHP SDK。

1. 安装PHP开发环境。

   需安装PHP 5.5.0或更高版本。访问PHP官网下载PHP安装包，并完成安装。

2. 安装Composer。

   目前，通过Composer管理IoT PHP SDK，因此需在系统中安装Composer。

   - Windows系统用户，请访问getcomposer.org，下载、安装Composer-Setup.exe。
   - 使用cURL命令安装Composer。

     ```
     curl -sS https://getcomposer.org/installer | php
     ```

3. 添加以下依赖，安装IoT PHP SDK。

   ```
   composer require alibabacloud/iot
   ```

   PHP SDK详情和使用指导，请参见openapi-sdk-php-iot和Alibaba Cloud SDK for PHP。

初始化SDK

初始化SDK示例代码如下：

```
<?php
include_once 'aliyun-php-sdk-core/Config.php';
use \Iot\Request\V20180120 as Iot;
//设置您的AccessKeyId/AccessSecret/ProductKey
$accessKeyId = "";
$accessSecret = "";
$iClientProfile = DefaultProfile::getProfile("cn-shanghai", $
accessKeyId, $accessSecret);
$client = new DefaultAcsClient($iClientProfile);
```

`accessKeyId`即您的账号的AccessKeyId，`accessSecret`即AccessKeyId对应的AccessKeySecret。您可在阿里云官网控制台AccessKey管理中创建或查看您的AccessKey。

发起调用

物联网平台云端API，请参见#unique_11。

以调用Pub接口发布数据到设备为例。

```
$request = new Iot\PubRequest();
$request->setProductKey("productKey");
$request->setMessageContent("aGVsbG93b3JsZA="); //hello world Base64
String.
$request->setTopicFullName("/productKey/deviceName/get"); //消息发送到的
Topic全名.
$response = $client->getAcsResponse($request);
print_r($response);
```

附录：Demo

单击下载云端SDK Demo。Demo中包含Java、Python、PHP、.NET版本SDK示例。

另外，阿里云提供API在线调试工具 OpenAPI Explorer。在OpenAPI Explorer页，您可以快速检索和试验调用API。系统会根据您输入的参数同步生成各语言SDK的Demo代码。各语言SDK Demo显示在页面右侧示例代码页签下。在调试结果页签下，查看API调用的真实请求URL和JSON格式的返回结果。

1.5 .NET SDK使用说明

物联网平台提供.NET语言的云端SDK供开发人员使用。本文介绍云端.NET SDK的安装和配置，及使用.NET SDK调用云端API的示例。

安装 IoT .NET SDK

1. 安装.NET开发环境。

   阿里云.NET SDK支持的开发环境如下：

   - .NET Framework 4.0及以上版本。
   - .NET Standard 2.0及以上版本。
   - C# 4.0及以上版本。
   - Visual Studio 2010 及以上版本。

2. 通过NuGet程序包管理器安装SDK。

   以使用Visual Studio为例。

   a. 在Visual Studio的解决方案资源管理器中，右键单击您的项目后，在菜单中选择管理NuGet程序包。
   b. 在NuGet 管理面板中，单击浏览。
   c. 在选项卡中，输入aliyun-net-sdk，然后在列表中选择Authors为Alibaba Cloud的aliyun-net-sdk-iot。
   d. 单击安装。

初始化SDK

> **说明:**
> 以下示例以华东2地域及其服务接入地址为例。您在设置时，需使用您的物联网平台地域和对应的服务接入地址。

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
DefaultProfile.AddEndpoint("cn-shanghai", "cn-shanghai", "Iot", "iot.
cn-shanghai.aliyuncs.com");
IClientProfile clientProfile = DefaultProfile.GetProfile("cn-shanghai
", "<your-access-key-id>", "<your-access-key-secret>");
DefaultAcsClient client = new DefaultAcsClient(clientProfile);
```

请在阿里云官网控制台AccessKey管理中创建或查看您的AccessKeyId和AccessKeySecret。

发起调用

物联网平台云端API，请参见#unique_11。

以调用Pub接口向Topic发布消息为例。

```
PubRequest request = new PubRequest();
request.ProductKey = "<productKey>";
request.TopicFullName = "/<productKey>/<deviceName>/get";
byte[] payload = Encoding.Default.GetBytes("Hello World.");
String payloadStr = Convert.ToBase64String(payload);
request.MessageContent = payloadStr;
request.Qos = 0;
try
{
   PubResponse response = client.GetAcsResponse(request);
   Console.WriteLine("publish message result: " + response.Success);
   Console.WriteLine(response.ErrorMessage);
}
catch (ServerException e)
{
   Console.WriteLine(e.ErrorCode);
   Console.WriteLine(e.ErrorMessage);
}
catch (ClientException e)
{
   Console.WriteLine(e.ErrorCode);
   Console.WriteLine(e.ErrorMessage);
}
```

附录：Demo

单击下载云端SDK Demo。Demo中包含Java、Python、PHP、.NET版本SDK示例。

另外，阿里云提供API在线调试工具 OpenAPI Explorer。在OpenAPI Explorer页，您可以快速检索和试验调用API。系统会根据您输入的参数同步生成各语言SDK的Demo代码。各

语言SDK Demo显示在页面右侧示例代码页签下。在调试结果页签下，查看API调用的真实请求URL和JSON格式的返回结果。

2 云端API参考

2.1 概述

物联网平台提供云端管理产品、设备、分组、Topic、规则、设备影子等API接口，和从云端发布消息的API接口。使用云端SDK，向API的服务端地址发送HTTPS/HTTP GET或POST请求，并按照API接口说明，在请求中加入相应请求参数来调用API。物联网平台根据请求的处理情况，返回处理结果。

调用API的方法和说明，请参见以下链接文档。

- #unique_17
- 公共参数
- 错误码

为更好的保护您的阿里云账号安全，建议使用RAM子账号用户身份来调用API。

授予子账号IoT API访问权限，请参见 IoT API 授权映射表。

2.2 调用API

本文档主要介绍调用物联网平台云端API的请求结构和请求示例。

请求结构

您可以通过发送HTTP或HTTPS请求调用物联网平台API。

请求结构如下：

```
http://Endpoint/?Action=xx&Parameters
```

参数	说明
Endpoint	调用云服务的接入地址。物联网平台的接入地址格式：`iot.${RegionId}.aliyuncs.com`。其中，变量*${RegionId}*需替换为您的物联网平台服务的地域代码。阿里云地域代码，请参见#unique_22。 接入地址示例： · 华东2（上海）：`iot.cn-shanghai.aliyuncs.com` · 新加坡：`iot.ap-southeast-1.aliyuncs.com` · 美国（硅谷）：`iot.us-west-1.aliyuncs.com` · 日本（东京）：`iot.ap-northeast-1.aliyuncs.com` · 德国（法兰克福）：`iot.eu-central-1.aliyuncs.com`
Action	要执行的操作，即云端API接口的名称。例如，调用Pub接口向指定Topic发布消息，`Action`对应的值就是Pub，即`Action=Pub`。
Parameters	请求参数。每个参数之间用（&）符号分隔。 请求参数由公共请求参数和API自定义参数组成。公共参数中包含API版本号、身份验证等信息。

下面以调用Pub接口向指定Topic发布消息为例：

说明：

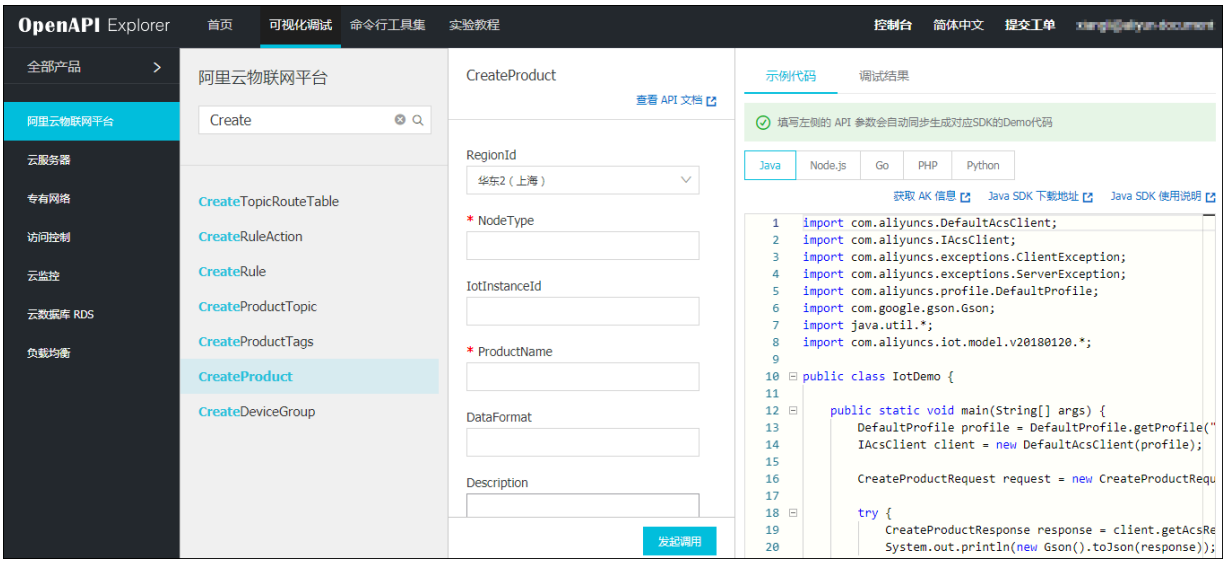
本文档示例均使用华东2（上海）地域的接入地址。为了便于阅读，本文档中的示例均做了格式化处理。

```
https://iot.cn-shanghai.aliyuncs.com/?Action=Pub
&Format=XML
&Version=2017-04-20
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=...
&Timestamp=2017-07-19T12:00:00Z
&RegionId=cn-shanghai
...
```

API在线调试

阿里云提供API在线调试工具 OpenAPI Explorer。在OpenAPI Explorer页，您可以快速检索和试验调用API。系统会根据您输入的参数同步生成各语言SDK的Demo代码。各语言SDK

Demo显示在页面右侧示例代码页签下供您参考。在调试结果页签下，查看API调用的真实请求URL和JSON格式的返回结果。



API授权

为了确保您的账号安全，建议您使用子账号的身份凭证调用API。如果您使用RAM子账号调用物联网平台API，您需要为该RAM子账号创建、授予相应的授权策略。

为子账号授权调用API，请参见IoT API 授权映射表。

2.3 公共参数

本文档介绍物联网平台云端API的公共请求参数和公共返回参数。

公共请求参数

公共请求参数是调用每个API时都需要使用的请求参数。

名称	类型	是否必需	描述
Format	String	否	返回值的类型，支持JSON和XML类型。默认为XML。
Version	String	是	API版本号，为日期形式：YYYY-MM-DD，最新版本为2018-01-20。每个接口可以存在多个版本。

名称	类型	是否必需	描述
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID。 登录阿里云控制台，将光标移至账号头像上，然后单击accesskeys，跳转至用户信息管理页，即可创建和查看AccessKey。
Signature	String	是	签名结果串。
SignatureMethod	String	是	签名方式，目前支持HMAC-SHA1。
Timestamp	String	是	请求的时间戳。日期格式按照ISO8601标准表示，并需要使用UTC时间。格式为YYYY-MM-DDThh:mm:ssZ。 例如，2016-01-04T12:00:00Z表示北京时间2016年01月04日20点0分0秒。
SignatureVersion	String	是	签名算法版本。目前版本是1.0。
SignatureNonce	String	是	唯一随机数。用于防止网络重放攻击。用户在不同请求中要使用不同的随机数值。
RegionId	String	是	设备所在地域（与控制台上的地域对应），如cn-shanghai。

示例

```
https://iot.cn-shanghai.aliyuncs.com/
?Format=XML
&Version=2018-01-20
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=...
&Timestamp=2018-05-20T12:00:00Z
&RegionId=cn-shanghai
```

公共返回参数

API返回结果采用统一格式，返回2xx HTTP状态码代表调用成功；返回4xx或5xx HTTP状态码代表调用失败。调用成功返回的数据格式有XML和JSON两种。可以在发送请求时，指定返回的数据格式。默认为XML格式。

每次接口调用，无论成功与否，系统都会返回一个唯一识别码RequestId。

- 调用成功的返回示例。
  - XML格式

```
<?xml version="1.0" encoding="UTF-8"?>
<!--结果的根结点-->
<接口名称+Response>
    <!--返回请求标签-->
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!--返回结果数据-->
</接口名称+Response>
```

  - JSON格式

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
    /* 返回结果数据 */
}
```

- 调用失败的返回示例。

  调用接口出错后，将不会返回结果数据。可根据错误码来定位错误原因。

  当调用出错时，HTTP请求返回一个4xx或5xx的HTTP状态码。返回的消息体中是具体的错误代码及错误信息。另外，还包含一个全局唯一的请求ID（RequestId）。在您不能确认错误的情况下，可以联系阿里云客服或提交工单，并提供RequestId值，以便工作人员尽快帮您解决问题。

  - XML格式

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
   <RequestId>8906582E-6722-409A-A6C4-0E7863B733A5</RequestId>
   <Code>UnsupportedOperation</Code>
   <Message>The specified action is not supported.</Message>
</Error>
```

  - JSON格式

```
{
    "RequestId": "8906582E-6722-409A-A6C4-0E7863B733A5",
    "Code": "UnsupportedOperation",
    "Message": "The specified action is not supported."
```

```
}
```

2.4 签名机制

物联网平台会对每个接口访问请求的发送者进行身份验证，所以无论使用HTTP还是HTTPS协议提交请求，都需要在请求中包含签名（Signature）信息。

签名方法

签名时，您需在控制台 AccessKey 管理页面查看您的阿里云账号的AccessKeyId和AccessKeySecret，然后进行对称加密。其中，AccessKeyId用于标识访问者身份；AccessKeySecret是用于加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密。

 说明:

物联网平台提供了Java、Python、PHP等语言的服务端SDK。使用这些SDK，可以免去签名过程。请参见#unique_25及各SDK的使用说明。

请按照下面的方法对请求进行签名：

1. 构造规范化的请求字符串（Canonicalized Query String）。

   a. 排序参数。

      按参数名的字典顺序，对请求参数进行排序，包括 公共请求参数（不包括Signature参数）和接口的自定义参数。

       说明:

当使用GET方法提交请求时，这些参数就是请求URL中的参数部分，即URL中?之后由&连接的部分。

b. 对参数名称和参数值进行URL编码。

使用UTF-8字符集按照 RFC3986 规则编码请求参数名和参数值。编码规则如下：

- 字符A~Z、a~z、0~9以及字符-、_、.、~不编码。
- 其它字符编码成%XY的格式，其中XY是字符对应ASCII码的16进制表示。例如英文的双引号"对应的编码为%22。
- 扩展的UTF-8字符，编码成%XY%ZA...的格式。
- 英文空格要编码成%20，而不是加号+。

该编码方式与application/x-www-form-urlencodedMIME格式编码算法相似，但又有所不同。

如果您使用的是Java标准库中的java.net.URLEncoder，可以先用标准库中percentEncode编码，随后将编码后的字符中加号+替换为%20、星号*替换为%2A、%7E替换为波浪号~，即可得到上述规则描述的编码字符串。

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws
UnsupportedEncodingException {
return value != null ? URLEncoder.encode(value, ENCODING).replace
("+", "%20").replace("*", "%2A").replace("%7E", "~") : null;
}
```

c. 使用等号=连接编码后的请求参数名和参数值。

d. 使用与号&连接编码后的请求参数。参数排序与步骤a的排序一致。

完成后，即获得规范化请求字符串（CanonicalizedQueryString）。

2. 构造签名字符串。

可以使用percentEncode处理步骤1得到的规范化字符串，构造签名字符串。可参考如下规则：

```
StringToSign=
  HTTPMethod + "&" + //HTTPMethod：发送请求的HTTP方法，例如GET。
  percentEncode("/") + "&" + //percentEncode("/")：字符（/）UTF-8编码得到的值，即%2F。
```

```
  percentEncode(CanonicalizedQueryString) //您的规范化请求字符串。
```

3. 计算HMAC值。

   按照RFC2104的定义，使用步骤2得到的字符串StringToSign计算签名HMAC值。示例使用的是Java Base64编码方法。

```
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(
StringToSign) ) )
```

   说明:

   计算签名时，使用的Key就是您的AccessKeySecret并加上一个与号&字符（ASCII:38）。

   使用的哈希算法是SHA1。

4. 计算签名值。

   按照Base64编码规则把步骤3中的HMAC值编码成字符串，即得到签名值（Signature）。

5. 添加签名。

   将得到的签名值作为Signature参数，按照RFC3986的规则进行URL编码后，再添加到请求参数中，即完成对请求签名的过程。

签名示例

以调用Pub接口为例。假设您的AccessKeyId=testid，AccessKeySecret=testsecret。

1. 组成签名前的请求URL。

```
http://iot.cn-shanghai.aliyuncs.com/?MessageContent=aGVsbG93b3JsZA
%3D&Action=Pub&Timestamp=2017-10-02T09%3A39%3A41Z&SignatureVersion
=1.0&ServiceCode=iot&Format=XML&Qos=0&SignatureNonce=0715a395-
aedf-4a41-bab7-746b43d38d88&Version=2017-04-20&AccessKeyId=testid&
SignatureMethod=HMAC-SHA1&RegionId=cn-shanghai&ProductKey=12345abcde
Z&TopicFullName=%2FproductKey%2Ftestdevice%2Fget
```

2. 计算得到待签名字符串StringToSign。

```
GET&%2F&AccessKeyId%3Dtestid%26Action%3DPub%26Format%3DXML%
26MessageContent%3DaGVsbG93b3JsZA%253D%26ProductKey%3D12345abcdeZ
%26Qos%3D0%26RegionId%3Dcn-shanghai%26ServiceCode%3Diot%26Signatur
eMethod%3DHMAC-SHA1%26SignatureNonce%3D0715a395-aedf-4a41-bab7-
746b43d38d88%26SignatureVersion%3D1.0%26Timestamp%3D2017-10-02T09
```

```
%253A39%253A41Z%26TopicFullName%3D%252FproductKey%252Ftestdevice%
252Fget%26Version%3D2017-04-20
```

3. 计算签名值。

   因为AccessKeySecret=testsecret，用于计算的Key为testsecret&，计算得到的签名值为：

```
Y9eWn4nF8QPh3c4zAFkM/k/u7eA=
```

4. 将签名作为Signature参数加入到URL请求中，最后得到的URL为：

```
http://iot.cn-shanghai.aliyuncs.com/?MessageContent=aGVsbG93b3JsZA
%3D&Action=Pub&Timestamp=2017-10-02T09%3A39%3A41Z&SignatureVersion
=1.0&ServiceCode=iot&Format=XML&Qos=0&SignatureNonce=0715a395-
aedf-4a41-bab7-746b43d38d88&Version=2017-04-20&AccessKeyId=testid
&Signature=Y9eWn4nF8QPh3c4zAFkM%2Fk%2Fu7eA%3D&SignatureMethod=HMAC
-SHA1&RegionId=cn-shanghai&ProductKey=12345abcdeZ&TopicFullName=%
2FproductKey%2Ftestdevice%2Fget
```

JAVA代码示例

以下为签名的JAVA demo供您参考。

1. 配置文件*Config.java*。

```
/*
 * Copyright © 2018 Alibaba. All rights reserved.
 */
package com.aliyun.iot.demo.sign;

/**
 * 服务端API签名配置文件
 *
 * @author: ali
 * @version: 0.1 2018-08-08 08:23:54
 */
public class Config {

    // AccessKey信息
    public static String accessKey = "1234567890123456";
    public static String accessKeySecret = "12345678901234567890
1234567890";

    public final static String CHARSET_UTF8 = "utf8";
}
```

2. 配置文件*UrlUtil.java*。

```
/*
 * Copyright © 2018 Alibaba. All rights reserved.
 */
package com.aliyun.iot.demo.sign;

import java.net.URLEncoder;
import java.util.Map;

import org.apache.commons.lang3.StringUtils;
```

```
/**
 * URL处理类
 *
 * @author: ali
 * @version: 0.1 2018-06-21 20:40:52
 */
public class UrlUtil {

    private final static String CHARSET_UTF8 = "utf8";

    public static String urlEncode(String url) {
        if (!StringUtils.isEmpty(url)) {
            try {
                url = URLEncoder.encode(url, "UTF-8");
            } catch (Exception e) {
                System.out.println("Url encode error:" + e.
getMessage());
            }
        }
        return url;
    }

    public static String generateQueryString(Map<String, String>
params, boolean isEncodeKV) {
        StringBuilder canonicalizedQueryString = new StringBuilder
();
        for (Map.Entry<String, String> entry : params.entrySet()) {
            if (isEncodeKV)
                canonicalizedQueryString.append(percentEncode(entry.
getKey())).append("=")
                        .append(percentEncode(entry.getValue())).
append("&");
            else
                canonicalizedQueryString.append(entry.getKey()).
append("=").append(entry.getValue()).append("&");
        }
        if (canonicalizedQueryString.length() > 1) {
            canonicalizedQueryString.setLength(canonicalizedQuerySt
ring.length() - 1);
        }
        return canonicalizedQueryString.toString();
    }

    public static String percentEncode(String value) {
        try {
            // 使用URLEncoder.encode编码后，将"+","*","%7E"做替换即满足
API规定的编码规范
            return value == null ? null
                    : URLEncoder.encode(value, CHARSET_UTF8).replace
("+", "%20").replace("*", "%2A").replace("%7E",
                            "~");
        } catch (Exception e) {

        }
        return "";
    }
}
```

3. 配置文件*SignatureUtils.java*。

```
/*
 * Copyright © 2018 Alibaba. All rights reserved.
 */
```

```
package com.aliyun.iot.demo.sign;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Map;
import java.util.TreeMap;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang3.StringUtils;

/**
 * 服务端API签名
 *
 * @author: ali
 * @version: 0.1 2018-06-21 20:47:05
 */
public class SignatureUtils {

    private final static String CHARSET_UTF8 = "utf8";
    private final static String ALGORITHM = "UTF-8";
    private final static String SEPARATOR = "&";

    public static Map<String, String> splitQueryString(String url)
            throws URISyntaxException, UnsupportedEncodingException
 {
        URI uri = new URI(url);
        String query = uri.getQuery();
        final String[] pairs = query.split("&");
        TreeMap<String, String> queryMap = new TreeMap<String,
String>();
        for (String pair : pairs) {
            final int idx = pair.indexOf("=");
            final String key = idx > 0 ? pair.substring(0, idx) :
pair;
            if (!queryMap.containsKey(key)) {
                queryMap.put(key, URLDecoder.decode(pair.substring(
idx + 1), CHARSET_UTF8));
            }
        }
        return queryMap;
    }

    public static String generate(String method, Map<String, String
> parameter, String accessKeySecret)
            throws Exception {
        String signString = generateSignString(method, parameter);
        System.out.println("signString---" + signString);
        byte[] signBytes = hmacSHA1Signature(accessKeySecret + "&",
signString);
        String signature = newStringByBase64(signBytes);
        System.out.println("signature----" + signature);
        if ("POST".equals(method))
            return signature;
        return URLEncoder.encode(signature, "UTF-8");
    }
```

```
    public static String generateSignString(String httpMethod, Map<
String, String> parameter) throws IOException {
        TreeMap<String, String> sortParameter = new TreeMap<String,
String>();
        sortParameter.putAll(parameter);
        String canonicalizedQueryString = UrlUtil.generateQu
eryString(sortParameter, true);
        if (null == httpMethod) {
            throw new RuntimeException("httpMethod can not be empty
");
        }
        StringBuilder stringToSign = new StringBuilder();
        stringToSign.append(httpMethod).append(SEPARATOR);
        stringToSign.append(percentEncode("/")).append(SEPARATOR);
        stringToSign.append(percentEncode(canonicalizedQueryString
));
        return stringToSign.toString();
    }

    public static String percentEncode(String value) {
        try {
            return value == null ? null
                    : URLEncoder.encode(value, CHARSET_UTF8).replace
("+", "%20").replace("*", "%2A").replace("%7E",
                            "~");
        } catch (Exception e) {
        }
        return "";
    }

    public static byte[] hmacSHA1Signature(String secret, String
baseString) throws Exception {
        if (StringUtils.isEmpty(secret)) {
            throw new IOException("secret can not be empty");
        }
        if (StringUtils.isEmpty(baseString)) {
            return null;
        }
        Mac mac = Mac.getInstance("HmacSHA1");
        SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(
CHARSET_UTF8), ALGORITHM);
        mac.init(keySpec);
        return mac.doFinal(baseString.getBytes(CHARSET_UTF8));
    }

    public static String newStringByBase64(byte[] bytes) throws
UnsupportedEncodingException {
        if (bytes == null || bytes.length == 0) {
            return null;
        }
        return new String(Base64.encodeBase64(bytes, false),
CHARSET_UTF8);
    }
}
```

4. **配置主入口文件***Main.java*。

```
/*
 * Copyright © 2018 Alibaba. All rights reserved.
 */
package com.aliyun.iot.demo.sign;

import java.io.UnsupportedEncodingException;
```

```
import java.net.URLEncoder;
import java.util.HashMap;
import java.util.Map;

/**
 * 签名工具主入口
 *
 * @author: ali
 * @version: 0.1 2018-09-18 15:06:48
 */
public class Main {

    // 1.需求修改Config.java中的AccessKey信息
    // 2.建议使用方法二，所有参数都需要一一填写
    // 3."最终signature"才是你需要的签名最终结果
    public static void main(String[] args) throws Unsupporte
dEncodingException {

        // 方法一
        System.out.println("方法一: ");
        String str = "GET&%2F&AccessKeyId%3D" + Config.accessKey
                + "%26Action%3DRegisterDevice%26DeviceName%
3D1533023037%26Format%3DJSON%26ProductKey%3DaxxxUtgaRLB%26RegionId
%3Dcn-shanghai%26SignatureMethod%3DHMAC-SHA1%26SignatureNonce%
3D1533023037%26SignatureVersion%3D1.0%26Timestamp%3D2018-07-31T07%
253A43%253A57Z%26Version%3D2018-01-20";
        byte[] signBytes;
        try {
            signBytes = SignatureUtils.hmacSHA1Signature(Config.
accessKeySecret + "&", str.toString());
            String signature = SignatureUtils.newStringByBase64(
signBytes);
            System.out.println("signString---" + str);
            System.out.println("signature----" + signature);
            System.out.println("最终signature: " + URLEncoder.encode(
signature, Config.CHARSET_UTF8));
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println();

        // 方法二
        System.out.println("方法二: ");
        Map<String, String> map = new HashMap<String, String>();
        // 公共参数
        map.put("Format", "JSON");
        map.put("Version", "2018-01-20");
        map.put("AccessKeyId", Config.accessKey);
        map.put("SignatureMethod", "HMAC-SHA1");
        map.put("Timestamp", "2018-07-31T07:43:57Z");
        map.put("SignatureVersion", "1.0");
        map.put("SignatureNonce", "1533023037");
        map.put("RegionId", "cn-shanghai");
        // 请求参数
        map.put("Action", "RegisterDevice");
        map.put("DeviceName", "1533023037");
        map.put("ProductKey", "axxxUtgaRLB");
        try {
            String signature = SignatureUtils.generate("GET", map,
Config.accessKeySecret);
            System.out.println("最终signature: " + signature);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            System.out.println();
        }
}
```

2.5 错误码

本文档列举调用物联网平台API出错时，返回的错误信息。入参数据格式错误、超出限定值、入参缺少必需参数等错误修改，请参见具体API文档的请求参数描述。

系统错误码

以`iot.system`开头的错误码为系统相关错误码。

错误码	描述
iot.system.SystemException	系统异常。 请稍后重试。

公共错误码

以`iot.common`开头的错误码为公共错误码。

错误码	描述
iot.common.InvalidPageParams	分页大小或者分页页号不合法。 请参见具体API文档的分页相关参数描述，如`PageSize`。
iot.common.InvalidTenant	不合法的租户。 请确认阿里云账号信息和账号权限。
iot.common.QueryDeviceActionError	查询设备失败。 请确认入参信息正确，然后重试。
iot.common.QueryDevicePropertyActionError	查询设备属性失败。 请确认入参信息正确，然后重试。
iot.common.QueryProductActionError	查询产品失败。 请确认入参信息正确，然后重试。
iot.common.QueryProductCountActionError	查询产品总数失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.common.RamActionPermissionDeny	没有资源访问控制（RAM）权限。 请参见子账号授权文档。
iot.common.AuthActionPermissionDeny	鉴权失败。 原因可能是入参的设备信息不属于当前账号。请确认AccessKey信息和设备信息。

产品（Product）相关错误码

以`iot.prod`开头的错误码为产品相关错误码。

错误码	描述
iot.prod.AlreadyExistedProductName	已经存在相同的产品名称。一个阿里云账号下的产品名称不能重复。
iot.prod.CreateProductFailed	创建产品失败。 请确认入参信息正确，然后重试。
iot.prod.CreateProductTopicFailed	创建产品的Topic类失败。 请确认入参信息正确，然后重试。
iot.prod.InvalidAliyunCommodityCode	入参`AliyunCommodityCode`值错误。 AliyunCommodityCode的可选值只有`iothub_senior`和`iothub`。
iot.prod.InvalidFormattedCatId	入参`CategoryId`（产品的设备类型）错误。
iot.prod.InvalidFormattedProductkey	入参产品ProductKey格式错误。 请核对输入的`ProductKey`值。
iot.prod.InvalidFormattedProductName	入参产品名称格式错误。 产品名应满足以下限制：由中文、英文字母、数字和下划线（_）组成，长度为4-30位（一个中文字符占两位）。
iot.prod.LongProductDesc	产品描述字符数超出限定值。 描述信息应在100字符以内。

错误码	描述
iot.prod.InvalidNodeType	产品的节点类型错误。 节点类型支持的可选值： · 0：设备 · 1：网关
iot.prod.NotExistedProduct	产品不存在。 输入的ProductKey值在当前账号下不存在。
iot.prod.NotOpenID2Service	没有开通ID²服务。 该产品在创建时没有开通ID²安全认证服务。ID²安全认证服务只能在创建产品时开通，并且，产品创建成功后，不能更改是否使用ID²认证的状态。
iot.prod.NotSeniorProduct	产品不是高级版产品。
iot.prod.NullProductKey	入参产品ProductKey不能为空。
iot.prod.NullProductName	入参产品名称不能为空。
iot.prod.ProductCountExceedMax	产品总数已超过最大限制数量。 一个阿里云账号下最多可有1,000个产品。
iot.prod.QueryDeviceCountActionError	查询产品下的设备总数失败。 请确认入参信息正确，然后重试。
iot.prod.QueryProductAbilitiesFailed	获取产品功能失败。 请确认入参信息是否正确，如Identifier值等。
iot.prod.QueryProductAbilityFailed	查询产品功能失败。 请确认入参信息是否正确，如Identifier值等。
iot.prod.QueryProductListActionError	获取产品列表数据失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.prod.UpdateProductFailed	更新产品信息失败。 请确认入参信息正确，然后重试。

设备（Device）相关错误码

以`iot.device`开头的错误码为设备相关错误码。

错误码	描述
iot.device.AddTopoRelationFailed	添加拓扑关系失败。 请确认入参信息正确，然后重试。
iot.device.AlreadyExistedDeviceName	设备名称已经存在。 设备名称需在产品维度唯一。
iot.device.ApplyManyDevicesFailed	申请批量创建设备失败。 请确认入参信息正确，然后重试。
iot.device.CreateDeviceFailed	创建设备失败。 请确认入参信息正确，然后重试。
iot.device.CreateDeviceTaskIsRunning	创建设备的申请任务还在执行中。
iot.device.DeviceApplyIsNotFound	申请设备的申请单不存在。 请确认输入的`ApplyId`值。其值需与您调用#unique_27返回的`ApplyId`值一致。
iot.device.DeviceCountExceeded	批量申请的设备数量超过最大值。 单次调用，最多批量注册1,000 个设备。
iot.device.DeleteDeviceFailed	删除设备失败。 请确认入参信息正确，然后重试。
iot.device.DeleteDevicePropertyFailed	删除设备属性失败。 请确认入参信息正确，然后重试。
iot.device.DisableDeviceFailed	禁用设备失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.device.EnableDeviceFailed	启用设备失败。 请确认入参信息正确，然后重试。
iot.device.InactiveDevice	设备未激活，即物理设备从未连接物联网平台。
iot.device.InvalidFormattedApplyId	创建设备的申请单（ApplyId）错误。 其值需与您调用#unique_27返回的ApplyId值一致。
iot.device.IncorrentDeviceApplyInfo	设备申请信息错误。 请确认入参信息，如ApplyId等。
iot.device.InvalidFormattedDeviceName	设备名称格式错误。 设备名称长度为4-32个字符，可以包含英文字母、数字和特殊字符：连字符（-）、下划线（_）、at符号（@）、点号（.）、和英文冒号（:）。
iot.device.InvalidFormattedDevicePropertyKey	设备属性标识符格式错误。 请查看相关API文档中，关于入参属性格式的描述。
iot.device.InvalidFormattedDevicePropertiesString	入参设备属性格式错误。 请查看相关API文档中，关于入参属性格式的描述。
iot.device.InvalidIoTId	设备ID错误。 请调用#unique_28或#unique_29查看正确的IotId值，或用ProductKey与DeviceName组合代替IotId。
iot.device.InvalidTimeBucket	指定的时间区间不合法。 请根据API文档中描述正确设置参数。 · Asc为0倒序查询时，StartTime必须大于EndTime。 · Asc为1正序查询时，StartTime必须小于EndTime。

错误码	描述
iot.device.InvokeThingServiceFailed	调用设备服务失败。 请检查输入参数是否正确，如Args的参数格式和取值等。
iot.device.LongDevicePropertiesString	入参设备属性长度超过最大值。 请查看相关API文档的限制说明。
iot.device.NoneDeviceNameElement	设备名称列表为空。
iot.device.NoneDeviceProperties	没有有效的设备属性。 请核对传入的属性Identifier是否与TSL中定义的一致。
iot.device.NotExistedDevice	设备不存在。 传入的设备IotId、ProductKey或DeviceName值错误。请调用#unique_28或#unique_29查看正确值。
iot.device.NullApplyId	创建设备的申请ID（ApplyId）不能为空。
iot.device.NullDeviceName	设备名称不能为空。
iot.device.NullDevicePropertyKey	设备属性名称不能为空。
iot.device.NullDevicePropertiesString	入参设备属性不能为空。
iot.device.QueryDeviceApplyActionError	查询设备申请单信息出错。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceAttrDataHistoryFailed	获取设备属性数据历史记录失败。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceAttrStatusFailed	获取设备属性状态信息失败。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceEventHistoryFailed	获取设备事件调用记录失败。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceListActionError	查询设备列表失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.device.QueryDeviceServiceHistoryFailed	获取设备服务调用记录失败。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceStatisticsFailed	查询设备统计信息失败。 请确认入参信息正确，然后重试。
iot.device.QueryDeviceStatusFailed	查询设备状态信息失败。 请确认入参信息正确，然后重试。
iot.device.QueryTopoRelationFailed	查询拓扑关系失败。 请确认入参信息是否正确。如，传入的PageSize值大于限定值50会报此错误。
iot.device.RemoveTopoRelationFailed	移除拓扑关系失败。 请确认入参信息正确，然后重试。
iot.device.SaveOrUpdateDevicePropertiesFailed	新增或者修改设备属性失败。 请确认入参信息正确，然后重试。
iot.device.SetDevicePropertyFailed	设置设备属性失败。 请检查入参Items的参数值和格式是否正确，指定的属性是否是读写型。
iot.device.TooManyDevicePropertiesPerTime	传入的属性个数超过限定值。 请参见相关API文档限制说明。
iot.device.TopoRelationCountExceeded	拓扑关系数量过多。 请参见#unique_30中网关与子设备数量限制。
iot.device.VerifyDeviceFailed	验证设备失败。 请确认入参信息正确，然后重试。

设备分组（Group）相关错误码

以iot.group开头的错误码为设备分组相关错误码。

错误码	描述
iot.group.NullGroupId	入参分组ID没有赋值。

错误码	描述
iot.group.DeleteGroupFailed	删除分组失败。 请确认入参信息正确，然后重试。
iot.group.SubGroupNotNull	分组下有子分组。 当分组下有子分组时，不能删除分组，需先删除子分组。
iot.group.InvalidGroupName	分组名称不合法。 分组名称可包含中文汉字、英文字母、数字和下划线（_）。长度范围 4 - 30 字符（一个中文汉字占二个字符）。
iot.group.GroupNameExisted	分组名称已存在。
iot.group.QueryGroupInfoFailed	查询分组详情失败。 请确认入参信息正确，然后重试。
iot.group.NotExistedGroup	分组不存在。 请确认GroupId值。
iot.group.QueryGroupCountFailed	查询分组数量失败。 请确认入参信息正确，然后重试。
iot.group.QueryGroupListFailed	查询分组列表失败。 请确认入参信息正确，然后重试。
iot.group.BindGroupRelationFailed	绑定分组关系失败。 请确认入参信息正确，然后重试。
iot.group.UpdateGroupFailed	修改分组信息失败。 请确认入参信息正确，然后重试。
iot.group.QueryGroupTreeFailed	获取分组关系结构失败。 请确认入参信息正确，然后重试。
iot.group.CreateGroupFailed	创建分组失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.group.InvalidFormattedTagString	标签格式不合法。 标签数据为JSON格式。由标签的tagKey和tagValue组成，tagKey和tagValue均不能为空。多个标签以英文逗号间隔。如，`[{"tagKey":"h1","tagValue":"rr"},{"tagKey":"7h","tagValue":"rr"}]`。
iot.group.TagCountExceedMax	标签数量超过最大值。 每个分组最多可有100个标签。
iot.group.GroupCountExceedMax	分组数量超过最大值。 · 一个分组最多可包含100个子分组。 · 一个设备最多可以被添加到10个分组中。
iot.group.SetGroupTagFailed	设置分组标签信息失败。 请确认入参信息正确，然后重试。
iot.group.QueryGroupTagFailed	查询分组标签信息失败。 请确认入参信息正确，然后重试。
iot.group.LongGroupDescError	分组描述字段过长。 分组描述长度限制为100字符（一个中文汉字占一个字符）。
iot.group.QueryGroupRelationFailed	查询分组关系失败。 请确认入参信息正确，然后重试。
iot.group.GroupLevelExceedingLimitError	分组层级超过限制。 分组只支持三级嵌套，即分组>子分组>子子分组。

消息相关错误码

以`iot.messagebroker`开头的错误码为消息相关错误码。此类错误码主要出现在调用消息通信相关API、设备影子相关API和规则引擎相关API失败时。（规则引擎相关API调用失败错误码，请见本文档下一章节。）

错误码	描述
iot.messagebroker.CreateTopicRouteFailed	创建Topic之间消息路由失败。 请确认入参信息正确，然后重试。
iot.messagebroker.CreateTopicTemplateException	创建Topic类过程发生异常。 请确认入参信息正确，然后重试。
iot.messagebroker.CreateTopicTemplateFailed	创建Topic类失败。 请确认入参信息正确，然后重试。
iot.messagebroker.DeleteTopicTemplateException	删除Topic类过程发生异常。 请确认入参信息正确，然后重试。
iot.messagebroker.DeleteTopicTemplateFailed	删除Topic类失败。 请确认入参信息正确，然后重试。
iot.messagebroker.DestTopicNameArraySizeIsLarge	同一消息源Topic配置的路由目标Topic数量超过最大限制数。 一个源Topic最多可对应100个目标Topic。
iot.messagebroker.DeleteTopicRouteFailed	删除指定Topic间的路由失败。 请确认入参信息正确，然后重试。
iot.messagebroker.DesireInfoInShadowMessageIsNotJson	设备影子中的desire信息不是JSON格式。
iot.messagebroker.DesireValueIsNullInShadowMessage	设备影子中的desire信息值为空。
iot.messagebroker.ElementKeyOrValueIsNullInDesire	desire信息包含有空的属性标识符或者属性值。
iot.messagebroker.ElementKeyOrValueIsNullInReport	report信息包含有空的属性标识符或者属性值。
iot.messagebroker.HALFCONN	由于设备为半连接状态导致失败。
iot.messagebroker.InvalidFormattedSrcTopicName	消息源Topic名称格式错误。 可在控制台设备详情页的Topic列表下查看设备的Topic。

错误码	描述
iot.messagebroker.InvalidFormattedTopicName	Topic格式错误。 可在控制台设备详情页的Topic列表下查看设备的Topic。
iot.messagebroker.InvalidFormattedTopicTemplateId	Topic类ID格式错误。 可调用#unique_31查看TopicId。
iot.messagebroker.InvalidTimeoutValue	超时时间参数设置有误。 请参见相关API文档查看时间设置方法。
iot.messagebroker.InvalidTopicTemplateOperationValue	Topic类的操作权限值错误。操作权限取值： SUB：订阅。 PUB：发布。 ALL：发布和订阅。
iot.messagebroker.InvalidVersionValueInShadowMessage	设备影子中的version值错误。
iot.messagebroker.MethodValueIsNotUpdate	设备影子中的method信息值不是update。
iot.messagebroker.MessageContentIsNotBase64Encode	消息内容没有经过base64编码。
iot.messagebroker.NoneElementInDesire	desire信息中没有属性。
iot.messagebroker.NoneElementInReport	report信息中没有属性。
iot.messagebroker.NoneElementDestTopicNameInArray	目标Topic列表中没有元素。
iot.messagebroker.NotFoundDesireInShadowMessage	设备影子的state信息中没有desire信息。
iot.messagebroker.NotFoundMethodInShadowMessage	设备影子没有method信息。
iot.messagebroker.NotFoundReportInShadowMessage	设备影子中没有report信息。
iot.messagebroker.NotFoundStateInShadowMessage	设备影子中没有state信息。

错误码	描述
iot.messagebroker.NotFoundVersionOrNullVersionValue	缺少version信息或者version值为空。
iot.messagebroker.NotMatchedProductKeyWithSrcTopicOwner	消息源Topic对应的产品ID不属于当前用户。
iot.messagebroker.NullMessageContent	消息内容不能为空。
iot.messagebroker.NullShadowMessage	设备影子内容不能为空。
iot.messagebroker.NullSrcTopicName	消息源Topic名称不能为空。
iot.messagebroker.NullTopicName	Topic不能为空。
iot.messagebroker.NullTopicTemplateId	Topic类ID不能为空。
iot.messagebroker.NullTopicTemplateOperation	Topic类的操作权限不能为空。
iot.messagebroker.OFFLINE	由于设备离线导致失败。
iot.messagebroker.PublishMessageException	发送消息过程出现异常。 请确认入参信息正确，然后重试。
iot.messagebroker.PublishMessageFailed	发送消息失败。 请确认入参信息正确，然后重试。
iot.messagebroker.QueryDeviceShadowActionError	查询设备影子失败。 请确认入参信息正确，然后重试。
iot.messagebroker.QueryProductTopicListActionError	获取Topic类列表失败。 请确认入参信息正确，然后重试。
iot.messageborker.QueryTopicReverseRouteTableListActionError	获取消息反向路由列表（即消息源Topic列表）失败。 请确认入参信息正确，然后重试。
iot.messageborker.QueryTopicRouteTableListActionError	获取消息路由列表失败。 请确认入参信息正确，然后重试。
iot.messagebroker.QueryTopicTemplateActionError	查询Topic类失败。 请确认入参信息正确，然后重试。
iot.messagebroker.QueryTopicTemplateException	获取Topic类过程发生异常。 请确认入参信息正确，然后重试。

错误码	描述
iot.messagebroker.RateLimit	由于限流导致失败。 请参见#unique_30。
iot.messagebroker.ReportInShadowMessageIsNotJson	设备影子中的state信息中的report信息不是JSON格式。
iot.messagebroker.RrpcException	RRPC发送消息过程出现异常。 请确认入参信息正确，然后重试。
iot.messagebroker.RrpcFailed	RRPC发送消息失败。 请确认入参信息正确，然后重试。
iot.messagebroker.ShadowMessageIsNotJson	设备影子不是JSON格式。
iot.messagebroker.ShadowMessageLengthIsLarge	设备影子的长度超过最大限制。 设备影子文档的大小限制16 KB。
iot.messagebroker.TIMEOUT	由于超时导致失败。
iot.messagebroker.TooManyElementInDesire	desire信息中包含的属性总数超过最大限定数。 设备影子JSON文档的属性数量限制为128。
iot.messagebroker.TooManyElementInReport	report信息包含的属性总数超过限定最大数。 设备影子JSON文档的属性数量限制为128。
iot.messagebroker.TopicAlreadyFound	同一产品下Topic类名称重复。
iot.messagebroker.TopicTemplateCountExceedMax	产品的Topic类数量超过最大值。 一个产品最多可以定义50个Topic类。
iot.messagebroker.TopicTemplateIsNotFound	Topic类不存在。 可调用#unique_31查看产品的Topic类。
iot.messagebroker.UpdateDeviceShadowMessageFailed	更新设备影子失败。 请确认入参信息正确，然后重试。
iot.messagebroker.UpdateTopicTemplateException	更新Topic类过程发生异常。 请确认入参信息正确，然后重试。

错误码	描述
iot.messagebroker.UpdateTopicTemplateFailed	更新Topic类失败。 请确认入参信息正确，然后重试。

规则相关错误码

以`iot.rule`和`iot.ruleng`开头的错误码，及少量`iot.messagebroker`开头的错误码，是规则引擎相关错误码。

提示出现异常或失败时，请确认入参信息正确，然后重试。

错误码	描述
iot.rule.CreateRuleException	创建规则过程发生异常。 请确认入参信息正确，然后重试。
iot.rule.DeleteRuleFailed	删除规则失败。 请确认入参信息正确，然后重试。
iot.rule.IncorrentRuleActionId	规则动作ID错误。 可调用#unique_32查看规则动作ID。
iot.rule.IncorrentRuleActionType	规则动作类型错误。 规则动作类型参数`Type`支持可选值： · `DATAHUB`：DataHub · `ONS`：消息队列（RokectMQ） · `MNS`：消息服务 · `FC`：函数计算 · `OTS`：表格存储 **说明：** 数据格式为二进制的规则（即规则的`DataType`参数是`BINARY`）不支持转发数据至OTS（表格存储）。 · `REPUBLISH`：另一个物联网平台Topic。
iot.rule.IncorrentRuleId	规则ID错误。

错误码	描述
iot.rule.NullForwardDestForRule	转发数据目的地不能为空。 Configuration中的具体配置方法，请参见#unique_33。
iot.rule.NullSqlForRule	规则的SQL语句不能为空。
iot.rule.NotFoundRule	规则不存在。 请输入正确的规则ID（RuleId）。可调用#unique_34查看账号下所有规则的RuleId。
iot.rule.NotFoundRuleAction	规则动作不存在。 请输入正确的规则动作ID（ActionId）。可调用#unique_32查看某个规则下的所有ActionId。
iot.rule.ParseRuleActionConfigError	无法正常解析规则动作的配置。 请确认入参信息正确，然后重试。
iot.rule.QueryRuleActionListError	查询规则动作列表失败。 请确认入参信息正确，然后重试。
iot.rule.QueryRulePageActionError	分页获取规则列表失败。 请确认入参信息正确，然后重试。
iot.rule.RuleActionIsAlreadyCreated	已存在相同的规则动作。
iot.rule.RuleCountExceedMax	规则总数超过最大限制数。 单账号最多可以设置1000条规则。
iot.rule.RuleNameIsAlreadyExisted	规则名称已经存在。
iot.rule.StartRuleFailed	启动规则失败。 请确认入参信息正确，然后重试。
iot.rule.StopRuleFailed	停止规则失败。 请确认入参信息正确，然后重试。

错误码	描述
iot.rule.TooManyRuleAction	规则动作数量超过最大限制。 一条规则中转发数据的操作不能超过10个。
iot.rule.UpdateRuleFailed	更新规则失败。 请确认入参信息正确，然后重试。
iot.ruleng.CreateRuleActionFailed	创建规则动作失败。 请确认入参信息正确，然后重试。
iot.ruleng.DeleteRuleActionFailed	删除规则动作失败。 请确认入参信息正确，然后重试。
iot.ruleng.IncorrectType	应用规则的Topic类型错误。 TopicType支持的可选值： · 0：系统Topic · 1：自定义Topic · 2：设备状态消息Topic
iot.ruleng.IncorrectSysTopic	错误的系统Topic。 可在控制台设备详情页的Topic列表页签下查看正确的Topic。
iot.ruleng.InvalidRamRole	非法的RAM角色。 请登录RAM控制台查看角色信息。
iot.ruleng.QueryRuleActionFailed	获取规则动作失败。 请确认入参信息正确，然后重试。
iot.ruleng.RuleActionConfigurationIsNotJson	规则动作配置不是JSON格式。 参数Configuration的值必须是正确的JSON格式。具体请参见#unique_33。
iot.ruleng.RuleAlreadyIsStarted	规则是已启动状态。
iot.ruleng.NullRamRoleArn	roleArn不能为空。
iot.ruleng.NullRamRoleName	roleName不能为空。
iot.ruleng.NullRuleActionConfig	规则动作配置（参数Configuration）不能为空。

错误码	描述
iot.ruleng.NullRuleActionType	规则动作类型（参数Type）不能为空。
iot.messagebroker.IncorrectRuleSql	规则的SQL配置错误。 请根据#unique_35说明配置SQL。
iot.messagebroker.QueryRuleConfigActionException	获取规则配置信息过程出现异常。 请确认入参信息正确，然后重试。

以下表格分别列举消息转发目标设置失败的特有错误码。

表 2-1: 目标为REPUBLISH（另一个IoT Topic）的错误码

错误码	描述
iot.messagebroker.InvalidFormattedTopicName	Topic格式错误。 可在控制台设备详情页的Topic列表页签下查看正确的Topic格式。
iot.prod.NotExistedProduct	产品不存在。 请确认输入的ProductKey正确，并该产品属于当前阿里云账号。
iot.common.QueryProductActionError	查询产品失败。 请确认入参信息正确，然后重试。
iot.ruleng.IncorrectSysTopic	系统Topic错误。 可在控制台设备详情页的Topic列表页签下查看正确的Topic。
iot.messagebroker.NullTopicName	Topic名称不能为空。

表 2-2: 目标为DATAHUB（DataHub）的错误码

错误码	描述
iot.ruleng.IncorrectRegionName	regionName值错误。
iot.ruleng.NullProjectOfDatahub	DataHub的projectName不能为空。
iot.ruleng.NullTopicInDatahubProject	DataHub产品下的project中topicName不能为空。

错误码	描述
iot.ruleng.EmptySchemaNameOfTopic	目标DataHub Topic的Schema的名称name值不能为空。
iot.ruleng.EmptySchemaTypeOfTopic	目标DataHub Topic的Schema的类型type值不能为空。
iot.ruleng.EmptySchemaValueOfTopic	目标DataHub Topic的Schema值value不能为空。
iot.ruleng.NullOrEmptySchemaOfTopic	目标DataHub Topic的Schema不能为空。
iot.ruleng.NotFoundProjectInDataHub	DataHub中不存在此项目（project）。 请在DataHub中确认项目名称是否正确。
iot.ruleng.IncorrectSchemaValueOfTopic	目标DataHub Topic的Schema值错误。

表 2-3: 目标为OTS（表格存储）的错误码

错误码	描述
iot.ruleng.NullOtsInstanceName	表格存储的实例名称不能为空。
iot.ruleng.NullTableNameInOtsInstance	表格存储中实例的表名不能为空。
iot.ruleng.NullPrimaryKeyInOtsTable	表格存储中表的主键不能为空。
iot.ruleng.NullPrimaryKeyNameInOts	主键的名称不能为空。
iot.ruleng.NullPrimaryKeyValueInOts	主键的值不能为空。
iot.ruleng.IncorrectPrimaryKeyValueInOtsTable	表格存储中主键值错误。 请在表格存储中，查看您创建数据表时定义的主键。

表 2-4: 目标为MNS（消息服务）的错误码

错误码	描述
iot.ruleng.NullTopicNameInMns	消息服务中的主题不能为空。
iot.ruleng.NotFoundTopicInMns	消息服务中不存在此主题。 请在消息服务中，确认主题（Topic）名称。
iot.ruleng.QueryMnsTopicListActionError	获取消息服务主题列表失败。 请确认入参信息正确，然后重试。

表 2-5: 目标为FC（函数计算）的错误码

错误码	描述
iot.ruleng.NullServiceNameInFc	函数计算服务名称为空。
iot.ruleng.NullFunctionNameInFc	函数计算函数名称为空。
iot.ruleng.NotFoundServiceInFc	函数计算服务不存在。 请在函数计算中，确认正确的服务名称。

表 2-6: 目标为ONS（消息队列）的错误码

错误码	描述
iot.messagebroker.NullTopicName	消息队列中接收消息的Topic不能为空。

数据开发API相关错误码

错误码	描述
iot.dap.noServeJobExit	数据开发服务API对应的任务不存在。
iot.dap.serveApiPathRepetition	服务API接口地址重复，即传入ApiPath已存在。
iot.dap.serveApiInvalidParam	调用服务API的参数检查不通过。
iot.dap.serveApiPublishStatusError	请先通过测试后，再发布任务。
iot.dap.serveApiDeleteStatusError	已发布的任务不可删除。
iot.dap.serveApiPublishedNotEditable	已发布的任务不可编辑。
iot.dap.folderHasServeApiPublished	存在已发布的服务API，不可删除文件夹。
iot.dap.serveApiNoPublished	服务API不在发布状态，无法回滚。
iot.dap.duplicateTableNameError	资源表名称重复。
iot.dap.serveApiAlreadyPublished	服务API已发布。
iot.dap.serveApiPathIsEmpty	服务API接口地址不能为空。
iot.dap.serveApiSqlTemplateError	SQL模板信息异常，请校验并更新后，再尝试调用服务。
iot.dap.serveApiSqlInvokeParamError	SQL参数错误，类型与值不匹配。
iot.dap.syncStartPipelineError	任务启动失败。
iot.dap.methodTimeout	接口调用超时。

2.6 数据开发API管理

2.6.1 CreateDataAPIService

调用该接口创建数据算法服务API。

限制说明

- 单阿里云账号调用该接口的每秒请求数（QPS）最大限制为1。

  > 说明:
  > 子账号共享主账号配额。

- 单客户端出口IP的最大QPS限制为100，即来自单个客户端出口IP，调用阿里云接口的每秒请求总数不能超过100。

请求参数

参数	类型	是否必需	描述
Action	String	是	要执行的操作。取值`CreateDataAPIService`。
ApiPath	String	是	API调用地址的自定义部分。作为API资源标识符，需具有全局唯一性。 说明: API调用地址的前一段部分由系统生成。
DisplayName	String	是	API的显示名称，需具有全局唯一性。仅支持中文汉字、英文字母、数字、下划线（_）、连接符（-）、英文圆括号和空格，长度不超过20个字符。
FolderId	String	否	保存API的文件夹。 可在控制台数据开发页左侧导航栏，单击API服务右侧的+号，新建文件夹。详情请参见数据开发。 若不传入，默认文件夹为API列表。
Desc	String	否	API的描述。

参数	类型	是否必需	描述
OriginSql	String	是	API对应的原始SQL，指定数据开发的SQL样式。 例如`select count(*) as deviceCount from ${system.device} where status = 1`。其中，*${system.device}*是平台系统的设备表，具体请参见#unique_38中的表管理。
TemplateSql	String	是	服务的模板SQL，即原始SQL的模板化。 例如 `select count(*) as deviceCount from ${system.device} where status = ${status}`。其中，*${status}*是模板化的参数。支持设置模板参数为动态值。
RequestParams	List<RequestParam>	否	调用API的请求参数列表。 基于TemplateSql中的模板参数，配置相关的请求参数。例如，针对以上模板中的*${status}*，配置设备状态参数。 详情参见下表RequestParam。
ResponseParams	List<ResponseParam>	否	API的响应参数列表。 基于TemplateSql中的模板参数，配置select字段相关的出参。例如，针对以上模板中的`select count(*) as deviceCount`，配置设备数量相关出参。 详情参见下表ResponseParam。
公共请求参数	-	是	详情请参见#unique_39 。

表 2-7: RequestParam

参数	类型	是否必需	描述
Name	String	是	参数名称。 例如，*${status}*格式的模板参数，参数名称就是status。
Type	String	是	参数类型，请参见JDBCType。目前仅支持：ARRAY、VARCHAR、INTEGER、BIGINT、BOOLEAN、
Desc	String	否	参数描述。
Example	String	否	参数值示例。
Required	Boolean	否	该参数是否必填。 · true：必填。 · false：非必填。 默认值为true。

表 2-8: ResponseParam

参数	类型	是否必需	描述
Name	String	是	参数名称。
Type	String	是	参数类型，请参见JDBCType。目前仅支持：VARCHAR、INTEGER、BIGINT、BOOLEAN、DECIMA
Desc	String	否	参数描述。
Example	String	否	参数值示例。

返回参数

参数	类型	描述
RequestId	String	阿里云为该请求生成的唯一标识符。

参数	类型	描述
Success	Boolean	是否调用成功。true表示调用成功，false表示调用失败。
ErrorMessage	String	调用失败时，返回的出错信息。
Code	String	调用失败时，返回的错误码。错误码详情，请参见#unique_40。
Data	Data	调用成功时，返回注册的设备信息。详情参见下表Data。

表 2-9: Data

参数	类型	描述
ApiSrn	String	API资源标识符，API的全局唯一标识。 示例： `acs:iot:*:1271039834 61****:serveapi/device/getDeviceCountByStatus2` 以上示例中的信息说明如下： · `1271039834 61****`是阿里云主账号ID。 · `/device/getDeviceCountByStatus`是请求参数`ApiPath`的值，即API调用地址的自定义部分。
CreateTime	Long	API的创建时间。
LastUpdateTime	Long	API的最后更新时间。

示例

请求示例

```
https://iot.cn-shanghai.aliyuncs.com/?Action=CreateDataAPIService
&ApiPath=%2Fdevice%2FgetDeviceCountByStatus
```

```
&DisplayName=%E6%A0%B9%E6%8D%AE%E7%8A%B6%E6%80%81%E6%9F%A5%E8%AF%A2%E8
%AE%BE%E5%A4%87%E6%80%BB%E6%95%B0%E6%8E%A5%E5%8F%A3
&OriginSql=SELECT%20COUNT%28%2A%29%20FROM%20%24%7Bsystem.device%7D%
20WHERE%20status%20%3D%201
&RequestParam.1.Desc=%E8%AE%BE%E5%A4%87%E7%8A%B6%E6%80%81
&RequestParam.1.Example=0%EF%BC%9A%E6%9C%AA%E6%BF%80%E6%B4%BB%EF%BC%
8C1%EF%BC%9A%E5%9C%A8%E7%BA%BF%EF%BC%8C3%EF%BC%9A%E7%A6%BB%E7%BA%BF%EF
%BC%8C%208%EF%BC%9A%E5%B7%B2%E7%A6%81%E6%AD%A2
&RequestParam.1.Name=status
&RequestParam.1.Required=true
&RequestParam.1.Type=VARCHAR
&ResponseParam.1.Desc=%E8%AE%BE%E5%A4%87%E6%95%B0
&ResponseParam.1.Example=100
&ResponseParam.1.Name=deviceCount
&ResponseParam.1.Required=true
&ResponseParam.1.Type=INTEGER
&TemplateSql=SELECT%20COUNT%28%2A%29%20as%20deviceCount%20FROM%20%24%
7Bsystem.device%7D%20WHERE%20status%20%3D%20%24%7Bstatus%7D&Timestamp=
2019-06-10T10%3A00%3A05Z
&公共请求参数
```

返回示例

- **JSON格式**

```
{
    "RequestId": "57b144cf-09fc-4916-a272-a62902d5b207",
    "Success": true,
    "Data": {
        "ApiSrn": "acs:iot:*:127103983461****:serveapi/device/
getDeviceCountByStatus2",
        "CreateTime": 1557839468865,
        "LastUpdateTime": 1557839468865
    }
}
```

- **XML格式**

```
<?xml version="1.0" encoding="UTF-8" ?>
<CreateDataAPIServiceResponse>
    <RequestId>57b144cf-09fc-4916-a272-a62902d5b207</RequestId>
    <Success>true</Success>
    <Data>
        <ApiSrn>acs:iot:*:127103983461****:serveapi/device/
getDeviceCountByStatus2</ApiSrn>
        <CreateTime>1557839468865</CreateTime>
        <LastUpdateTime>1557839468865</LastUpdateTime>
    </Data>
```

```
</CreateDataAPIServiceResponse>
```

2.6.2 GetDataAPIServiceDetail

调用该接口获取数据算法服务API详情。

限制说明

- 单阿里云账号调用该接口的每秒请求数（QPS）最大限制为1。

  说明:
  子账号共享主账号配额。

- 单客户端出口IP的最大QPS限制为100，即来自单个客户端出口IP，调用阿里云接口的每秒请求总数不能超过100。

请求参数

参数	类型	是否必需	描述
Action	String	是	要执行的操作。取值：GetDataAPIServiceDetail。

ApiSrn	String	是	API资源标识符，API的全局唯一标识。 调用#unique_42成功创建API，返回的ApiSrn值。
			格式：
			`acs:iot:*:${aliyunuserID}:serveapi/${ApiPath}`
			示例：
			`acs:iot:*:127103983461****:serveapi/device/getDeviceCountByStatus2`
			以上示例中的信息说明如下：
			· 127103983461****是阿里云主账号ID。
			· /device/getDeviceCountByStatus是API调用地址的自定义部分。

公共请求参数	-	是	详情参见#unique_39 。

返回参数

参数	类型	描述
RequestId	String	阿里云为该请求生成的唯一标识符。
Success	Boolean	是否调用成功。true表示调用成功，false表示调用失败。
ErrorMessage	String	调用失败时，返回的出错信息。
Code	String	调用失败时，返回的错误码。错误码详情，请参见#unique_40。
Data	Data	调用成功时，返回的数据。详情参见下表Data。

表 2-10: Data

参数	类型	描述
ApiSrn	String	API资源标识符，API的全局唯一标识。
Status	Integer	API的状态。 · 0：可编辑。 · 1：已测试。 · 2：已发布。
DisplayName	String	API名称。

参数	类型	描述
ApiPath	String	API调用地址的自定义部分。
CreateTime	Long	API的创建时间。
LastUpdateTime	Long	API的最后更新时间。
Description	String	API的描述信息。
SqlTemplateDTO	String	SQL模板信息。 调用成功时，返回的SQL模板数据。详情参见下表SqlTemplateDTO。

表 2-11: SqlTemplateDTO

参数	类型	描述
OriginSql	String	API对应的原始SQL。
TemplateSql	String	原始SQL的模板化SQL。

参数	类型	描述
RequestParams	List<RequestParam>	调用API的请求参数列表。 详情参见下表RequestParam。
ResponseParams	List<ResponseParam>	API的响应参数列表。 详情参见下表ResponseParam。

表 2-12: RequestParam

参数	类型	是否必需	描述
Name	String	是	请求参数名称。
Type	String	是	参数类型，请参见JDBCType。目前仅支持：ARRAY、VARCHAR、INTEGER、BIGINT、BOOLEAN、
Desc	String	否	参数描述。
Example	String	否	参数值示例。
Required	Boolean	否	该参数是否必填。 · true：必填。 · false：非必填。 默认值为true。

表 2-13: ResponseParam

参数	类型	是否必需	描述
Name	String	是	返回参数名称。

参数	类型	是否必需	描述
Type	String	是	参数类型，请参见JDBCType。目前仅支持：VARCHAR、INTEGER、BIGINT、BOOLEAN、DECIMA
Desc	String	否	参数描述。
Example	String	否	参数值示例。

示例

请求示例

```
https://iot.cn-shanghai.aliyuncs.com/?Action=GetDataAPIServiceDetail
&ApiSrn=acs:iot:*:127103983461****:serveapi/device/getDeviceC
ountByStatus2
&公共请求参数
```

返回示例

- JSON格式

```
{
   "RequestId":"57b144cf-09fc-4916-a272-a62902d5b207",
   "Success":true,
   "Data":{
      "ApiSrn":"acs:iot:*:127103983461****:serveapi/device/
getDeviceCountByStatus2",
      "CreateTime":1557839468865,
      "LastUpdateTime":1557839468865,
      "Status": 1,
      "DisplayName":"根据状态获取设备数",
      "ApiPath":"/device/getDeviceCountByStatus",
      "Description":"描述",
      "SqlTemplateDTO":{
         "OriginSql":"SELECT COUNT(*) FROM ${system.device} WHERE
status = 1",
         "TemplateSql":"SELECT COUNT(*) as deviceCount FROM ${system
.device} WHERE status = ${status}",
         "RequestParams":[
            {
               "name":"status",
               "type":"INTEGER",
               "desc":"设备状态",
               "example":"0",
               "required":true
            }
         ],
         "ResponseParams":[
            {
               "name":"deviceCount",
               "type":"INTEGER",
               "desc":"设备数",
               "example":"100"
            }
         ]
      }
   }
```

```
}
```

- XML格式

```
<?xml version="1.0" encoding="UTF-8" ?>
<GetDataAPIServiceDetailResponse>
    <RequestId>57b144cf-09fc-4916-a272-a62902d5b207</RequestId>
    <Success>true</Success>
    <Data>
        <ApiSrn>acs:iot:*:127103983461****:serveapi/device/
getDeviceCountByStatus2</ApiSrn>
        <CreateTime>1557839468865</CreateTime>
        <LastUpdateTime>1557839468865</LastUpdateTime>
        <Status>1</Status>
        <DisplayName>根据状态获取设备数</DisplayName>
        <ApiPath>/device/getDeviceCountByStatus</ApiPath>
        <Description>描述</Description>
        <SqlTemplateDTO>
            <OriginSql>SELECT COUNT(*) FROM ${system.device} WHERE
status = 1</OriginSql>
            <TemplateSql>SELECT COUNT(*) as deviceCount FROM ${
system.device} WHERE status = ${status}</TemplateSql>
            <RequestParams>
                <name>status</name>
                <type>INTEGER</type>
                <desc>设备状态</desc>
                <example>0</example>
                <required>true</required>
            </RequestParams>
            <ResponseParams>
                <name>deviceCount</name>
                <type>INTEGER</type>
                <desc>设备数</desc>
                <example>100</example>
            </ResponseParams>
        </SqlTemplateDTO>
    </Data>
</GetDataAPIServiceDetailResponse>
```

2.6.3 InvokeDataAPIService

调用该接口调用数据算法服务API，获取SQL查询结果。

限制说明

- 单阿里云账号调用该接口的每秒请求数（QPS）最大限制为1。

  说明:

  子账号共享主账号配额。

- 单客户端出口IP的最大QPS限制为100，即来自单个客户端出口IP，调用阿里云接口的每秒请求总数不能超过100。

请求参数

参数	类型	是否必需	描述
Action	String	是	要执行的操作。取值InvokeDataAPIService。
ApiSrn	String	是	API资源标识符，API的全局唯一标识。调用#unique_42成功创建API，返回的ApiSrn值。 示例： `acs:iot:*:127103983461****:serveapi/device/getDeviceCountByStatus2` 以上示例中的信息说明如下： · 127103983461****是阿里云主账号ID。 · /device/getDeviceCountByStatus是请求参数ApiPath的值，即API调用地址的自定义部分。
Params	List<Param>	是	调用API的请求参数，需根据您调用#unique_42创建API时，定义的RequestParam传入请求参数。详情请参见下表Param。

表 2-14: Param

参数	类型	是否必需	描述
ParamName	String	是	调用API的入参参数名称。必须与调用#unique_42创建API时，RequestParam中定义的Name保持一致。
ParamValue	String	否	调用API的入参参数值。 说明： · 统一使用String类型存储，物联网平台会根据创建API时定义的ParamType转换成JDBC类型对象。 · 创建API时，如果API请求参数类型Type定义为ARRAY类型，则不传入该参数，而需传入ListParamType和ListParamValue。

参数	类型	是否必需	描述
ListParamValues	List<String>	否	ARRAY类型的参数值列表。 说明: 统一使用String类型存储，平台会跟据ListParamType对应的值转换成JDBC类型对象。
ListParamType	String	否	ARRAY类型的参数值的数据类型。请参见JDBCType。 目前仅支持VARCHAR、INTEGER、BIGINT、BOOLEAN、DECIMAL、TIMESTAMP。

返回参数

参数	类型	描述
RequestId	String	阿里云为该请求生成的唯一标识符。
Success	Boolean	是否调用成功。true表示调用成功，false表示调用失败。
ErrorMessage	String	调用失败时，返回的出错信息。
Code	String	调用失败时，返回的错误码。错误码详情，请参见#unique_40。
Data	Data	调用成功时，返回注册的设备信息。详情参见下表Data。

表 2-15: Data

参数	类型	描述
ApiSrn	String	API资源标识符，API的全局唯一标识。 示例： `acs:iot:*:1271039834 61****:serveapi/device/getDeviceCountByStatus2` 以上示例中的信息说明如下： · `1271039834 61****`是阿里云主账号ID。 · `/device/getDeviceCountByStatus`是请求参数`ApiPath`的值，即API调用地址的自定义部分。

参数	类型	描述
PageNo	Integer	显示的查询结果的页码。分页码从0开始，默认为0。 如果您要自定义显示结果页，建议您在请求入参Params中增加自定义参数，如pageNo。
PageSize	Integer	每页显示的查询结果记录数。 如果您要自定义每页显示的记录数，建议您在请求入参Params中增加自定义参数，如pageSize。
FieldNameList	List<String>	结果字段列表。 列表元素即调用#unique_42创建API时，ResponseParam中的Name定义的参数名称。

参数	类型	描述
ResultList	List<Map<String, Object>>	返回的SQL处理结果。 根据调用#unique_42创建API时，ResponseParam中的Name参数，返回处理结果。 列表元素Map<String, Object>说明如下： · key是String类型，是Name定义的参数名称。 · Object是参数对应的值，其数据类型与ResponseParam中的Type一致。

示例

请求示例

```
https://iot.cn-shanghai.aliyuncs.com/?Action=InvokeDataAPIService
&ApiSrn=acs:iot:*:127103983461****:serveapi/device/getDeviceC
ountByStatus2
&Param.1.ParamName=status
&Param.1.ParamValue=1
&公共请求参数
```

返回示例

· JSON格式

```
{
```

```
  "Data": {
    "ResultList": {
      "ResultList": [{
          "deviceCount": 47
        }]
    },
    "PageSize": 1,
    "PageNo": 0,
    "ApiSrn": "acs:iot:*:127103983461****:serveapi/device/getDeviceC
ountByStatus2",
    "FieldNameList": {
      "FieldNameList": ["deviceCount"]
    }
  },
  "RequestId": "E68FE5DC-4D7B-4987-B785-DF8C6F191F5D",
  "Success": true
}
```

- **XML格式**

```
<?xml version="1.0" encoding="UTF-8" ?>
<InvokeDataAPIServiceResponse>
    <Data>
        <ResultList>
            <ResultList>
                <deviceCount>47</deviceCount>
            </ResultList>
        </ResultList>
        <PageSize>1</PageSize>
        <PageNo>0</PageNo>
        <ApiSrn>acs:iot:*:127103983461****:serveapi/device/
getDeviceCountByStatus2</ApiSrn>
        <FieldNameList>
            <FieldNameList>deviceCount</FieldNameList>
        </FieldNameList>
    </Data>
    <RequestId>E68FE5DC-4D7B-4987-B785-DF8C6F191F5D</RequestId>
    <Success>true</Success>
</InvokeDataAPIServiceResponse>
```