阿里云物联网套件

云端开发指南

为了无法计算的价值 | 阿里云

云端开发指南

总体说明

什么情况使用服务器端API

当您需要对接自己的业务系统时，您可以使用服务器端API与云端互通，比如通过您的服务器发消息给您自己的设备等。

IoT套件同时提供了几种较为普遍的语言对应的SDK，比如JAVA、PHP、Python、.NET

调用方式

概述

对API接口调用是通过向API的服务端地址发送HTTP GET/POST请求，并按照接口说明在请求中加入相应请求参数来完成的；根据请求的处理情况，系统会返回处理结果。

1. 请求结构
2. 公共参数
3. 返回结果
4. 签名机制

请求结构

服务地址

IoT API的服务接入地址为：

华东2：iot.cn-shanghai.aliyuncs.com

新加坡： iot.ap-southeast-1.aliyuncs.com

美国西部1：iot.us-west-1.aliyuncs.com

杭州：iot.aliyuncs.com

通信协议

支持通过HTTP或HTTPS通道进行请求通信。为了获得更高的安全性，推荐您使用HTTPS通道发送请求。

请求方法

支持HTTP GET方法发送请求，这种方式下请求参数需要包含在请求的URL中，如果是get需要对参数进行urlencode。也可以使用POST。

请求参数

每个请求都需要指定要执行的操作，即Action参数（例如Sub），以及每个操作都需要包含的公共请求参数和指定操作所特有的请求参数。

字符编码

请求及返回结果都使用UTF-8字符集进行编码。

公共参数

公共请求参数

公共请求参数是指每个接口都需要使用到的请求参数。

名称	类型	是否必须	描述

Format	String	否	返回值的类型，支持JSON与XML。默认为XML
Version	String	是	API版本号，为日期形式：YYYY-MM-DD，最新版本为2017-04-20 ，每个接口可以存在多个版本
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID
Signature	String	是	签名结果串，关于签名的计算方法，请参见签名机制。
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（为北京时间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。用户在不同请求间要使用不同的随机数值
RegionId	String	是	取值 cn-shanghai

示例

```
https://iot.cn-shanghai.aliyuncs.com/
?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
```

公共返回参数

用户发送的每次接口调用请求，无论成功与否，系统都会返回一个唯一识别码RequestId给用户。

示例

XML示例

```
<?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示例

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

返回结果

调用API服务后返回数据采用统一格式，返回的HTTP状态码为2xx，代表调用成功；返回4xx或5xx的HTTP状态码代表调用失败。调用成功返回的数据格式主要有XML和JSON两种，外部系统可以在请求时传入参数来制定返回的数据格式，默认为XML格式。本文档中的返回示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

成功结果

XML示例

```
<?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示例

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

错误结果

调用接口出错后，将不会返回结果数据。调用方可根据每个接口对应的错误码以及下述2.3.3的公共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和一个您该次请求访问的站点ID：HostId。在调用方找不到错误原因时，可以联系阿里云客服，并提供该HostId和RequestId，以便我们尽快帮您解决问题。

XML示例

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

JSON示例

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

公共错误码

签名机制

IoT套件会对每个访问的请求进行身份验证，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。

签名需要您在控制台 Access Key 管理 页面获得 Access Key ID 和 Access Key Secret，进行对称加密来验证请求的发送者身份。其中，Access Key ID 用于标识访问者身份；Access Key Secret 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。

注意：IoT套件提供了Java、Python、PHP语言的服务端SDK，可以免去您对签名算法进行编码的麻烦。

签名操作

您在访问时，需要按照下面的方法对请求进行签名处理。

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

按参数名进行排序。

按照参数名称的字典顺序对请求中所有的请求参数进行排序，包括“公共请求参数”（不包括 Signature 参数）和接口的自定义参数。

注意：当使用 GET 方法提交请求时，这些参数就是请求 URI 中的参数部分（即 URI 中“?”之后由“&”连接的部分）。

对参数名称和参数值进行 URL 编码。

名称和值要使用 UTF-8 字符集进行 URL 编码。URL 编码的规则如下：

i. 字符 A~Z、a~z、0~9 以及字符“-”、“_”、“.”、“~”不编码；
ii. 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。比如英文的双引号（”）对应的编码为 %22；
iii. 对于扩展的 UTF-8 字符，编码成 %XY%ZA... 的格式；
iv. 英文空格（ ）要编码成 %20，而不是加号（+）。

注意：一般支持URL编码的库（比如 Java 中的 java.net.URLEncoder）都是按照“application/x-www-form-urlencoded”的 MIME 类型的规则进行编码的。实现时可以直接使用这类方式进行编码，把编码后的字符串中加号（+）替换成 %20、星号（*）替换成 %2A、%7E 替换回波浪号（~），即可得到上述规则描述的编码字符串。

对编码后的参数名称和值使用英文等号（=）进行连接。

按参数名称的字典顺序，把英文等号连接得到字符串依次使用 & 符号连接，即得到规范化请求字符串。

构造被签名字符串。

基于步骤 1 得到的规范化字符串构造用于计算签名的字符串，可参考如下规则：

```
 StringToSign=
HTTPMethod + “&” +
percentEncode(“/”) + ”&” +
percentEncode(CanonicalizedQueryString)
```

其中：

- HTTPMethod 是提交请求用的 HTTP 方法，比如 GET。
- percentEncode(“/”) 是按照步骤 1.i 中描述的 URL 编码规则对字符 “/” 进行编码得到的值，即 %2F。
- percentEncode(CanonicalizedQueryString) 是对步骤 1 中构造的规范化请求字符串按步骤 1.ii 中描述的 URL 编码规则编码后得到的字符串。

计算 HMAC 值。

按照 RFC2104 的定义，使用步骤 2 得到的字符串计算签名 HMAC 值。

> **注意**：计算签名时使用的 Key 就是您持有的 Access Key Secret 并加上一个 “&” 字符（ASCII:38），使用的哈希算法是 SHA1。

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）。

添加签名。

将得到的签名值作为 Signature 参数添加到请求参数中，即完成对请求签名的过程。

> **注意**：得到的签名值在作为最后的请求参数值提交给服务器时，要和其它参数一样，按照 RFC3986 的规则进行 URL 编码。

示例

以Pub接口为例，假设使用的Access Key Id为testid，Access Key Secret为testsecret，那么签名前的请求

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&SignatureMethod=HMAC-SHA1&RegionId=cn-shanghai&ProductKey=12345abcdeZ&TopicFullName=%2FproductKey%2Ftestdevice%2Fget
```

计算得到的待签名字符串StringToSign为：

```
GET&%2F&AccessKeyId%3Dtestid%26Action%3DPub%26Format%3DXML%26MessageContent%3DaGVsbG93b3JsZA%253D%26ProductKey%3D12345abcdeZ%26Qos%3D0%26RegionId%3Dcn-shanghai%26ServiceCode%3Diot%26SignatureMethod%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
```

因为Access Key Secret为testsecret，所以用于计算HMAC的Key为testsecret&，计算得到的签名值为：

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

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 示例代码

本示例不需要依赖第三方的库包，可以直接使用，签名步骤如下所示。

构造规范化的请求字符串（排序及URL编码）。

```
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;
}

/** 对参数名称和参数值进行URL编码**/
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");

}
```

构造成待签名的字符串。

```
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.generateQueryString(
sortParameter, true);
if (null == httpMethod) {
throw new RuntimeException("httpMethod can not be empty");
}

/** 构造待签名的字符串* */
StringBuilder stringToSign = new StringBuilder();
stringToSign.append(httpMethod).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
stringToSign.append(percentEncode(canonicalizedQueryString));

return stringToSign.toString();
}
```

计算待签名字符串的 HMAC 值。

```
public static byte[] hmacSHA1Signature(String secret, String baseString)
throws Exception {
if (isEmpty(secret)) {
throw new IOException("secret can not be empty");
}
if (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));
}

private static boolean isEmpty(String str) {
return (str == null || str.length() == 0);
}
```

按照 Base64 编码规则编码成字符串，获取签名值。

```
 public static String newStringByBase64(byte[] bytes)
throws UnsupportedEncodingException {
if (bytes == null || bytes.length == 0) {
return null;
}
return new String(new BASE64Encoder().encode(bytes));
}
public static String composeStringToSign(Map<String, String> queries) {
String[] sortedKeys = (String[]) queries.keySet()
.toArray(new String[0]);
Arrays.sort(sortedKeys);
StringBuilder canonicalizedQueryString = new StringBuilder();
for (String key : sortedKeys) { canonicalizedQueryString.append("&").append(percentEncode(key))
.append("=")
.append(percentEncode((String) queries.get(key)));
}
StringBuilder stringToSign = new StringBuilder();
stringToSign.append("GET");
stringToSign.append("&");
stringToSign.append(percentEncode("/"));
stringToSign.append("&"); stringToSign.append(percentEncode(canonicalizedQueryString.toString()
.substring(1)));
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 "";
}
/**
* get SignatureNonce
** */
public static String getUniqueNonce() {
UUID uuid = UUID.randomUUID();
return uuid.toString();
}
/**
* get timestamp
```

```
**/
public static String getISO8601Time() {
Date nowDate = new Date();
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
df.setTimeZone(new SimpleTimeZone(0, "GMT"));
return df.format(nowDate);
}
```

添加签名。

```
 public static String composeUrl(String endpoint, Map<String, String> queries)
throws UnsupportedEncodingException {
Map<String, String> mapQueries = queries;
StringBuilder urlBuilder = new StringBuilder("");
urlBuilder.append("http");
urlBuilder.append("://").append(endpoint);
if (-1 == urlBuilder.indexOf("?")) {
urlBuilder.append("/?");
}
urlBuilder.append(concatQueryString(mapQueries));
return urlBuilder.toString();
}
public static String concatQueryString(Map<String, String> parameters)
throws UnsupportedEncodingException {
if (null == parameters) {
return null;
}
StringBuilder urlBuilder = new StringBuilder("");
for (Map.Entry<String, String> entry : parameters.entrySet()) {
String key = (String) entry.getKey();
String val = (String) entry.getValue();
urlBuilder.append(encode(key));
if (val != null) {
urlBuilder.append("=").append(encode(val));
}
urlBuilder.append("&");
}
int strIndex = urlBuilder.length();
if (parameters.size() > 0) {
urlBuilder.deleteCharAt(strIndex - 1);
}
return urlBuilder.toString();
}
public static String encode(String value)
throws UnsupportedEncodingException {
return URLEncoder.encode(value, "UTF-8");
}
}
```

SDK下载

SDK下载

- JAVA SDK
- Python SDK
- PHP SDK
- .NET SDK

SDK Demo下载

- SDK Demo

SDK Demo包含JAVA、Python、PHP、.NET版本

SDK版本更迭

2017-12-08

- 华东2节点发布.NET版本的SDK

2017-12-01

- 新增设备自定义属性接口

2017-07-26

- 新增基于MQTT协议的RRPC接口，可以发消息给设备并同步返回响应

2017-05-25

- 新增设备影子、发送广播相关的接口

2016-11-18

- 升级SDK版本到2.1.0
- 新增产品创建、设备状态查询等接口

2016-08-15

- 升级SDK版本到2.0.3
- 增加服务端订阅者上线API
- 增加Publish消息Qos特性

2016-01-26

- 增加取消订阅topic接口
- 增加推送消息至设备接口
- 增加推送消息并获取设备结果接口
- 增加针对Topic或者设备的授权接口

SDK使用参考

使用说明-java

JAVA SDK使用说明

公共参数

名称	类型	是否必须	描述
accessKeyId	String	必须	阿里云的Access Key ID
accessKeySecret	String	必须	阿里云的Access Key Secret

- accessKeyId和accessKeySecret 在阿里云官网控制台获取 https://ak-console.aliyun.com/#/accesskey
- 各个方法中需要用到 ProductKey 在IoT控制台 查看ProductKey数据.

引入SDK

*1 Maven坐标：

```
<!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-iot -->
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-iot</artifactId>
<version>4.0.0</version>
</dependency>
```

依赖公共包：

```
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>3.2.10</version>
</dependency>
```

*2 SDK源码地址参考 - github https://github.com/aliyun/aliyun-openapi-java-sdk/tree/master/aliyun-java-sdk-iot

初始化

```
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
```

发起调用

以发布消息到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
PubResponse response = client.getAcsResponse(request);
System.out.println(response.getSuccess());
```

```
System.out.println(response.getErrorMessage());
```

使用说明-python

Python SDK使用说明

公共参数

名称	类型	是否必须	描述
accessKeyId	String	必须	阿里云的Access Key ID
accessKeySecret	String	必须	阿里云的Access Key Secret

- accessKeyId和accessKeySecret 在阿里云官网控制台获取 https://ak-console.aliyun.com/#/accesskey
- 各个方法中需要用到 ProductKey 在IoT控制台 查看ProductKey数据.

引入Python SDK

安装Python SDK包：

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

在Python文件中引入Python SDK相关文件：

```
from aliyunsdkcore import client
from aliyunsdkiot.request.v20170420 import RegistDeviceRequest
from aliyunsdkiot.request.v20170420 import PubRequest
...
```

初始化.

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

发起调用

以publish数据到设备为例:

```
request = PubRequest.PubRequest()
request.set_accept_format('json') #设置返回数据格式，默认为XML
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
```

使用说明-php

PHP SDK使用说明

公共参数

名称	类型	是否必须	描述
accessKeyId	String	必须	阿里云的Access Key ID
accessKeySecret	String	必须	阿里云的Access Key Secret

- accessKeyId和accessKeySecret 在阿里云官网控制台获取 https://ak-console.aliyun.com/#/accesskey
- 各个方法中需要用到 ProductKey 在IoT控制台 查看ProductKey数据.

引入PHP SDK

- 下载PHP CORE SDK 代码 PHP-SDK
- 在PHP文件中包含其头文件（注意文件路径要正确），如：

```
<?php
include_once '../aliyun-php-sdk-core/Config.php';
//Do something below
...
```

初始化

```
<?php
include_once 'aliyun-php-sdk-core/Config.php';
use \Iot\Request\V20170420 as Iot;
//设置你的AccessKeyId/AccessSecret/ProductKey
$accessKeyId = "";
$accessSecret = "";
$iClientProfile = DefaultProfile::getProfile("cn-shanghai", $accessKeyId, $accessSecret);
$client = new DefaultAcsClient($iClientProfile);
```

发起调用

以publish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);
```

使用说明-.net

.NET SDK使用说明

公共参数

名称	类型	是否必须	描述
accessKeyId	String	是	阿里云的Access Key ID
accessKeySecret	String	是	阿里云的Access Key Secret

- accessKeyId和accessKeySecret可以从阿里云控制台官网查询。
- 各个方法中用到的productKey和deviceName可以从Iot控制台查询。

下载IoT .NET SDK

- .NET SDK

引入.NET SDK DLL

- 引入阿里云.NET SDK核心库。从.NET SDK发布列表 中下载SDK核心库的压缩包，解压后获得DLL文件。
- 添加核心库的引用。在visual studio的 *解决方案资源管理器* 中，选中您的项目，右键 *引用*，在弹出的菜单中选择 *添加引用*，在弹出的添加引用菜单中，选择 *浏览*，选择之前下载并解压好的DLL文件，点击**确认**即可。

初始化

```
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;

IClientProfile clientProfile = DefaultProfile.GetProfile("<your-region-id>", "<your-access-key-id>", "<your-access-key-secret>");
DefaultAcsClient client = new DefaultAcsClient(clientProfile);
```

发起调用

以向topic发布消息为例

```
PubRequest request = new PubRequest();
request.ProductKey = "<productKey>";
request.TopicFullName = "/" + request.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);
}
```