# 阿里云

## NodeJS SDK

文档版本：20200514

# 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：**<br>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：**<br>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：**<br>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：**<br>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<br>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 环境要求与配置

> 注：目前的JS版本的Link SDK最新版本号为1.2.8。

安装 Node.js 运行环境，版本需要 `>=4.0.0` 。

**通过 npm 包管理工具安装SDK**

**将SDK安装到Nodejs项目所在目录**

适用于开发者已创建自己的项目，然后集成阿里云的Link SDK：

```
npm install alibabacloud-iot-device-sdk --save
```

**将SDK进行全局安装**

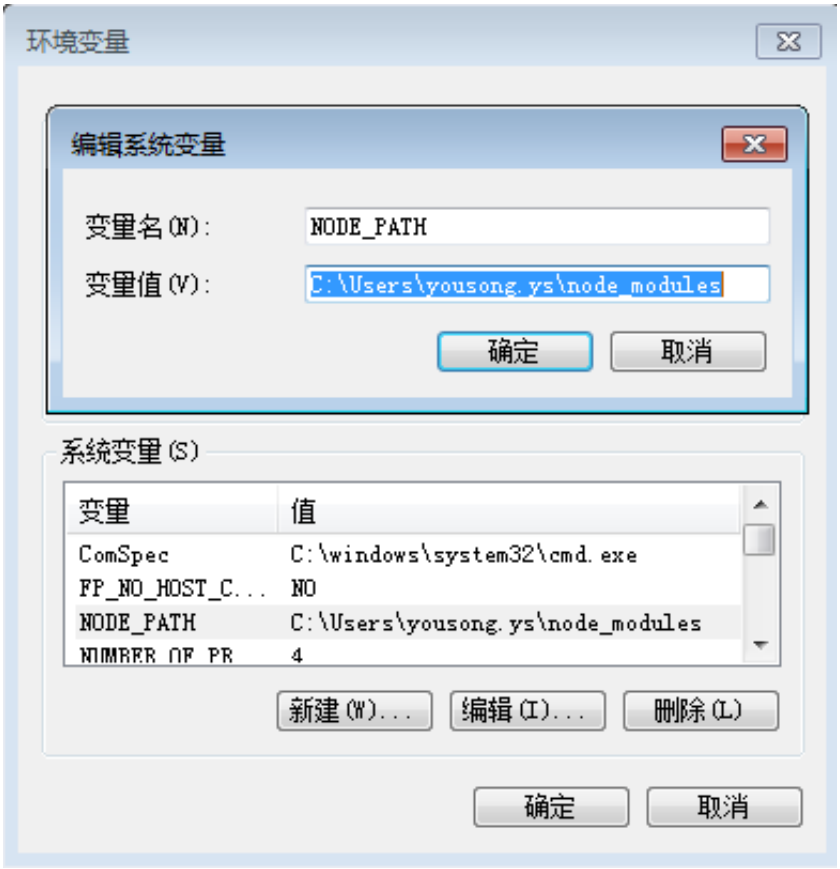
适用于开发者直接编写了一个js文件（该文件引用了阿里云的Link SDK）、然后直接使用node xxx.js运行该js程序的场景。这种情况需要将SDK进行全局安装，命令如下所示：

```
npm install -g alibabacloud-iot-device-sdk
```

注：不推荐全局安装的方式，因为扩展性不好。

**全局安装后运行程序常见错误：**

- 如果在**windows**下运行js程序时提示“Error: Cannot find module 'alibabacloud-iot-device-sdk'”，请在环境变量中增加"NODE_PATH"变量，其值设置为SDK安装所在的目录。如下图所示：




注：上图中的“NODE_PATH”的变量值需要修改为开发者自己设备上SDK安装所在的目录。

- 如果在**Linux**下直接使用node 运行指定的js程序时，提示“Error: Cannot find module 'alibabacloud-iot-device-sdk'”，也是由于没有设置NODE_PATH变量引起，可以运行命令 `npm root -g` 获取npm的模块安装目录，并将该目录设置到NODE_PATH环境变量并生效（比如重启Linux）后再次使用node加载js程序。

**源码下载**

目前js版本sdk已经开源，项目地址：https://github.com/aliyun/alibabacloud-iot-device-sdk 。

**如何引用Link SDK**

**node程序引用SDK**

使用SDK提供的API需要在js代码中加入对SDK的依赖，如下所示：

```
const iot = require('alibabacloud-iot-device-sdk');
```

**浏览器、微信小程序、支付宝小程序集成SDK**

```
const iot = require('./dist/alibabacloud-iot-device-sdk.js'); //引用路径以实际为准
```

```
// 支付宝小程序请使用./dist/alibabacloud-iot-device-sdk-1.2.8-alimin-compatible.js
```

**说明:**

支付宝最新版本容器升级后导致支付宝小程序无法使用 'alibabacloud-iot-device-sdk.js' ，请使用支付宝定制版本 'alibabacloud-iot-device-sdk-1.2.8-alimin-compatible.js', 该文件位于dist目录下

**JS版本下载地址**

github下载：https://github.com/aliyun/alibabacloud-iot-device-sdk/tree/master/dist

cdn下载：

alibabacloud-iot-device-sdk.js 下载地址 https://unpkg.com/alibabacloud-iot-device-sdk@1.2.4/dist/alibabacloud-iot-device-sdk.js

alibabacloud-iot-device-sdk.min.js 下载地址 https://unpkg.com/alibabacloud-iot-device-sdk@1.2.4/dist/alibabacloud-iot-device-sdk.min.js

**版本变更**

记录各个版本的主要变动

**1.2.8**

解决支付宝最新版本容器升级后导致支付宝小程序无法使用的问题

**1.2.7**

- 修改了事件上报、属性上报中给出参数的定义
- 增加onProps对属性设置进行处理
- 包名的修改，从 aliyun-iot-device-sdk 正式改名为 alibabacloud-iot-device-sdk
- 增加对微信小程序、支付宝小程序、浏览器的支持
- 增加onService中reply函数,并支持同步和异步调用
- 增加onConfig方法用于订阅云端远程配置更新
- 增加部分功能的example
- 重写了网关子设备subdevice的实现

**1.0.1**

- 增加设备标签上报功能
- 增加删除标签功能
- 增加了设备动态注册功能

- 增加设备影子相关功能
- 增加获取设备配置功能

# 2 认证与连接

设备的身份认证支持两种方法，不同方法在填写不同信息。

- 若使用一机一密认证方式，设备上需要烧写product_key、device_name和device_secret
- 若使用一型一密认证方式，设备上需要烧写product_key、和product_secret，并且厂家需要为每个设备设置一个唯一标识（比如SN、MAC地址），这个唯一标识将被用来作为device_name使用，由于product_key和product_secret对于一个产品来说是固定的数值，烧写的时候是固定的，因此称为一型一密。

注：

- 阿里云IoT的一型一密，是通过一个动态注册过程去获取设备的device_secret，然后再使用product_key、device_name、device_secret去连接阿里云物联网平台并对设备进行认证，本质上仍然是一机一密
- 动态注册过程只能执行一次，也即当设备获取到device_secret之后，设备需要将其存入NVRAM/FLASH，当重启或者再次连接阿里云物联网平台时，设备需要判断如果设备已经获取到了device_secret，那么不需要再次通过动态注册去获取device_secret，而是直接使用product_key、device_name、device_secret去连接阿里云物联网平台
- 如果一个设备已经成功的通过动态注册过程获取到了device_secret，再次调用动态注册接口将会返回错误

**认证与连接API描述**

| **API原型** | **iot.device(options)** |
| --- | --- |
| 功能描述 | 创建一个设备实例，并连接阿里云IoT |
| 入参 | options：- productKey (String)- deviceName (String)- deviceSecret (String)- region (String)：阿里云 region，默认值:cn-shanghai- keepalive (int)：心跳报文时间间隔,默认值60秒- clean (bool)：是否清除连接session设置,默认值false |
| 返回值 | MQTT Client连接实例 |
| Event | - connect：当连接到云端成功时触发- offline：当连接断开时触发- message：当接收到来自云端消息时触发- error：当发生错误时触发，比如PK、DN、DS有误导致连接失败时 |

### 使用示例

#### 一机一密设置

在创建实例时输入设备的三元组信息，下面是代码实例：

```
// node引入包名
const iot = require('alibabacloud-iot-device-sdk');

//创建iot.device对象将会发起到阿里云IoT的连接
const device = iot.device({
  productKey: '<productKey>', //将<productKey>修改为实际产品的ProductKey
  deviceName: '<deviceName>',//将<deviceName>修改为实际设备的DeviceName
  deviceSecret: '<deviceSecret>',//将<deviceSecret>修改为实际设备的DeviceSecret
});


//监听connect事件
device.on('connect', () => {
  //将<productKey> <deviceName>修改为实际值
  device.subscribe('/<productKey>/<deviceName>/user/get');
  console.log('connect successfully!');
  device.publish('/<productKey>/<deviceName>/user/update', 'hello world!');
});

//监听message事件
device.on('message', (topic, payload) => {
  console.log(topic, payload.toString());
});
```

注：

- 如果设备异常断开，程序会自动尝试与云端建立连接
- keep alive默认值是60，设置时不能小于60

#### 设置云端站点 region

阿里云IoT在多个国家与地区部署了服务器，默认地域为上海。厂商可设置设备需要连接的地域，在创建device时将文档中指定的**Region ID** 填入regionId即可，下面的代码示例指定地域为 ap-northeast-1（东京站点）：

```
var device = iot.device({
  productKey: '<productKey>',
  deviceName: '<deviceName>',
  deviceSecret: '<deviceSecret>',
  regionId: 'ap-northeast-1'
});

device.on('connect', () => {
  console.log('connect successfully!');
});
```

注：region需要与产品在IoT平台上创建产品时所在的region保持一致。

### 一型一密设置

若产品被设置为一型一密，设备可以通过iot.register()去获取设备的device_secret，示例代码如下：

```
const params = {
  productKey:"xxxxxx",
  productSecret:"xxxxxx",
  deviceName:"xxxxxx"
}

var device;

iot.register(params,(res)=>{
  console.log("register:",res);
  if(res.code == '200'){
    // res.data.deviceSecret 是云端反馈的设备密钥，请妥善保存该密钥，
    // 如果设备使用三元组连接物联网平台成功，再次使用本函数去获取DeviceSecret将会失败

    //创建设备对象去连接阿里云IoT
    device = iot.device({
       productKey: '<productKey>',
       deviceName: '<deviceName>',
       deviceSecret: res.data.deviceSecret, //res.data.deviceSecret 是云端反馈的设备密钥
    });
  }
})
```

Git中的示例文件：https://github.com/aliyun/alibabacloud-iot-device-sdk/blob/master/examples/one_model_one_secret.js

动态注册时res.code可能的数值列表：

| 数值 | 含义说明 |
| --- | --- |
| 200 | 成功获取DeviceSecret |
| 5005 | 无效产品，设备提供的ProductKey有误 |
| 6100 | 无效设备，云端查找不到设备提供的deviceName对应的设备 |
| 6288 | 产品不支持动态注册，请在IoT平台为产品打开动态注册功能 |
| 6289 | 设备已经激活，云端拒绝提供DeviceSecret |
| 6600 | 校验错误，设备提供的ProductSecret有误 |

### 断开与云端的连接

如果希望断开与云端的连接，可以调用end函数来断开云端的连接：

```
const iot = require('alibabacloud-iot-device-sdk');
```

```
const device = iot.device({
  productKey: '<productKey>',
  deviceName: '<deviceName>',
  deviceSecret: '<deviceSecret>',
});

...

/*Disconnect from aliyun IoT platform*/
device.end();
```