

# 通用设备接入协议

## 通用协议

### 设备注册



动态注册设备接口： --

该接口为线上接口,如需调试,请更换域名

请求方式：POST

请求数据类型：application/json

请求示例:

```
JSON
1  {
2      "cipherText": "",
3      "productKey": ""
4  }
```

请求参数:

参数名称	参数说明	是否必须	数据类型	schema
cipherText	密文	true	string	数据
productKey	产品标识	true	string	线下分配

响应示例:

```
JSON
1  {
2      "code":200,
3      "data":"返回数据密文",
4      "message":"请求成功!!!",
5      "traceId":""
6  }
```



## 数据加密规则

加密算法 : AES/ECB/PKCS5Padding

密钥数字是16个字节，参数值应为128

cipherText 的取值数据:

JSON

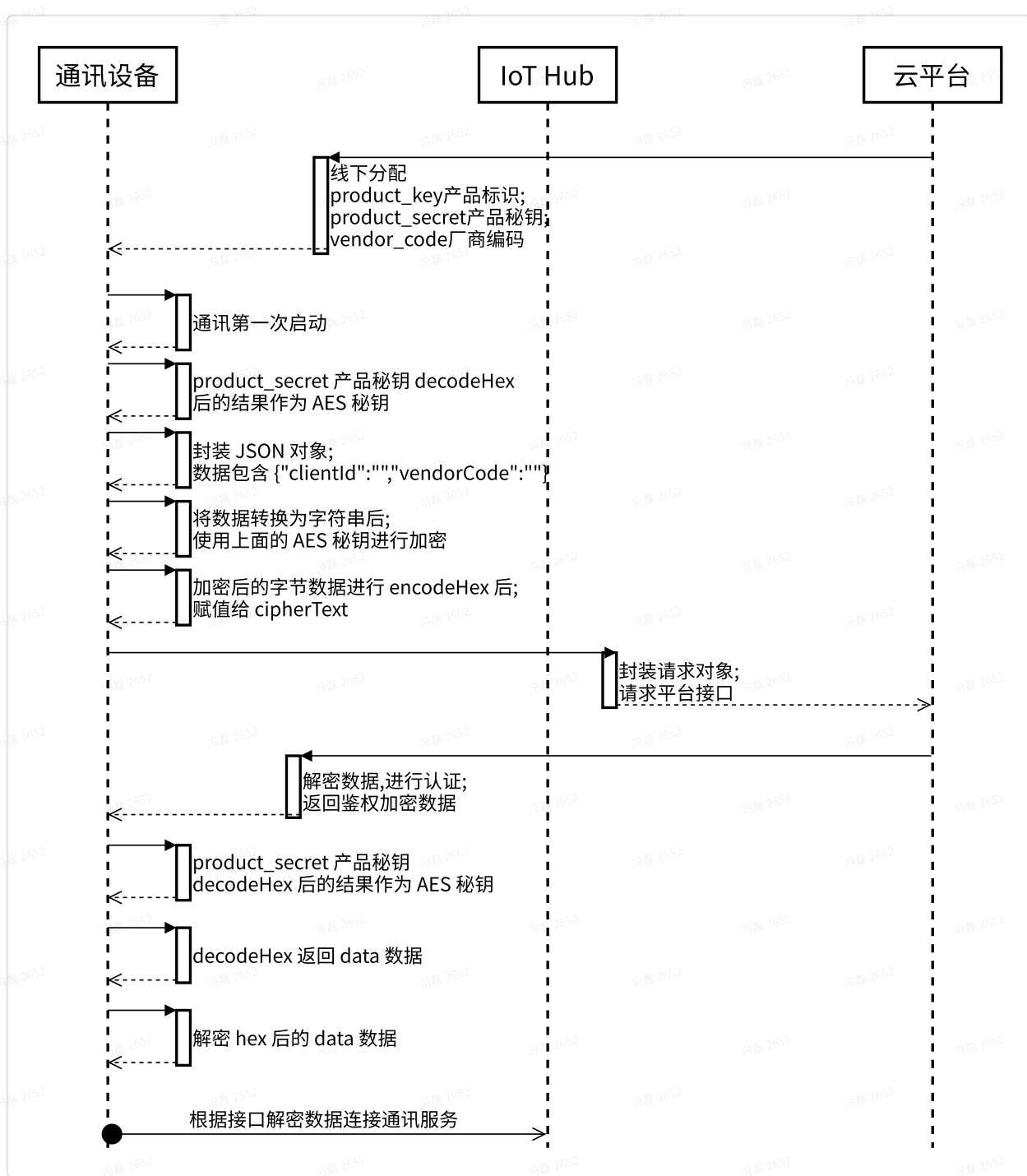
```
1 {  
2   "clientId": "",  
3   "vendorCode": ""  
4 }
```

data 的密文取值数据:

JSON

```
1 {  
2   "deviceKey": "",  
3   "deviceSecret": "",  
4   "clientId": ""  
5 }
```

设备注册加密流程



## 通讯数据格式

### 公共数据报文

JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "",
4      "controlType": "",
5      "messageType": "",
6      "vendor": "厂商编码",
7      "timestamp": "时间戳",
8      "sequence": "唯一序列",
9      "data": [
10
11  ]
12 }
```

参数	说明	取值范围
version	报文版本	报文版本 2.0
flowDirection	数据流向	0 下行指令 ,1 上行消息
controlType	指令类型	unbind/ota/control.func/control.prop/control.scene/query.p op/scene.config
messageType	消息类型	scene/control.prop/event/ota/online/offline/unbind/bind /heartbeat/discovery/ack
vendor	设备厂商	
sequence	指令序列	采用不重复序列,有序增长
data	具体的数据	控制/上报数据
timestamp	当前消息时间	指令下发/设备数据时间(长整形,毫秒)

参数描述

controlType

取值	说明
unbind	设备解绑指令
ota	设备 OTA 升级指令
control.func	设备方法控制指令(存在方法参数)

control.prop	设备属性控制指令
control.scene	下发指令,触发场景指令
query.prop	获取属性指令,不进行控制
scene.config	场景配置
double.control.config	双控
curtain.switch.relation	窗帘面板配置关系

messageType

取值	说明
scene	场景执行结果
control.prop	设备控制状态上报/本地按键触发变化上报
event	设备事件上报
ota	OTA 状态数据/百分比相关数据上报
online	设备在线上报
offline	设备离线上报
unbind	设备解绑上报
bind	设备绑定上报
heartbeat	心跳上报
discovery	设备状态/设备列表数据上报
ack	控制指令回复上报
curtain.switch.relation	窗帘面板配置关系
control.func	方法执行返回

指令下行数据

指令下行报文,一般适用于设备控制,触发控制设备属性或者设备方法  
例如 : 触发灯光的开关或者网关的组网

query.prop 属性获取

control.prop 属性控制

control.func 方法控制

当 controlType 的取值 为 control.prop 时 code 为具体需要修改属性, value 为具体的值

当 controlType 的取值为 control.func 时 code 为具体需要触发的方法名, value 为 key-value 数据, key为方法依赖的参数, value 为具体的参数数据取值

JSON

```
1 {
2     "model": "",
3     "parentDeviceId": "",
4     "deviceId": "",
5     "code": "",
6     "value": ""
7 }
```

参数	说明	取值范围
model	设备模型编码	具体定义的设备编码
parentDeviceId	父级设备标识	只有设备数据为子设备时,存在数据
deviceId	设备标识	实际产生设备数据下行/上行的标识
code	属性编码	模型数据定义(control.prop/control.func/query.prop ) 时数据取值)
value	数据取值	属性值(control.prop/control.func 时数据取值)

指令上行数据

指令上行报文,当设备控制下行指令执行成功或手动触发设备变化时,进行数据上行

例如 : 设备断网离线,入网在线,或者手动打开灯光时均为指令上行

```
JSON

1  {
2      "model": "",
3      "parentDeviceId": "",
4      "deviceId": "",
5      "code": "",
6      "value": "",
7  }
```


参数	说明	取值范围
model	设备模型编码	具体定义的设备编码
parentDeviceId	父级设备标识	只有设备数据为子设备时,存在数据
deviceId	设备标识	实际产生设备数据下行/上行的标识
code	属性编码	模型数据定义(control.prop/query.prop/event ) 时数据取值)
value	数据取值	属性值(control.prop/queue.prop 时数据取值)

## Server Topic

### 变量描述

modelCode	模型编码
productKey	产品编码
productSecret	产品密钥
deviceKey	实例连接用户
deviceSecret	实例连接密钥
vendor_code	分配的厂商编码
hex_mode_id	将设备模型的编码.替换为 _

### Topic 定义

 {vendor\_code}/{hex\_mode\_id}/{clientId}/message/up ---> 数据上行


- 设备状态变化
- 设备事件数据上报
- 主动查询上报数据
- 设备心跳上行

 {vendor\_code}/{hex\_mode\_id}/{clientId}/command/down ---> 数据下行

- 数据指令下行

 {vendor\_code}/{hex\_mode\_id}/{clientId}/command/ack ---> 数据上行

- 设备指令执行结果上行

 {vendor\_code}/{hex\_mode\_id}/{clientId}/ota/ack ---> 数据上/下行

- 设备下行 OTA 指令,拉取升级报文
- 设备上行升级结果

## **vendor\_code/hex\_mode\_id/{clientId}/message/up**

control.prop 设备控制状态上报/本地按键触发变化上报

event 设备事件上报

online 设备在线上报

offline 设备离线上报

unbind 设备解绑上报

bind 设备绑定上报

heartbeat 心跳上报

discovery 设备状态/设备列表数据上报

## **vendor\_code/hex\_mode\_id/{clientId}/command/down**

unbind 设备解绑指令

control.func 设备方法控制指令(存在方法参数)

control.prop 设备属性控制指令

control.scene 下发指令,触发场景指令



query.prop 获取属性指令,不进行控制

scene.config 场景配置

vendor\_code/hex\_mode\_id/{clientId}/command/ack

scene 场景执行结果h

ack 控制指令回复上报

vendor\_code/hex\_mode\_id/{clientId}/ota/ack

OTA 状态数据/百分比先关数据上报

设备 OTA 升级指令

## 📌 数据公约

1. 所有数据类型均为 String 都为字符串类型
2. 当上报数据 messageType 为 event 数据类型时,可能涉及到属性变更,data 中 value 的数据为键值对象
3. 当下行指令数据 controlType 的 control.func 数据类型时,可能涉及到方法属性参数,data 中 value 的数据为键值对象
4. 时间数据交互均以 GMT+8 毫秒时间数据,如需其他精度,请自行转换数据
5. 解绑/绑定/离线/在线上报的数据中 data 中 code 与 value 均无值

## 数据描述

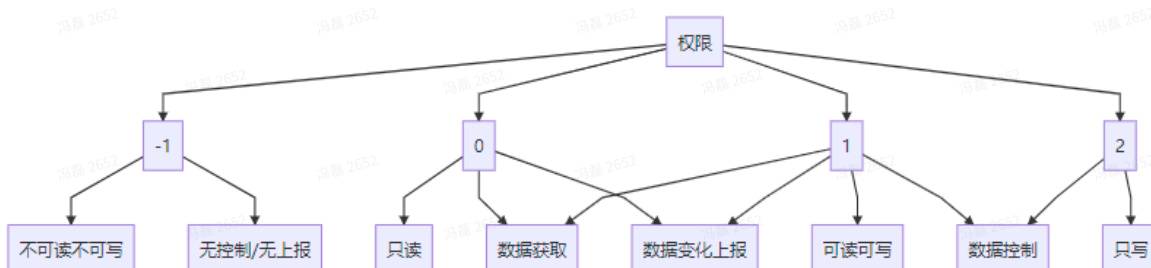
### 协议版本

版本数据指令 version 均为 2.0 版本

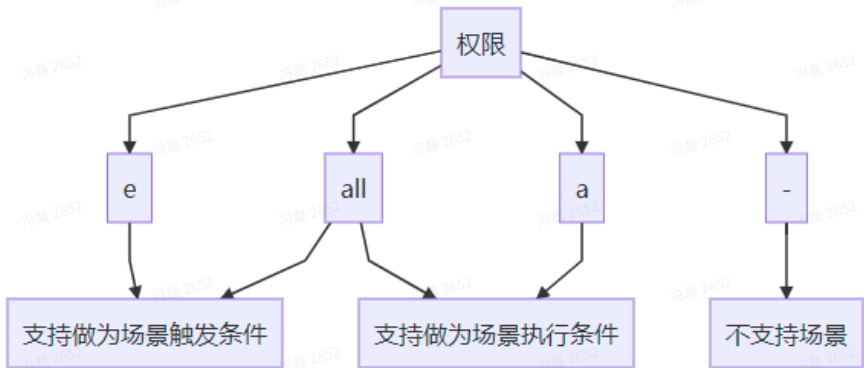
## 📌 设备标识规则

采用 {vendor\_code}.XXX不重复的序列(控制在12 位以 ,字母与数字组成)

## 属性控制权限



场景支持权限




通用报文数据

修改MQTT 通讯地址

JSON

```
1 {
2     "version": "2.0",
3     "flowDirection": "0",
4     "controlType": "change.server",
5     "messageType": "",
6     "vendor": "{vendor_code}",
7     "timestamp": "1604556783590",
8     "sequence": "16045483329904544",
9     "data": [
10        {
11            "model": "{model_code}",
12            "parentDeviceId": "",
13            "deviceId": "{设备唯一标识}",
14            "code": "server_address",
15            "value": "U2FsdGVkX1/g6efnzI5dm+30p2bsVRzm+3BoWpsVM6W06Yfv2RYVtSOMJi+ol0JdEOKbASeetSHYNniGmCfCGQ=="
16        }
17    ]
18 }
19 }
```

 修改 MQTT 通讯地址,该指令相对较为特殊进行特殊处理

数据加解密规则

## Plain Text

- 1 将 productSecret 进行 hex 后作为 AES 的密钥
- 2 解析 value 中的加密数据, value 中的数据如下

## 加密数据格式

### JSON

```
1 {
2   "authServerAddr": "https://*****",
3   "mqttAddr": "aiot*****",
4   "mqttPort": "*****"
5 }
```

## 在线/离线

### JSON

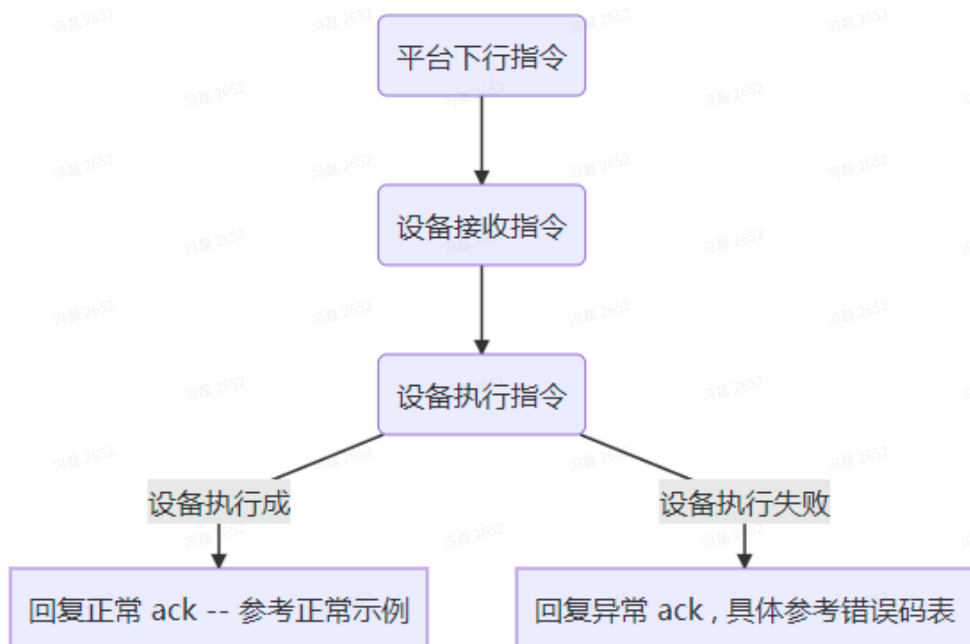
```
1 {
2   "version": "2.0",
3   "flowDirection": "1",
4   "controlType": "",
5   "messageType": "online | offline",
6   "vendor": "{vendor_code}",
7   "timestamp": "1604556783590",
8   "sequence": "16045483329904544",
9   "data": [
10    {
11      "model": "{model_code}",
12      "parentDeviceId": "11915899361150612313",
13      "deviceId": "11915899361150612313",
14      "code": "online | offline",
15      "value": "{online:1 ; offline:0}"
16    }
17  ]
18 }
```

## ack 回复

注意:

所有的下行指令,均要以下行 sequence 指令序列作为 ack 回复的上行 sequence 序列进行回复

下行指令都要进行 ack 回复,断定指令是否执行成功



## 正常示例

JSON

```
1 {
2   "version": "2.0",
3   "flowDirection": "1",
4   "controlType": "",
5   "messageType": "ack",
6   "vendor": "{vendor_code}",
7   "timestamp": "1617008027337",
8   "sequence": "下行指令序列",
9   "data": [
10    {
11      "model": "{modelCode}",
12      "parentDeviceId": "",
13      "deviceId": "{clientId}",
14      "code": "ack",
15      "value": "1"
16    }
17  ]
18 }
```

## 异常示例

异常数据反馈时需要告知执行错误原因

## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "ack",
6      "vendor": "{vendor_code}",
7      "timestamp": "1617008027337",
8      "sequence": "下行指令序列",
9      "data": [
10         {
11             "model": "{modelCode}",
12             "parentDeviceId": "",
13             "deviceId": "{mac}",
14             "code": "ack",
15             "value": "0:100010:人员数据无效"
16         }
17     ]
18 }
```

## 电量变化

低功耗设备,电池设备均要进行电量变化上报,接受大跨度区间

## JSON

```
1  {
2      "version": "1.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "control.prop",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{modelCode}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "battery",
15             "value": "1%-100%"
16         }
17     ]
18 }
```

## 心跳

直接连接平台通讯设备,需要每 90 秒上报心跳的数据

### JSON

```
1  {
2      "version":"2.0",
3      "flowDirection":"1",
4      "controlType": "",
5      "messageType": "heartbeat",
6      "vendor": "{vendor_code}",
7      "timestamp": "1617008027337",
8      "sequence": "",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "",
13             "deviceId": "设备唯一标识",
14             "code": "ota_version",
15             "value": "MCUP134_1V2.19"
16         }
17     ]
18 }
```

## 低电量

低功耗设备,电池设备均要进行电量低事件上报,接受大跨度区间

## JSON

```
1  {
2      "version": "1.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "event",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{modelCode}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "low_battery",
15             "value": {}
16         }
17     ]
18 }
```

## 属性获取

注意：设备模型中的属性,均要可以支持主动下发报文,进行数据上报

## 属性下行查询

## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "0",
4      "controlType": "query.prop",
5      "messageType": "",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "{具体需要查询的属性}",
15             "value": ""
16         }
17     ]
18 }
```

## 属性查询上报

## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "event",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "power",
15             "value": "1"
16         }
17     ]
18 }
```

## 设备离线上线状态同步



## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "device.prop.sync",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "{设备属性}",
15             "value": "{取值取值}"
16         },
17         {
18             "model": "{model_code}",
19             "parentDeviceId": "11915899361150612313",
20             "deviceId": "11915899361150612313",
21             "code": "{设备属性}",
22             "value": "{取值取值}"
23         }
24     ]
25 }
```

## 注意

设备状态同步,走特殊的上报类型,不能进行设备的场景触发

## 属性控制

## 属性下行控制

## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "0",
4      "controlType": "control.prop",
5      "messageType": "",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "{具体需要控制的属性}",
15             "value": "{取值范围}"
16         }
17     ]
18 }
```

## 属性变化上报

### JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "control.prop",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "{具体被控制的属性}",
15             "value": "{取值范围}"
16         }
17     ]
18 }
```

## 注意

所有的设备属性;属性状态发生变化后;不管是物理控制,还是指令控制,都需要上报该报文

## 方法控制

### JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "0",
4      "controlType": "control.func",
5      "messageType": "",
6      "vendor": "{vendor_code}",
7      "timestamp": "1617008027337",
8      "sequence": "",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "",
13             "deviceId": "{mac}",
14             "code": "{具体需要实现的方法}",
15             "value": {
16                 "{方法输入参数的 KEY}": "{方法输出参数的值}"
17             }
18         }
19     ]
20 }
```

## 事件推送

## JSON

```
1  {
2      "version": "2.0",
3      "flowDirection": "1",
4      "controlType": "",
5      "messageType": "event",
6      "vendor": "{vendor_code}",
7      "timestamp": "1604556783590",
8      "sequence": "16045483329904544",
9      "data": [
10         {
11             "model": "{model_code}",
12             "parentDeviceId": "11915899361150612313",
13             "deviceId": "11915899361150612313",
14             "code": "{触发的事件}",
15             "value": {}
16         }
17     ]
18 }
```

注意：下行指令执行,均需要 ack 进行数据回复,告知是否成功或失败原因