ZHC466C_JSON_应用指导

目 录

Content

1.AO

1. 1. 设置 AO 输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
AO1	否	字符	AO 输出值 0~65535mV/uA **非数字字符均认为设置 0** 例如:“3000”则 AO 输出 3000mV/uA “abc”则 AO 输出 0mV/uA
AO2	否	字符	
AO3	否	字符	
AO4	否	字符	

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
AO1	否	字符	+success+:成功
AO2	否	字符	
AO3	否	字符	
AO4	否	字符	

1.1.1 示例 1

以设置 AO1、AO2、AO3 为例：

```
{
    "msgType": "setAOValue",
    "devID": "466C210730024164",
    "data": {
        "AO1": "3000",
        "AO2": "3000",
        "AO3": "3000"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOValueAck",
    "data": {
        "AO1": "+success+",
        "AO2": "+success+",
        "AO3": "+success+"
    },
    "timestamp": "1628481774"
}
```

1.1.2.示例 2

以设置 AO1、AO2、AO3 为例：

```
{
    "msgType": "setAOValue",
    "devID": "466C210730024164",
    "data": {
        "AO1": "abc",
        "AO2": "3000",
        "AO3": "3000"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOValueAck",
    "data": {
```

```
            "AO1": "+success+",
            "AO2": "+success+",
            "AO3": "+success+"
        },
        "timestamp": "1628481774"
}
```

AO1 实际输出 0。

1. 2. 读取 AO 输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	设参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
AO1	是	字符	0~65535mV/uA
AO2	是	字符	0~65535mV/uA
AO3	是	字符	0~65535mV/uA
AO4	是	字符	0~65535mV/uA

1.2.1 示例

读取 AO：

```
{
   "msgType": "getAOValue",
   "devID": "466C210730024164",
   "data": {}
}
```

响应：

```
{
   "devID": "466C210730024164",
   "msgType": "getAOValueAck",
```

```
  "data": {
    "AO1": "3000",
    "AO2": "3000",
    "AO3": "3000",
    "AO4": "3000"
  },
  "timestamp": "1628131528"
}
```


1.3.读取 AO 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	是	字符	状态改变使能。65535/启用 0/禁用
rstStatus	是	字符	重启后 AO 的输出值。 1/保持最近一次输出值 2/输出 0mV/uA

1.3.1 示例

读取 AO：


```
{
  "msgType": "getAOBasic",
  "devID": "466C210730024164",
  "data": {}
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "getAOBasicAck",
```

```
    "data": {
        "enRpt": 0,
        "rstStatus": 0
    },
    "timestamp": "1628757040"
}
```


1.4.写入 AO 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	否	字符	状态改变使能。65535/启用 0/禁用
rstStatus	否	字符	重启后 AO 的输出值。 1/保持最近一次输出值 2/输出 0mV/uA

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	否	字符	+success+:成功 +value_error+:值错误
rstStatus	否	字符	+success+:成功 +value_error+:值错误

1.4.1 示例 1

写入 AO Basic：


```
{
    "msgType": "setAOBasic",
    "devID": "466C210730024164",
```

```
    "data": {
        "enRpt": 1,
        "rstStatus": 1
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOBasicAck",
    "data": {
        "enRpt": "+success+",
        "rstStatus": "+success+"
    },
    "timestamp": "1628489741"
}
```

1.4.2 示例 2

写入 AO Basic：

```
{
    "msgType": "setAOBasic",
    "devID": "466C210730024164",
    "data": {
        "enRpt": 1,
        "rstStatus": 3
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOBasicAck",
    "data": {
        "enRpt": "+success+",
        "rstStatus": "+value_error+"
    },
    "timestamp": "1628757188"
}
```

1.5.读取 AO 输出保持时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOHold
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOHoldAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
hold1	是	字符	状态维持时间 1 单位：s/秒
hold2	是	字符	状态维持时间 2 单位：s/秒
hold3	是	字符	状态维持时间 3 单位：s/秒
hold4	是	字符	状态维持时间 4 单位：s/秒

1.5.1 示例

读取 AO Hold：

```
{
   "msgType": "getAOHold",
   "devID": "466C210730024164",
   "data": {}
}
```

响应：

```
{
   "devID": "466C210730024164",
   "msgType": "getAOHoldAck",
   "data": {
      "hold1": "10",
      "hold2": "10",
      "hold3": "10",
      "hold4": "10"
   },
   "timestamp": "1628131528"
}
```

1.6.写入 AO 输出保持时间

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOHold
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
hold1	否	字符	状态维持时间 1 单位：s/秒
hold2	否	字符	状态维持时间 2 单位：s/秒
hold3	否	字符	状态维持时间 3 单位：s/秒
hold4	否	字符	状态维持时间 4 单位：s/秒

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOHoldAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	设置结果根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
hold1	否	字符	+success+:成功
hold2	否	字符	+success+:成功
hold3	否	字符	+success+:成功
hold4	否	字符	+success+:成功

1.6.1 示例 1

写入 AO Hold：

```
{
   "msgType": "setAOHold",
   "devID": "466C210730024164",
   "data": {
      "hold1": "10",
      "hold2": "10",
      "hold3": "10",
      "hold4": "10"
   }
```

```
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOHoldAck",
    "data": {
        "hold1": "+success+",
        "hold2": "+success+",
        "hold3": "+success+",
        "hold4": "+success+"
    },
    "timestamp": "0"
}
```

1.6.2 示例 2

写入 AO Hold：

```
{
  "msgType": "setAOHold",
  "devID": "466C210730024164",
  "data": {
    "hold1": "abc",
    "hold2": "10",
    "hold3": "10",
    "hold4": "10"
  }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOHoldAck",
    "data": {
        "hold1": "+success+",
        "hold2": "+success+",
        "hold3": "+success+",
        "hold4": "+success+"
    },
    "timestamp": "0"
}
```

AO1 实际输出维持时间为 0。

1.7.读取 AO 默认输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAODefault
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAODefaultAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
en1	是	字符	默认输出使能 1。 65535/启用 0/禁用
en2	是	字符	默认输出使能 2。 65535/启用 0/禁用
en3	是	字符	默认输出使能 3。 65535/启用 0/禁用
en4	是	字符	默认输出使能 4。 65535/启用 0/禁用
v1	是	字符	状态维持时间 1 单位：s/秒
v2	是	字符	状态维持时间 2 单位：s/秒
v3	是	字符	状态维持时间 3 单位：s/秒
v4	是	字符	状态维持时间 4 单位：s/秒

1.7.1 示例

读取 AO Default：

```
{
   "msgType": "getAODefault",
   "devID": "466C210730024164",
   "data": {}
}
```

响应：

```
{
      "devID": "466C210730024164",
      "msgType": "getAODefaultAck",
      "data": {
            "en1": 0,
            "v1": "0",
            "en2": 0,
            "v2": "0",
```

```
            "en3": 0,
            "v3": "0",
            "en4": 0,
            "v4": "0"
        },
        "timestamp": "0"
}
```


1.8.写入 AO 默认输出

请求帧格式：

字段	属性	必须	类型	描述
msgType	消息类型	是	字符	setAODefault
devID	设备 ID	是	字符	设备唯一识别码，16 位
data	值根节点	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
en1	否	字符	默认输出使能 1。 65535/启用 0/禁用
en2	否	字符	默认输出使能 2。 65535/启用 0/禁用
en3	否	字符	默认输出使能 3。 65535/启用 0/禁用
en4	否	字符	默认输出使能 4。 65535/启用 0/禁用
v1	否	字符	状态维持时间 1 单位：s/秒
v2	否	字符	状态维持时间 2 单位：s/秒
v3	否	字符	状态维持时间 3 单位：s/秒
v4	否	字符	状态维持时间 4 单位：s/秒

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAODefaultAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
en1	否	字符	+success+:成功 +value_error+:值错误
en2	否	字符	+success+:成功
en3	否	字符	+value_error+:值错误
en4	否	字符	+success+:成功 +value_error+:值错误
v1	否	字符	+success+:成功

v2	否	字符	+success+:成功
v3	否	字符	+success+:成功
v4	否	字符	+success+:成功

1.8.1 示例 1

写入 AO Default：

```
{
    "devID": "466C210730024164",
    "msgType": "setAODefault",
    "data": {
        "en1": 1,
        "v1": "1000",
        "en2": 1,
        "v2": "1000",
        "en3": 1,
        "v3": "3000",
        "en4": 1,
        "v4": "4000"
    },
    "timestamp": "0"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAODefaultAck",
    "data": {
        "en1": "+success+",
        "v1": "+success+",
        "en2": "+success+",
        "v2": "+success+",
        "en3": "+success+",
        "v3": "+success+",
        "en4": "+success+",
        "v4": "+success+"
    },
    "timestamp": "1628757742"
}
```

1.8.2 示例 2

写入 AO Default：

```
{
    "devID": "466C210730024164",
    "msgType": "setAODefault",
    "data": {
        "en1": 3,
        "v1": "1000",
        "en2": 3,
        "v2": "1000",
        "en3": 3,
        "v3": "3000",
        "en4": 3,
        "v4": "4000"
    },
    "timestamp": "0"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAODefaultAck",
    "data": {
        "en1": "+value_error+",
        "v1": "+success+",
        "en2": "+value_error+",
        "v2": "+success+",
        "en3": "+value_error+",
        "v3": "+success+",
        "en4": "+value_error+",
        "v4": "+success+"
    },
    "timestamp": "1628757848"
}
```

1.9.读取 AO 循环输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

data 帧格式

字段	必须	类型	描述
index	是	字符	AO 通道。1~4

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAOHbtAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	指定的输出通道。0~3
cyc	是	字符	相邻两个输出值的间隔时间。 单位：s/秒
v1	是	字符	0~65535mV/uA
v2	是	字符	0~65535mV/uA
v3	是	字符	0~65535mV/uA
v4	是	字符	0~65535mV/uA
v5	是	字符	0~65535mV/uA
v6	是	字符	0~65535mV/uA
v7	是	字符	0~65535mV/uA
v8	是	字符	0~65535mV/uA

1.9.1 示例

读取 AO Hbt：

```
{
    "msgType": "getAOHbt",
    "devID": "466C210730024164",
    "data": {
        "index": "1"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getAOHbtAck",
    "data": {
        "index": "1",
        "cyc": "10",
```

```
            "v1": "3000",
            "v2": "4000",
            "v3": "5000",
            "v4": "6000",
            "v5": "7000",
            "v6": "8000",
            "v7": "9000",
            "v8": "10000"
        },
        "timestamp": "1628492404"
}
```


1.10.写入 AO 循环输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	指定的输出通道。AO1/AO2/AO3/AO4
cyc	否	字符	相邻两个输出值的间隔时间。 单位：s/秒
v1	否	字符	0~65535mV/uA
v2	否	字符	0~65535mV/uA
v3	否	字符	0~65535mV/uA
v4	否	字符	0~65535mV/uA
v5	否	字符	0~65535mV/uA
v6	否	字符	0~65535mV/uA
v7	否	字符	0~65535mV/uA
v8	否	字符	0~65535mV/uA

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOCarouselAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述

index	是	字符	+value_error+:值错误 指定的输出通道。1~4
cyc	否	字符	+success+:成功
v1	否	字符	+success+:成功
v2	否	字符	+success+:成功
v3	否	字符	+success+:成功
v4	否	字符	+success+:成功
v5	否	字符	+success+:成功
v6	否	字符	+success+:成功
v7	否	字符	+success+:成功
v8	否	字符	+success+:成功

1.10.1 示例 1

写入 AO Hbt：

```
{
    "msgType": "setAOHbt",
    "devID": "466C210730024164",
    "data": {
        "index": "1",
        "cyc": "10",
        "v1": "3000",
        "v2": "4000",
        "v3": "5000",
        "v4": "6000",
        "v5": "7000",
        "v6": "8000",
        "v7": "9000",
        "v8": "10000"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAOHbtAck",
    "data": {
        "index": "1",
        "cyc": "+success+",
        "v1": "+success+",
        "v2": "+success+",
        "v3": "+success+",
        "v4": "+success+",
        "v5": "+success+",
```

```
            "v6": "+success+",
            "v7": "+success+",
            "v8": "+success+"
        },
        "timestamp": "1628492187"
}
```

2.AI

2.1.读取 AI 输入

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
AI1	是	字符	0~65535mV/uA
AI2	是	字符	0~65535mV/uA
AI3	是	字符	0~65535mV/uA
AI4	是	字符	0~65535mV/uA

2.1.1 示例

读取 AI Value：

```
{
  "msgType": "getAIValue",
  "devID": "466C210730024164",
  "data": {}
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getAIValueAck",
    "data": {
        "AI1": "43",
        "AI2": "39",
        "AI3": "39",
```

```
            "AI4": "39"
        },
        "timestamp": "1628492639"
}
```


2.2.读取 AI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	是	字符	状态改变使能。65535/启用 0/禁用
cyc	是	字符	循环上报周期。单位：s/秒

2.2.1 示例

读取 AI Basic：


```
{
   "msgType": "getAIBasic",
   "devID": "466C210730024164",
   "data": {}
}
```


响应：


```
{
        "devID": "466C210730024164",
        "msgType": "getAIBasicAck",
        "data": {
              "enRpt": 0,
              "cyc": "30"
        },
        "timestamp": "1628758069"
```

}

2.3.写入 AI 基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
enRpt	否	字符	状态改变使能。65535/启用 0/禁用
cyc	是	字符	循环上报周期。单位：s/秒

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
enRpt	否	字符	+success+:成功 +value_error+:值错误
cyc	否	字符	+success+:成功

2.3.1 示例 1

写入 AI Basic：

```
{
   "msgType": "setAIBasic",
   "devID": "466C210730024164",
   "data": {
      "enRpt": 0,
       "cyc": "30"
   }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAIBasicAck",
    "data": {
        "enRpt": "+success+",
        "cyc": "+success+"
    },
    "timestamp": "1628758123"
}
```


2.3.2 示例 2

写入 AI Basic：


```
{
    "msgType": "setAIBasic",
    "devID": "466C210730024164",
    "data": {
        "enRpt": 5,
        "cyc": "30"
    }
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setAIBasicAck",
    "data": {
        "enRpt": "+value_error+",
        "cyc": "+success+"
    },
    "timestamp": "1628758180"
}
```


2.4.读取 AI 低通参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIFilter
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIFilterAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
filter1	是	字符	低通参数 1。0~100
filter2	是	字符	低通参数 2。0~100
filter3	是	字符	低通参数 3。0~100
filter4	是	字符	低通参数 4。0~100

2.4.1 示例

读取 AI Filter：

```
{
    "msgType": "getAIFilter",
    "devID": "466C210730024164",
    "data": {}
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getAIFilterAck",
    "data": {
        "filter1": "50",
        "filter2": "50",
        "filter3": "50",
        "filter4": "50"
    },
    "timestamp": "1628493016"
}
```

2.5.写入 AI 低通参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOFilter
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
filter1	否	字符	低通参数 1。0~100
filter2	否	字符	低通参数 2。0~100
filter3	否	字符	低通参数 3。0~100
filter4	否	字符	低通参数 4。0~100

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAOFilterAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
filter1	否	字符	+success+:成功 +value_error+:值错误
filter2	否	字符	+success+:成功 +value_error+:值错误
filter3	否	字符	+success+:成功 +value_error+:值错误
filter4	否	字符	+success+:成功 +value_error+:值错误

2.5.1 示例 1

写入 AI Filter：

```
{
    "msgType": "setAIFilter",
    "devID": "466C210730024164",
    "data": {
        "filter1": "100",
        "filter2": "10",
        "filter3": "10",
        "filter4": "10"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
```

```
    "msgType": "setAIFilterAck",
    "data": {
        "filter1": "+success+",
        "filter2": "+success+",
        "filter3": "+success+",
        "filter4": "+success+"
    },
    "timestamp": "1628493407"
}
```


2.5.2 示例 2

写入 AI Filter：


```
{
    "msgType": "setAIFilter",
    "devID": "466C210730024164",
    "data": {
        "filter1": "101",
        "filter2": "10",
        "filter3": "10",
        "filter4": "10"
    }
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setAIFilterAck",
    "data": {
        "filter1": "+value_error+",
        "filter2": "+success+",
        "filter3": "+success+",
        "filter4": "+success+"
    },
    "timestamp": "1628493412"
}
```


2.6.读取 AI 上报规则

请求帧格式：

字段	必须	类型	描述

msgType	是	字符	getAIRule
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	AO 通道。1~4

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAIRuleAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	AO 通道。1~4
en	是	字符	规则使能标志。655535/启用 0/禁用
min	是	字符	上报区间下限。0~65535
max	是	字符	上报区间上限。0~65535

2.6.1 示例

读取 AI Rule：

```
{
    "msgType": "getAIRule",
    "devID": "466C210730024164",
    "data": {
        "index": "1"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getAIRuleAck",
    "data": {
        "index": "1",
        "en": 0,
        "min": "0",
```

```
            "max": "0"
        },
        "timestamp": "1628758286"
}
```


2.7.写入 AI 上报规则

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setAIRule
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	AO 通道。1~4
en	否	字符	规则使能标志。655535/启用 0/禁用
min	否	字符	上报区间下限。0~65535
max	否	字符	上报区间上限。0~65535

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setAORuleAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 AO 通道。1~4
en	否	字符	+success+:成功 +value_error+:值错误
min	否	字符	+success+:成功
max	否	字符	+success+:成功

2.7.1 示例 1

写入 AI Rule：

```
{
    "msgType": "setAIRule",
    "devID": "466C210730024164",
    "data": {
        "index": "1",
        "en": 1,
        "min": "3000",
        "max": "6000"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAIRuleAck",
    "data": {
        "index": "1",
        "en": "+success+",
        "min": "+success+",
        "max": "+success+"
    },
    "timestamp": "0"
}
```

2.7.2 示例 2

写入 AI Rule：

```
{
    "msgType": "setAIRule",
    "devID": "466C210730024164",
    "data": {
        "index": "1",
        "en": 5,
        "min": "3000",
        "max": "6000"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAIRuleAck",
    "data": {
        "index": "1",
        "en": "+value_error+",
```

```
            "min": "+success+",
            "max": "+success+"
        },
        "timestamp": "1628758377"
}
```

3.串口

3.1.设置串口立即输出

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartValue
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
payload	是	字符	16 进制字符串。例如：要发送“123”，则 payload 为“313233”

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartValueAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
payload	是	字符	+success+:成功

3.1.1 示例

写入 Uart Value：

```
{
    "msgType": "setUartValue",
    "devID": "466C210730024164",
    "data": {
        "payload": "31323334353637383930"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setUartValueAck",
    "data": {
        "payload": "+success+"
    },
    "timestamp": "1628497881"
}
```

3.2.读取串口基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
br	是	字符	波特率。1200~115200
sb	是	字符	停止位。1:1 位 2:1.5 位 3:2 位
wl	是	字符	数据位。8,9
pa	是	字符	校验位。1:无校验 2:奇校验 3:偶校验

3.2.1 示例

读取 Uart Basic：

```
{
  "msgType": "getUartBasic",
  "devID": "466C210730024164",
  "data": {}
}
```

响应：


```
{
    "devID": "466C210730024164",
    "msgType": "getUartBasicAck",
    "data": {
        "br": "115200",
        "sb": 0,
        "wl": 0,
        "pa": 0
    },
    "timestamp": "1628758432"
}
```


3.3.写入串口基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
br	否	字符	波特率。1200~115200
sb	否	字符	停止位。1:1 位 2:1.5 位 3:2 位
wl	否	字符	数据位。8,9
pa	否	字符	校验位。1:无校验 2:奇校验 3:偶校验

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
br	否	字符	+success+:成功
sb	否	字符	+success+:成功 +value_error+:值错误
wl	否	字符	+success+:成功 +value_error+:值错误
pa	否	字符	+success+:成功

			+value_error+:值错误

1.4.1 示例 1

写入 Uart Basic：

```
{
    "msgType": "setUartBasic",
    "devID": "466C210730024164",
    "data": {
        "br": "115200",
        "sb": 1,
        "wl": 1,
        "pa": 1
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setUartValueAck",
    "data": {
        "br": "+success+",
        "sb": "+success+",
        "wl": "+success+",
        "pa": "+success+"
    },
    "timestamp": "1628758478"
}
```

1.4.2 示例 2

写入 Uart Basic：

```
{
    "msgType": "setUartBasic",
    "devID": "466C210730024164",
    "data": {
        "br": " 115200",
        "sb": 5,
        "wl": 8,
        "pa": 1
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setUartValueAck",
    "data": {
        "br": "+success+",
        "sb": "+value_error+",
        "wl": "+value_error+",
        "pa": "+success+"
    },
    "timestamp": "1628758529"
}
```

3.4.读取串口心跳

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getUartHbtAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
cyc1	是	字符	距离上一条串口心跳的时间
cyc2			
cyc3			
cyc4			
payload1	是	字符	16 进制字符串。例如：要发送“123”，则 payload 为“313233”
payload2			
payload3			
payload4			

3.4.1 示例

读取串口心跳：

```
{
   "msgType": "getUartCarousel",
   "devID": "466C210730024164",
   "data": {}
}
```


响应：


```
{
   "devID": "466C210730024164",
   "msgType": "getUartCarouselAck",
   "data": {
      "cyc1": "30",
      "payload1": "313233",
      "cyc2": "30",
      "payload2": "313233",
      "cyc3": "30",
      "payload3": "313233",
      "cyc4": "30",
      "payload4": "313233"
   },
   "timestamp": "1628131528"
}
```


3.5.写入串口心跳

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
cyc1	否	字符	距离上一条串口心跳的时间
cyc2			
cyc3			
cyc4			
payload1	否	字符	16 进制字符串。例如：要发送“123”，则 payload 为“313233”
payload2			
payload3			
payload4			

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setUartHbtAck
devID	是	字符	设备唯一识别码，16 位

data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
cyc1	否	字符	+success+:成功
cyc2			
cyc3			
cyc4			
payload1	否	字符	+success+:成功 +length_error+:长度错误
payload2			
payload3			
payload4			

3.5.1.示例 1

写入串口心跳：


```
{
    "msgType": "setUartHbt",
    "devID": "466C210730024164",
    "data": {
        "cyc1": "30",
        "payload1": "31323334313233343132333431323334",
        "cyc2": "30",
        "payload2": "31323334313233343132333431323334",
        "cyc3": "30",
        "payload3": "31323334313233343132333431323334",
        "cyc4": "30",
        "payload4": "31323334313233343132333431323334"
    }
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setUartHbtAck",
    "data": {
        "cyc1": "+success+",
        "payload1": "+success+",
        "cyc2": "+success+",
        "payload2": "+success+",
        "cyc3": "+success+",
        "payload3": "+success+",
        "cyc4": "+success+",
        "payload4": "+success+"
```

```
    },
    "timestamp": "0"
}
```

3.5.2.示例 2

```
{
    "msgType": "setUartHbt",
    "devID": "466C210730024164",
    "data": {
        "cyc1": "30",
        "payload1": "31323334313233343132333431323334",
        "cyc2": "30",
        "payload2": "313233343132333431323334313233433",
        "cyc3": "30",
        "payload3": "31323334313233343132333431323334",
        "cyc4": "30",
        "payload4": "31323334313233343132333431323334"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setUartHbtAck",
    "data": {
        "cyc1": "+success+",
        "payload1": "+success+",
        "cyc2": "+success+",
        "payload2": "+length error+",
        "cyc3": "+success+",
        "payload3": "+success+",
        "cyc4": "+success+",
        "payload4": "+success+"
    },
    "timestamp": "1628498857"
}
```

4.闹钟

4.1.读取闹钟

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getClock
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	闹钟索引。1~8

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getClockAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	闹钟序号。1~8
en	是	字符	当前闹钟是否启用。65535/启用 0/禁用
hh	是	字符	时。北京时间
mm	是	字符	分。北京时间
ss	是	字符	秒。北京时间
actType	是	字符	执行的动作。2:重启 3:AO 输出
actor	是	字符	AO 通道。AO1~AO4
AO	是	字符	0~65535mV/uA

4.1.1 示例

读取闹钟：

```
{
  "msgType": "getClock",
  "devID": "466C210730024164",
```

```
    "data": {
        "index": "1"
    }
}
```


响应：


```
{
        "devID": "466C210730024164",
        "msgType": "getClockAck",
        "data": {
                "index": "1",
                "en": 0,
                "hh": "0",
                "mm": "0",
                "ss": "0",
                "actType": 1,
                "actor": 1,
                "AO": "256"
        },
        "timestamp": "1628758568"
}
```


4.2.写入闹钟

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setClock
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	闹钟序号。1~8
en	否	字符	当前闹钟是否启用。65535/启用 0/禁用
hh	否	字符	时。北京时间
mm	否	字符	分。北京时间
ss	否	字符	秒。北京时间
actType	否	字符	执行的动作。2:重启 3:AO 输出
actor	否	字符	AO 通道。AO1~AO4
AO	否	字符	0~65535mV/uA

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setClockAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 闹钟序号。1~8
en	否	字符	+success+:成功 +value_error+:值错误
hh	否	字符	+success+:成功 +value_error+:值错误
mm	否	字符	+success+:成功 +value_error+:值错误
ss	否	字符	+success+:成功 +value_error+:值错误
actType	否	字符	+success+:成功 +value_error+:值错误
actor	否	字符	+success+:成功 +value_error+:值错误
AO	否	字符	+success+:成功 +value_error+:值错误

4.2.1 示例 1

写入闹钟：


```
{
    "devID": "466C210730024164",
    "msgType": "setClock",
    "data": {
        "index": "1",
        "en": 1,
        "hh": "0",
        "mm": "0",
        "ss": "0",
        "actType": 1,
        "actor": 2,
        "AO": "256"
    },
    "timestamp": "1628758568"
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setClockAck",
    "data": {
        "index": "1",
        "en": "+success+",
```

```
            "hh": "+success+",
            "mm": "+success+",
            "ss": "+success+",
            "actType": "+success+",
            "actor": "+success+",
            "AO": "+success+"
        },
        "timestamp": "1628758642"
}
```


4.2.2 示例 2

写入闹钟：


```
{
        "devID": "466C210730024164",
        "msgType": "setClock",
        "data": {
            "index": "1",
            "en": 6,
            "hh": "0",
            "mm": "0",
            "ss": "0",
            "actType": 5,
            "actor": 9,
            "AO": "256"
        },
        "timestamp": "1628758568"
}
```


响应：


```
{
        "devID": "466C210730024164",
        "msgType": "setClockAck",
        "data": {
            "index": "1",
            "en": "+value_error+",
            "hh": "+success+",
            "mm": "+success+",
            "ss": "+success+",
            "actType": "+value_error+",
            "actor": "+value_error+",
            "AO": "+success+"
        },
        "timestamp": "1628758713"
}
```

5.条件逻辑

5.1.读取条件逻辑

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLogic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLogicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	逻辑序号。1~8
cdt	是	字符	满足此条件时执行。0:禁用 3:AO 跟随 AI 4:大于等于 5:小于等于
addr	是	字符	使用设备间逻辑时，此项为触发源的 modbus 地址。否则，设置为 0
inReg	是	字符	输入寄存器。1~4
outType	是	字符	输出类型。2:AO 输出
outReg	是	字符	输出寄存器。1~4
AI	是	字符	AI 阈值。cdt ==4||cdt==5 时有效
AO	是	字符	0~65535mV/uA cdt ==4||cdt==5 时有效

5.1.1 示例

读取逻辑：

```
{
    "msgType": "getLogic",
    "devID": "466C210730024164",
    "data": {
        "index": "2"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getLogicAck",
    "data": {
        "index": "2",
        "cdt": 0,
        "addr": " 0",
        "inReg": 1,
        "outReg": 1,
        "AI": "0",
        "AO": "0"
    },
    "timestamp": "0"
}
```


5.2.写入条件逻辑

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setLogic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	逻辑序号。1~8
cdt	是	字符	满足此条件时执行。0:禁用 3:AO 跟随 AI 4:大于等于 5:小于等于
addr	是	字符	使用设备间逻辑时，此项为触发源的 modbus 地址。否则，设置为 0
inReg	是	字符	输入寄存器。1~4
outType	是	字符	输出类型。2:AO 输出
outReg	是	字符	输出寄存器。1~4
AI	是	字符	AI 阈值。cdt ==4‖cdt==5 时有效
AO	是	字符	0~65535mV/uA cdt ==4‖cdt==5 时有效

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setLogicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 逻辑序号。1~8
cdt	是	字符	+success+:成功 +value_error+:值错误
addr	是	字符	+success+:成功
inReg	是	字符	+success+:成功 +value_error+:值错误
outType	是	字符	+success+:成功 +value_error+:值错误
outReg	是	字符	+success+:成功 +value_error+:值错误
AI	是	字符	+success+:成功
AO	是	字符	+success+:成功

5.2.1 示例 1

写入逻辑：

```
{
    "devID": "466C210730024164",
    "msgType": "setLogic",
    "data": {
        "index": "2",
        "cdt": 2,
        "addr": " 0",
        "inReg": 2,
        "outReg": 2,
        "AI": "1230",
        "AO": "3330"
    },
    "timestamp": "1628758905"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setLogicAck",
    "data": {
        "index": "2",
        "cdt": "+success+",
        "addr": "+success+",
        "inReg": "+success+",
        "outReg": "+success+",
        "AI": "+success+",
        "AO": "+success+"
    },
```

```
    "timestamp": "1628759482"
}
```

5.2.2 示例 2

写入逻辑：

```
{
    "msgType": "setLogic",
    "devID": "466C210730024164",
    "data": {
        "index": "2",
        "cdt": 6,
        "addr": "0",
        "inReg": 9,
        "outReg": 2,
        "AI": "1230",
        "AO": "3330"

    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setLogicAck",
    "data": {
        "index": "2",
        "cdt": "+value_error+",
        "addr": "+success+",
        "inReg": "+value_error+",
        "outReg": "+success+",
        "AI": "+success+",
        "AO": "+success+"
        },
    "timestamp": "1628759585"
}
```

6.系统参数

6.1.读取系统参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
id	是	字符	设备出厂唯一序列号
pw	是	字符	纵横云密码
ver	是	字符	固件版本
format	是	字符	数据格式。0:modbus RTU 1:modbus TCP 2:JSON
rptDec	是	字符	数据上报方向。0:通过 LTE 1:通过串口 2:以上两种
addr	是	字符	modbus 通讯地址码。16 进制字符。例如 0x55 则 addr=55

6.1.1 示例

读取系统参数：

```
{
    "msgType": "getSysBasic",
    "devID": "AAAAAAAAAAAAAAA",
    "data": {}
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getSysBasicAck",
    "data": {
        "id": "466C210730024164",
        "pw": "123456",
```

```
            "ver": "1004",
            "format": 1,
            "rptDec": 1,
            "addr": "55"
        },
        "timestamp": "0"
}
```

6.2.写入系统参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
pw	否	字符	纵横云密码
format	否	字符	数据格式。1:modbus RTU 2:modbus TCP 3:JSON
rptDec	否	字符	数据上报方向。1:通过 LTE 2:通过串口 3:以上两种
addr	否	字符	modbus 通讯地址码。16 进制字符。例如 0x55 则 addr=55
cmd	否	字符	1:重启 2:恢复出厂设置 3:本地升级 4:远程升级

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
pw	否	字符	+success+:成功 +length_error+:长度错误
format	否	字符	+success+:成功 +value_error+:值错误
rptDec	否	字符	+success+:成功 +value_error+:值错误
addr	否	字符	+success+:成功 +value_error+:值错误
cmd	否	字符	+success+:成功 +value_error+:值错误

6.2.1.示例 1

写入系统 Basic：

```
{
    "msgType": "setSysBasic",
    "devID": "466C210730024164",
    "data": {
        "pw": "abc123456",
        "format": 2,
        "rptDec": 1,
        "addr": "55"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSysBasicAck",
    "data": {
        "pw": "+success+",
        "format": "+success+",
        "rptDec": "+success+",
        "addr": "+success+"
    },
    "timestamp": "0"
}
```

6.2.2.示例 2

写入系统 Basic：

```
{
    "msgType": "setSysBasic",
    "devID": "466C210730024164",
    "data": {
        "pw": "abc123456",
        "format": 4,
        "rptDec": 1,
        "addr": "55"
    }
}
```

响应:

```
{
```

```
    "devID": "466C210730024164",
    "msgType": "setSysBasicAck",
    "data": {
        "pw": "+success+",
        "format": "+value_error+",
        "rptDec": "+success+",
        "addr": "+success+"
    },
    "timestamp": "1628760182"
}
```


6.3.读取组网参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysGroup
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSysGroupAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
en	是	字符	组能使能。65535:启用 0:禁用
type	是	字符	组类型。1:Type A 2:Type B
id	是	字符	组 ID。最大 20 位字符
pw	是	字符	组密码。最大 20 位字符

6.3.1 示例 1

获取组网参数:


```
{
    "msgType": "getSysGroup",
    "devID": "466C210730024164",
    "data": {}
}
```

响应：


```
{
    "devID": "466C210730024164",
    "msgType": "getSysGroupAck",
    "data": {
        "en": 0,
        "type": 0,
        "id": "123",
        "pw": "123"
    },
    "timestamp": "1628760243"
}
```


6.4.写入组网参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysGroup
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
en	否	字符	组能使能。65535:启用 0:禁用
type	否	字符	组类型。1:Type A 2:Type B
id	否	字符	组 ID。最大 20 位字符
pw	否	字符	组密码。最大 20 位字符

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSysGroupAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
en	否	字符	+success+:成功 +value_error+:值错误
type	否	字符	+success+:成功 +value_error+:值错误
id	否	字符	+success+:成功 +length_error+:长度错误

pw	否	字符	+success+:成功 +length_error+:长度错误

6.4.1 示例 1

写入组网参数：

```
{
    "msgType": "setSysGroup",
    "devID": "466C210730024164",
    "data": {
        "en": 1,
        "type": 1,
        "id": "zhc123",
        "pw": "zhc123"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSysGroupAck",
    "data": {
        "en": "+success+",
        "type": "+success+",
        "id": "+success+",
        "pw": "+success+"
    },
    "timestamp": "1628760306"
}
```

6.4.2 示例 2

写入组网参数：

```
{
    "msgType": "setSysGroup",
    "devID": "466C210730024164",
    "data": {
        "en": 1,
        "type": 3,
        "id": "zhc123",
        "pw": "zhc123"
    }
```

```
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSysGroupAck",
    "data": {
        "en": "+success+",
        "type": "+value_error+",
        "id": "+success+",
        "pw": "+success+"
    },
    "timestamp": "1628760336"
}
```

6.5.读取定位信息

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLocation
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLocationAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
cyc	是	字符	周期上报。单位:s/秒
location			定位信息。最大 100 位字符

6.3.1 示例

读取定位信息：

```
{
  "msgType": "getLocation",
  "devID": "466C210730024164",
  "data": {}
```

```
}
```

响应：

```
{
   "devID": "466C210730024164",
   "msgType": "getLocationAck",
   "data": {
      "cyc": "60",
      "location":"GPGGA,121252.000,3937.3032,N,11611.6046,E,1,05,2.0,45.9,M,-
5.7,M,,0000*77$GPRMC,121252.000,A,3958.3032,N,11629.6046,E,15.15,359.95,070306,,,A*54
$"
   },
   "timestamp": "1628131528"
}
```

6.6.写入定位信息

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setLocation
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
cyc	否	字符	周期上报。单位:s/秒
location	否	字符	定位信息。最大 100 位字符

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setLocationAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
cyc	否	字符	+success+:成功
location	否	字符	+success+:成功 +length_error+:长度错误

6.6.1 示例 1

写入定位信息：

{
  "msgType": "setLocation",
  "devID": "466C210730024164",
  "data": {
    "cyc": "60",
    "location":"abc"
  }
}

响应：

{
    "devID": "466C210730024164",
    "msgType": "setLocationAck",
    "data": {
      "cyc": "+success+",
      "location": "+success+"
    },
    "timestamp": "1628561809"
}

6.6.2 示例 2

写入定位信息：

{
    "msgType": "setLocation",
    "devID": "466C210730024164",
    "data": {
      "cyc": "60",
      "location":
"abcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabcabc
abcabcabcabcabcabcabcabcabcabcabcabcabcabcabc"
    }
}

响应：

{
    "devID": "466C210730024164",
    "msgType": "setLocationAck",
    "data": {
      "cyc": "+success+",
      "location": "+length_error+"

```
    },
    "timestamp": "1628561892"
}
```

7.网络

7.1.读取网络基础参数

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getLTEBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getLTEBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
sim	是	字符	SIM 卡号
signal	是	字符	信号强度

7.1.1 示例

读取 LTE 基础参数：

```
{
  "msgType": "getLTEBasic ",
  "devID": "466C210730024164",
  "data": {}
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "getLTEBasicAck",
    "data": {
```

```
            "sim": "89860474282090004416",
            "signal": "\"LTE\",94,-118,125,-7"
        },
        "timestamp": "1628562848"
}
```


7.2.读取 APN

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getAPN
devID	是	字符	设备唯一识别码，16 位
data	是	字符	空

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getAPNAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
addr	是	字符	APN 地址
user	是	字符	APN 用户名
pass	是	字符	APN 密码

7.2.1 示例

读取 APN：


```
{
   "msgType": "getAPN",
   "devID": "466C210730024164",
   "data": {}
}
```


响应：


```
{
   "devID": "466C210730024164",
   "msgType": "getAPNAck",
   "data": {
      "addr": "cnnet",
```

```
        "user": "cnnet",
        "pass": "cnnet"
    },
    "timestamp": "1628131528"
}
```


7.3.写入 APN

请求帧格式：

字段	属性	必须	类型	描述
msgType	消息类型	是	字符	setAPN
devID	设备 ID	是	字符	设备唯一识别码，16 位
data	参数根节点	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	属性	必须	类型	描述
addr	设备 ID	否	字符	APN 地址
user	设备密码	否	字符	APN 用户名
pass	固件版本	否	字符	APN 密码

响应帧格式：

字段	属性	必须	类型	描述
msgType	消息类型	是	字符	setAPNAck
devID	设备 ID	是	字符	设备唯一识别码，16 位
data	参数根节点	是	字符	参数根节点，详见 data 帧格式
timestamp	时间戳	是	字符	北京时间

data 帧格式

字段	属性	必须	类型	描述
addr	设备 ID	否	字符	+success+:成功 +length_error+:长度错误
user	设备密码	否	字符	+success+:成功 +length_error+:长度错误
pass	固件版本	否	字符	+success+:成功 +length_error+:长度错误

7.3.1 示例 1

写入 APN：


```
{
```

```
    "msgType": "setAPN",
    "devID": "466C210730024164",
    "data": {
        "addr": "cnnet",
        "user": "cnnet",
        "pass": "cnnet"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAPNAck",
    "data": {
        "addr": "+success+",
        "user": "+success+",
        "pass": "+success+"
    },
    "timestamp": "1628562974"
}
```

7.3.2 示例 2

写入 APN：

```
{
    "msgType": "setAPN",
    "devID": "466C210730024164",
    "data": {
        "addr": "cnnetcnnetcnnetcnnetcnnetcnnetcnnetcnnet",
        "user": "cnnetcnnetcnnetcnnetcnnetcnnetcnnetcnnetcnnet",
        "pass": "cnnetcnnetcnnetcnnetcnnetcnnetcnnetcnnetcnnet"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setAPNAck",
    "data": {
        "addr": "+length_error+",
        "user": "+length_error+",
        "pass": "+length_error+"
    },
    "timestamp": "1628563029"
```

}

7.4.读取 SOCKET 信息

7.4.1.读取 SOCKET 状态、模式、目的 IP

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号 1~2

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2
status	是	字符	状态。65535:启用 0:禁用
mode	是	字符	工作模式。1:TCP Client 4:MQTT Client(仅 Socket1)
desIP	是	字符	目的地址
desPort	是	字符	目的端口

7.4.1.1 示例

读取定位信息：

{

```
    "msgType": "getSocketBasic",
    "devID": "466C210730024164",
    "data": {
        "index":"1"
    }
}
```

响应：

```
{
        "devID": "466C210730024164",
        "msgType": "getSocketBasicAck",
        "data": {
                "index": "1",
                "status": 1,
                "mode": 0,
                "desIP": "cloud.iotrouter.cn",
                "desPort": "56000"
        },
        "timestamp": "1628760428"
}
```

7.4.2.写入 SOCKET 状态、模式、目的 IP

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号。1~2
status	否	字符	状态。65535:启用 0:禁用
mode	否	字符	工作模式。1:TCP Client 4:MQTT Client(仅 Socket1)
desIP	否	字符	目的地址
desPort	否	字符	目的端口

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2
status	否	字符	+success+:成功 +value_error+:值错误
mode	否	字符	+success+:成功 +value_error+:值错误
desIP	否	字符	+success+:成功 +length_error+:长度错误
desPort	否	字符	+success+:成功

7.4.2.1 示例 1

写入组网参数：


```
{
    "devID": "466C210730024164",
    "msgType": "setSocketBasic",
    "data": {
        "index": "1",
        "status": 1,
        "mode": 1,
        "desIP": "cloud.iotrouter.cn",
        "desPort": "56000"
    },
    "timestamp": "1628760428"
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setSocketBasicAck",
    "data": {
        "index": "1",
        "status": "+success+",
        "mode": "+success+",
        "desIP": "+success+"
    },
    "timestamp": "0"
}
```

7.4.2.2 示例 2

写入组网参数：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketBasic",
    "data": {
        "index": "2",
        "status": 1,
        "mode": 1,
        "desIP": "cloud.iotrouter.cn",
        "desPort": "56000"
    },
    "timestamp": "1628760428"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketBasicAck",
    "data": {
        "index": "2",
        "status": "+success+",
        "mode": "+value_error+",
        "desIP": "+success+"
    },
    "timestamp": "1628760724"
}
```

7.4.3.读取 SOCKET 身份包

请求帧格式：

字段	属性	必须	类型	描述
msgType	消息类型	是	字符	getSocketID
devID	设备 ID	是	字符	设备唯一识别码，16 位
data	参数根节点	是	字符	data 帧格式

data 帧格式

字段	属性	必须	类型	描述
index		是	字符	socket 编号 1~2

响应帧格式：

字段	属性	必须	类型	描述
msgType	消息类型	是	字符	getSocketIDAck

devID	设备 ID	是	字符	设备唯一识别码，16 位
data	参数根节点	是	字符	参数根节点，详见 data 帧格式
timestamp	时间戳	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2
mode	是	字符	身份包模式。0:禁用 1:默认 2:自定义 3:ID 4:卡号
cdt	是	字符	身份包触发模式。1:连接发送 2:数据携带 3:以上两种
pkg	是	字符	身份包内容。16 进制字符串。例如“123”，则 regPkg 为“313233”

7.4.1.1 示例

读取定位信息：

```
{
   "msgType": "getSocketID",
   "devID": "466C210730024164",
   "data": {
      "index":"1"
   }
}
```

响应：

```
{
      "devID": "466C210730024164",
      "msgType": "getSocketIDAck",
      "data": {
            "index": "1",
            "mode": 1,
            "cdt": 0,
            "pkg":
"415F315F312E343636433231303733303032343136342E59574A6A4D54497A4E4455322E353
532666363616461646561643664663135613734316437353931633830306 52E414243444531323
33435"
      },
      "timestamp": "1628760920"
}
```

7.4.4.写入 SOCKET 身份包

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketID
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号。1~2
mode	否	字符	身份包模式。0:禁用 1:默认 2:自定义 3:ID 4:卡号
cdt	否	字符	身份包触发模式。1:连接发送 2:数据携带 3:以上两种
pkg	否	字符	身份包内容。16 进制字符串。例如“123”，则 regPkg 为“313233”

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketIDAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2
mode	否	字符	+success+:成功 +value_error+:值错误
cdt	否	字符	+success+:成功 +value_error+:值错误
pkg	否	字符	+success+:成功 +length_error+:长度错误

7.4.2.1 示例 1

写入 Socket 身份包：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketID",
    "data": {
        "index": "1",
        "mode": 3,
        "cdt": 0,
        "pkg": "12313123131"
    },
    "timestamp": "1628760920"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketIDAck",
    "data": {
        "index": "1",
        "mode": "+success+",
        "cdt": "+success+",
        "pkg": "+success+"
    },
    "timestamp": "1628761004"
}
```

7.4.2.2 示例 2

写入 Socket 身份包：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketID",
    "data": {
        "index": "1",
        "mode": 3,
        "cdt": 4,
        "pkg": "12313123131"
    },
    "timestamp": "1628760920"
}
```

响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setSocketIDAck",
    "data": {
        "index": "1",
        "mode": "+success+",
        "cdt": "+value_error+",
        "pkg": "+success+"
    },
    "timestamp": "1628761075"
}
```


7.4.5.读取 SOCKET 心跳包

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号 1~2

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketHbtAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2

mode	是	字符	心跳包模式。0:禁用 1:默认 2:自定义 3:ID 4:卡号
cyc	是	字符	心跳包周期。单位: s/秒
pkg	是	字符	心跳包内容。16 进制字符串。例如“123”，则 regPkg 为“313233”

7.4.5.1 示例

读取 Socket Hbt：

```
{
   "msgType": "getSocketHbt",
   "devID": "466C210730024164",
   "data": {
      "index":"1"
   }
}
```

响应：

```
{
      "devID": "466C210730024164",
      "msgType": "getSocketHbtAck",
      "data": {
            "index": "1",
            "mode": 1,
            "cyc": "30",
            "pkg": "50494E472150494E472150494E4721"
      },
      "timestamp": "1628761112"
}
```

7.4.6.写入 SOCKET 心跳包

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketHbt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述

index	是	字符	socket 编号。1~2
mode	否	字符	心跳包模式。0:禁用 1:默认 2:自定义 3:ID 4:卡号
cyc	否	字符	心跳包周期。单位: s/秒
pkg	否	字符	心跳包内容。16 进制字符串。例如“123”，则 regPkg 为“313233”

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketHbtAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1~2
mode	否	字符	+success+:成功 +value_error+:值错误
cyc	否	字符	+success+:成功
pkg	否	字符	+success+:成功 +length_error+:长度错误

7.4.6.1 示例 1

写入组网参数：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketHbt",
    "data": {
        "index": "1",
        "mode": 1,
        "cyc": "30",
        "pkg": "ping!ping!ping!"
    },
    "timestamp": "1628761112"
}
```

响应：

```
{
    "devID": "466C210730024164",
```

```
    "msgType": "setSocketHbtAck",
    "data": {
        "index": "1",
        "mode": "+success+",
        "cyc": "+success+",
        "pkg": "+success+"
    },
    "timestamp": "1628565027"
}
```


7.4.6.2 示例 2

写入 Socket Hbt：


```
{
    "devID": "466C210730024164",
    "msgType": "setSocketHbt",
    "data": {
        "index": "1",
        "mode": 5,
        "cyc": "30",
        "pkg":
"50494E472150494E472150494E472150494E472150494E472150494E472150494E472150494E472150494E4721"
    },
    "timestamp": "1628761112"
}
```


响应：


```
{
    "devID": "466C210730024164",
    "msgType": "setSocketHbtAck",
    "data": {
        "index": "1",
        "mode": "+value_error+",
        "cyc": "+success+",
        "pkg": "+length_error+"
    },
    "timestamp": "1628761135"
}
```

7.4.7.读取 SOCKET MQTT Basic

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketMqttBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号 1

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketMqttBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1
id	是	字符	客户端 ID
name	是	字符	用户名
pw	是	字符	密码
alive	是	字符	保活时间。单位: s/秒

7.4.7.1 示例

读取 Mqtt Basic：

```
{
   "msgType": "getSocketMqttBasic",
   "devID": "466C210730024164",
   "data": {
      "index":"1"
   }
}
```

响应：

```
{
      "devID": "466C210730024164",
```

```
    "msgType": "getSocketMqttBasicAck",
    "data": {
        "index": "1",
        "id": "",
        "name": "",
        "pw": "",
        "alive": "30"
    },
    "timestamp": "0"
}
```


7.4.8.写入 SOCKET MQTT Basic

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketMqttBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号。1
id	否		客户端 ID
name	否		用户名
pw	否		密码
alive	否		保活时间。单位: s/秒

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketMqttBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1
id	否	字符	+success+:成功 +length_error+:长度错误
name	否	字符	+success+:成功

			+length_error+:长度错误
pw	否	字符	+success+:成功 +length_error+:长度错误
alive	否	字符	+success+:成功

7.4.8.1 示例 1

写入组网参数：

```
{
    "msgType": "setSocketMqttBasic",
    "devID": "466C210730024164",
    "data": {
        "index": "1",
        "id": "466C210730024164",
        "name": "466C",
        "pw": "123456",
        "alive": "120"
    }
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketMqttAck",
    "data": {
        "index": "1",
        "id": "+success+",
        "name": "+success+",
        "pw": "+success+",
        "alive": "+success+"
    },
    "timestamp": "1628565319"
}
```

7.4.8.2 示例 2

写入组网参数：

```
{
    "msgType": "setSocketMqttBasic",
    "devID": "466C210730024164",
    "data": {
        "index": "1",
        "id":
```

"466C210730024164466C210730024164466C210730024164466C210730024164466C210730024164",
"name": "466C210730024164466C210730024164466C210730024164466C210730024164466C210730024164",
"pw": "466C210730024164466C210730024164466C210730024164466C210730024164466C210730024164",
"alive": "120"
}
}

响应：

{
"devID": "466C210730024164",
"msgType": "setSocketMqttBasicAck",
"data": {
"index": "1",
"id": "+length_error+",
"name": "+length_error+",
"pw": "+length_error+",
"alive": "+success+"
},
"timestamp": "1628565845"
}

7.4.9.读取 SOCKET MQTT Topic

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketMqttTopic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号 1

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	getSocketMqttTopicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1
id	是	字符	客户端 ID
name	是	字符	用户名
pw	是	字符	密码
alive	是	字符	保活时间。单位: s/秒

7.4.7.1 示例

读取 Mqtt Topic：

```
{
   "msgType": "getSocketMqttTopic",
   "devID": "466C210730024164",
   "data": {
      "index":"1"
   }
}
```

响应：

```
{
      "devID": "466C210730024164",
      "msgType": "getSocketMqttTopicAck",
      "data": {
            "index": "1",
            "sub": "/public/zhcpub",
            "pub": "/public/zhcsub"
      },
      "timestamp": "1628565980"
}
```

7.4.10.写入 SOCKET MQTT Topic

请求帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketMqttBasic
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
index	是	字符	socket 编号。1
sub	否	字符	订阅主题
pub	否	字符	发布主题

响应帧格式：

字段	必须	类型	描述
msgType	是	字符	setSocketMqttBasicAck
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式
timestamp	是	字符	北京时间

data 帧格式

字段	必须	类型	描述
index	是	字符	+value_error+:值错误 socket 编号。1
sub	否	字符	+success+:成功 +length_error+:长度错误
pub	否	字符	+success+:成功 +length_error+:长度错误

7.4.10.1 示例 1

写入 Mqtt Topic：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketMqttTopic",
    "data": {
        "index": "1",
        "sub": "/public/zhcpub",
        "pub": "/public/zhcsub"
    },
    "timestamp": "1628565980"
}
```

响应：

```
{
    "devID": "466C210730024164",
    "msgType": "setSocketMqttTopicAck",
    "data": {
        "index": "1",
        "sub": "+success+",
```

```
            "pub": "+success+"
        },
        "timestamp": "1628566022"
}
```


7.4.10.2 示例 2

写入 Mqtt Topic：


```
{
        "devID": "466C210730024164",
        "msgType": "setSocketMqttTopic",
        "data": {
            "index": "1",
            "pub":
"/public/zhcsub/12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890"
        },
        "timestamp": "1628565980"
}
```


响应：


```
{
        "devID": "466C210730024164",
        "msgType": "setSocketMqttTopicAck",
        "data": {
            "index": "1",
            "pub": "+length_error+"
        },
        "timestamp": "1628566107"
}
```

8.数据主动上报

8.1.串口数据上报

上报帧格式：

字段	必须	类型	描述
msgType	是	字符	uartValueRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
payload	是	字符	16 进制字符串。例如：要发送“123”，则 payload 为“313233”

8.1.1 示例

```
{
    "devID": "466C210730024164",
    "msgType": "uartValueRpt",
    "data": {
        "payload": "3231333231333323133"
    },
    "timestamp": "1628579518"
}
```

8.2.AI 数据主动上报

上报帧格式：

字段	必须	类型	描述
msgType	是	字符	AIValueRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
AI1	是	字符	0~65535uA/mV
AI2	是	字符	0~65535uA/mV

AI3	是	字符	0~65535uA/mV
AI4	是	字符	0~65535uA/mV

8.2.1 示例

```
{
    "devID": "466C210730024164",
    "msgType": "AIValueRpt",
    "data": {
        "AI1": "43",
        "AI2": "39",
        "AI3": "39",
        "AI4": "41"
    },
    "timestamp": "1628579581"
}
```

8.3.AO 数据主动上报

上报帧格式：

字段	必须	类型	描述
msgType	是	字符	AOValueRpt
devID	是	字符	设备唯一识别码，16 位
data	是	字符	参数根节点，详见 data 帧格式

data 帧格式

字段	必须	类型	描述
AO1	是	字符	0~65535uA/mV
AO2	是	字符	0~65535uA/mV
AO3	是	字符	0~65535uA/mV
AO4	是	字符	0~65535uA/mV

8.3.1 示例

```
{
    "devID": "466C210730024164",
    "msgType": "AOValueRpt",
    "data": {
        "AO1": "3000",
        "AO2": "0",
        "AO3": "0",
        "AO4": "0"
```

```
    },
    "timestamp": "1628579964"
}
```