CAT1 系列 JSON

应用指导

LTE Cat 1 Modbus RTU

版本：CAT1 系列_JSON_应用指导_V1.1

日期：2020-08-10

更新历史

版本	日期	作者	变更描述
1.0	2020-08-04	LXL	Initial

目　录

Content

1.通用说明

本文档适用于 ZHC-CAT1 系列产品，支持 TCP 和 MQTT 两种不同通信模式下的 JSON 协议交互。客户在使用 JSON 协议和设备通信时，请严格遵循本文档格式要求。

本文档对 JSON 协议格式规范不做赘述。

本文档 JSON 所有字段的数据类型均为字符串-String。

本文档适用于 ZHC-CAT1 系列多款产品，每款产品接口数量不同，相应的协议会有略微差异，客户使用过程中，需根据目标产品接口数量调整参数。

不同的消息用“msgType”来区分，即不同的消息“msgType”不同。

设备上行数据中，会携带“devId”字段和“timestamp”字段，示例中不在一一列出。“timestamp”依赖于网络时间，获取需要一定的时间，因此不能完全依赖，仅供参考。

支持一次下发设置多个参数，如果下发参数中某个参数错误，设备会返回错误码。但是其余正确参数依然会执行。因此用户需注意各个参数的取值范围，保证参数的准确性。

2.DO 接口

DO 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getDoValue	服务器->设备	请求 DO 状态
getDoValueAck	设备->服务器	请求 DO 状态的回复
setDoValue	服务器->设备	控制 DO 状态
setDoValueAck	设备->服务器	控制 DO 状态的回复
getDoConfig	服务器->设备	获取 DO 配置信息
getDoConfigAck	设备->服务器	获取 DO 配置信息的回复
setDoConfig	服务器->设备	修改 DO 配置信息
setDoConfigAck	设备->服务器	修改 DO 配置信息的回复
doValueRpt	设备->服务器	DO 参数主动上报

2.1.请求 DO 状态

请求帧格式：

字段	是否必须	描述
msgType	是	getDoValue
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getDoValueAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
DO1	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO2	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO3	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO4	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO5	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO6	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO7	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO8	硬件接口数量决定	1：继电器吸合 0：继电器断开

示例：

```
{
    "msgType": "getDoValue",
    "data": ""
}
{
    "msgType": "getDoValueAck",
    "data": {
        "DO1": "0",
        "DO2": "1",
        "DO3": "0",
        "DO4": "1",
        "DO5": "0",
        "DO6": "1",
        "DO7": "0",
        "DO8": "1"
    }
}
```

注：指令下发后立即生效。

2.2.控制 DO 状态

请求帧格式：

字段	是否必须	描述
msgType	是	setDoValue
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
DO1	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO2	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO3	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO4	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO5	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO6	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO7	硬件接口数量决定	1：继电器吸合 0：继电器断开
DO8	硬件接口数量决定	1：继电器吸合 0：继电器断开

响应帧格式：

字段	是否必须	描述
msgType	是	setDoValueAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setDoValue",
    "data": {
        "DO1": "0",
        "DO2": "1",
        "DO3": "0",
        "DO4": "1",
```

```
            "DO5": "0",
            "DO6": "1",
            "DO7": "0",
            "DO8": "1"
        }
}
{
    "msgType": "setDoValueAck",
    "data":"0"
}

{
    "msgType":"setDoValue",
    "data": {
            "DO1": "1",
            "DO2": "3",
            "DO3": "0",
            "DO4": "1",
            "DO5": "0",
            "DO6": "1",
            "DO7": "0",
            "DO8": "1"
        }
}
{
    "msgType": "setDoValueAck",
    "data":"1"
}
```


注：指令下发后立即生效。

注：可单独控制某一路或某几路继电器。

2.3.获取 DO 配置

请求帧格式：

字段	是否必须	描述
msgType	是	getDoConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getDoConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
enRpt	否	1：启用主动上报　0：关闭主动上报
rstStatus	否	1：掉电保持 DO 状态　0：掉电不保持
holdTime1	否	0-65535，继电器 1 输出保持时间，单位 秒
holdTime2	否	0-65535，继电器 2 输出保持时间，单位 秒
holdTime3	否	0-65535，继电器 3 输出保持时间，单位 秒
holdTime4	否	0-65535，继电器 4 输出保持时间，单位 秒
holdTime5	否	0-65535，继电器 5 输出保持时间，单位 秒
holdTime6	否	0-65535，继电器 6 输出保持时间，单位 秒
holdTime7	否	0-65535，继电器 7 输出保持时间，单位 秒
holdTime8	否	0-65535，继电器 8 输出保持时间，单位 秒
turnTime1	否	0-65535，继电器 1 周期翻转时间，单位 秒
turnTime2	否	0-65535，继电器 2 周期翻转时间，单位 秒
turnTime3	否	0-65535，继电器 3 周期翻转时间，单位 秒
turnTime4	否	0-65535，继电器 4 周期翻转时间，单位 秒
turnTime5	否	0-65535，继电器 5 周期翻转时间，单位 秒
turnTime6	否	0-65535，继电器 6 周期翻转时间，单位 秒
turnTime7	否	0-65535，继电器 7 周期翻转时间，单位 秒
turnTime8	否	0-65535，继电器 8 周期翻转时间，单位 秒

示例：


```
{
    "msgType": "getDoConfig",
    "data":""
}
{
    "msgType": "getDoConfigAck",
    "data": {
        "enRpt": "1",
        "rstStatus": "0",
        "holdTime1": "0",
        "holdTime2": "0",
        "holdTime3": "0",
        "holdTime4": "0",
        "holdTime5": "0",
        "holdTime6": "0",
        "holdTime7": "0",
        "holdTime8": "0",
        "turnTime1": "0",
        "turnTime2": "0",
        "turnTime3": "0",
        "turnTime4": "0",
        "turnTime5": "0",
        "turnTime6": "0",
        "turnTime7": "0",
        "turnTime8": "0"
    }
}
```


注：指令下发后立即生效。

2.4.修改 DO 配置

请求帧格式：

字段	是否必须	描述
msgType	是	setDoConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
enRpt	否	1：启用主动上报　0：关闭主动上报
rstStatus	否	1：掉电保持 DO 状态　0：掉电不保持
holdTime1	否	0-65535，继电器 1 输出保持时间，单位 秒
holdTime2	否	0-65535，继电器 2 输出保持时间，单位 秒
holdTime3	否	0-65535，继电器 3 输出保持时间，单位 秒
holdTime4	否	0-65535，继电器 4 输出保持时间，单位 秒
holdTime5	否	0-65535，继电器 5 输出保持时间，单位 秒
holdTime6	否	0-65535，继电器 6 输出保持时间，单位 秒
holdTime7	否	0-65535，继电器 7 输出保持时间，单位 秒
holdTime8	否	0-65535，继电器 8 输出保持时间，单位 秒
turnTime1	否	0-65535，继电器 1 周期翻转时间，单位 秒
turnTime2	否	0-65535，继电器 2 周期翻转时间，单位 秒
turnTime3	否	0-65535，继电器 3 周期翻转时间，单位 秒
turnTime4	否	0-65535，继电器 4 周期翻转时间，单位 秒
turnTime5	否	0-65535，继电器 5 周期翻转时间，单位 秒
turnTime6	否	0-65535，继电器 6 周期翻转时间，单位 秒
turnTime7	否	0-65535，继电器 7 周期翻转时间，单位 秒
turnTime8	否	0-65535，继电器 8 周期翻转时间，单位 秒

响应帧格式：

字段	是否必须	描述
msgType	是	setDoConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setDoConfig",
    "data": {
```

```
            "enRpt": "1",
            "rstStatus": "1",
            "holdTime1": "0",
            "holdTime2": "0",
            "holdTime3": "0",
            "holdTime4": "0",
            "holdTime5": "0",
            "holdTime6": "0",
            "holdTime7": "0",
            "holdTime8": "0",
            "turnTime1": "0",
            "turnTime2": "0",
            "turnTime3": "0",
            "turnTime4": "0",
            "turnTime5": "0",
            "turnTime6": "0",
            "turnTime7": "0",
            "turnTime8": "0"
        }
    }
    {
        "msgType": "setDoConfigAck",
        "data":"0"
    }

    {
        "msgType":"setDoConfig",
        "data": {
            "enRpt": "3",
            "rstStatus": "1",
            "holdTime1": "0",
            "holdTime2": "0",
            "holdTime3": "0",
            "holdTime4": "0",
            "holdTime5": "0",
            "holdTime6": "0",
            "holdTime7": "0",
            "holdTime8": "0",
            "turnTime1": "0",
            "turnTime2": "0",
            "turnTime3": "3",
            "turnTime4": "0",
            "turnTime5": "0",
            "turnTime6": "0",
```

```
            "turnTime7": "0",
            "turnTime8": "0"
        }
}
{
    "msgType": "setDoConfigAck",
    "data":"1"
}
```

注：指令下发后立即生效。

2.5.DO 主动上报

主动上报帧格式：

字段	是否必须	描述
msgType	是	doValueRpt
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
DO1	接口数量决定	1：继电器吸合　0：继电器断开
DO2	接口数量决定	1：继电器吸合　0：继电器断开
DO3	接口数量决定	1：继电器吸合　0：继电器断开
DO4	接口数量决定	1：继电器吸合　0：继电器断开
DO5	接口数量决定	1：继电器吸合　0：继电器断开
DO6	接口数量决定	1：继电器吸合　0：继电器断开
DO7	接口数量决定	1：继电器吸合　0：继电器断开
DO8	接口数量决定	1：继电器吸合　0：继电器断开

示例：

```
{
    "msgType": "doValueRpt",
    "data": {
        "DO1": "0",
        "DO2": "1",
        "DO3": "0",
```

```
            "DO4": "1",
            "DO5": "0",
            "DO6": "1",
            "DO7": "0",
            "DO8": "1"
        }
    }
```

注：主动上报指令服务器无需回复。

3.DI 接口

DI 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getDiValue	服务器->设备	请求 DI 状态
getDiValueAck	设备->服务器	请求 DI 状态的回复
getDiConfig	服务器->设备	获取 DI 配置信息
getDiConfigAck	设备->服务器	获取 DI 配置信息的回复
setDiConfig	服务器->设备	修改 DI 配置信息
setDiConfigAck	设备->服务器	修改 DI 配置信息的回复
diValueRpt	服务器->设备	DI 参数主动上报

3.1.请求 DI 状态

请求帧格式：

字段	是否必须	描述
msgType	是	getDiValue
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getDiValueAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
DI1	硬件接口数量决定	1：有输入信号　0：无输入信号

DI2	硬件接口数量决定	1：有输入信号　0：无输入信号
DI3	硬件接口数量决定	1：有输入信号　0：无输入信号
DI4	硬件接口数量决定	1：有输入信号　0：无输入信号
DI5	硬件接口数量决定	1：有输入信号　0：无输入信号
DI6	硬件接口数量决定	1：有输入信号　0：无输入信号
DI7	硬件接口数量决定	1：有输入信号　0：无输入信号
DI8	硬件接口数量决定	1：有输入信号　0：无输入信号

示例：

```
{
    "msgType":"getDiValue",
    "data":""
}
{
    "msgType": "getDiValueAck",
    "data": {
        "DI1": "0",
        "DI2": "1",
        "DI3": "0",
        "DI4": "1",
        "DI5": "0",
        "DI6": "1",
        "DI7": "0",
        "DI8": "1"
    }
}
```

注：指令下发后立即生效。

3.2.获取 DI 配置

请求帧格式：

字段	是否必须	描述
msgType	是	getDiConfigAck
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getDiConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
enRpt	否	1：开启周期上报　0：关闭周期上报
cyc	否	0-65535：DI 状态上报周期，单位 秒

示例：

```
{
    "msgType": "getDiConfig",
    "data":""
}
{
    "msgType": "getDiConfigAck",
    "data": {
        "enRpt": "0",
        "cyc": "10"
    }
}
```

注：指令下发后立即生效。

3.3.修改 DI 配置

请求帧格式：

字段	是否必须	描述
msgType	是	setDiConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述

enRpt	否	1：启用主动上报　0：关闭主动上报
cyc	否	0-65535：DI 状态上报周期，单位 秒

响应帧格式：

字段	是否必须	描述
msgType	是	setDiConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setDiConfig",
    "data": {
        "enRpt": "1",
        "cyc": "10"
    }
}
{
    "msgType": "setDiConfigAck",
    "data":"0"
}

{
    "msgType":"setDiConfig",
    "data": {
        "enRpt": "3",
        "cyc": "10"
    }
}
{
    "msgType": "setDiConfigAck",
    "data":"1"
}
```

注：指令下发后立即生效。

3.4.DI 主动上报

主动上报帧格式：

字段	是否必须	描述
msgType	是	diValueRpt
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
DI1	接口数量决定	1：有输入信号　0：无输入信号
DI2	接口数量决定	1：有输入信号　0：无输入信号
DI3	接口数量决定	1：有输入信号　0：无输入信号
DI4	接口数量决定	1：有输入信号　0：无输入信号
DI5	接口数量决定	1：有输入信号　0：无输入信号
DI6	接口数量决定	1：有输入信号　0：无输入信号
DI7	接口数量决定	1：有输入信号　0：无输入信号
DI8	接口数量决定	1：有输入信号　0：无输入信号

示例：

```
{
    "msgType": "diValueRpt",
    "data": {
        "DI1": "0",
        "DI2": "1",
        "DI3": "0",
        "DI4": "1",
        "DI5": "0",
        "DI6": "1",
        "DI7": "0",
        "DI8": "1"
    }
}
```

注：主动上报指令服务器无需回复。

4.PI 接口

PI 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getPiValue	服务器->设备	请求 PI 状态
getPiValueAck	设备->服务器	请求 PI 状态的回复
setPiConfig	服务器->设备	清空 PI 脉冲数据
setPiConfigAck	设备->服务器	清空 PI 脉冲数据

4.1.请求 PI 计数

请求帧格式：

字段	是否必须	描述
msgType	是	getPiValue
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getPiValueAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
PI1	硬件接口数量决定	0-65535 脉冲 1 计数信号
PI2	硬件接口数量决定	0-65535 脉冲 2 计数信号

示例：

```
{
    "msgType":"getPiValue",
    "data":""
}
{
    "msgType": "getPiValueAck",
    "data": {
        "PI1": "0-65535",
        "PI2": "0-65535"
    }
}
```

注：指令下发后立即生效。

4.2.清空 PI 计数

请求帧格式：

字段	是否必须	描述
msgType	是	setPiConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
resetPi1	否	1：清空
resetPi2	否	1：清空

响应帧格式：

字段	是否必须	描述
msgType	是	setPiConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setPiConfig",
    "data": {
        "resetPi1": "1",
        "resetPi2": "1"
```

```
        }
    }
    {
        "msgType": "setPiConfigAck",
        "data":"0"
    }

    {
        "msgType":"setPiConfig",
        "data": {
            "resetPi1": "2",
            "resetPi2": "1"
        }
    }
    {
        "msgType": "setPiConfigAck",
        "data":"1"
    }
```

注：指令下发后立即生效。

5.AI 接口

AI 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getAiValue	服务器->设备	请求 AI 状态
getAiValueAck	设备->服务器	请求 AI 状态的回复
getAiConfig	服务器->设备	获取 AI 配置信息
getAiConfigAck	设备->服务器	获取 AI 配置信息的回复
setAiConfig	服务器->设备	修改 AI 配置信息
setAiConfigAck	设备->服务器	修改 AI 配置信息的回复
aiValueRpt	服务器->设备	AI 参数主动上报

5.1.请求 AI 状态

请求帧格式：

字段	是否必须	描述
msgType	是	getAiValue
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getAiValueAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
AI1	硬件接口数量决定	0-20000 AI1 输入电流值，单位 uA

AI2	硬件接口数量决定	0-20000 AI2 输入电流值，单位 uA
AI3	硬件接口数量决定	0-20000 AI3 输入电流值，单位 uA
AI4	硬件接口数量决定	0-20000 AI4 输入电流值，单位 uA
AI5	硬件接口数量决定	0-20000 AI5 输入电流值，单位 uA
AI6	硬件接口数量决定	0-20000 AI6 输入电流值，单位 uA
AI7	硬件接口数量决定	0-20000 AI7 输入电流值，单位 uA
AI8	硬件接口数量决定	0-20000 AI8 输入电流值，单位 uA

示例：

```
{
    "msgType":"getAiValue",
    "data":""
}
{
    "msgType": "getAiValueAck",
    "data": {
        "AI1": "4000",
        "AI2": "5000",
        "AI3": "6000",
        "AI4": "7000",
        "AI5": "8000",
        "AI6": "9000",
        "AI7": "10000",
        "AI8": "20000"
    }
}
```

注：指令下发后立即生效。

5.2.请求 AI 配置

请求帧格式：

字段	是否必须	描述
msgType	是	getAiConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getAiConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
enRpt	否	1：开启周期上报　0：关闭周期上报
cyc	否	0-65535：DI 状态上报周期，单位 秒
rptRule1	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin1	否	0-20000，阈值最小值，单位 uA
rptMax1	否	0-20000，阈值最大值，单位 uA
rptRule2	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin2	否	0-20000，阈值最小值，单位 uA
rptMax2	否	0-20000，阈值最大值，单位 uA
rptRule3	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin3	否	0-20000，阈值最小值，单位 uA
rptMax3	否	0-20000，阈值最大值，单位 uA
rptRule4	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin4	否	0-20000，阈值最小值，单位 uA
rptMax4	否	0-20000，阈值最大值，单位 uA
rptRule5	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin5	否	0-20000，阈值最小值，单位 uA
rptMax5	否	0-20000，阈值最大值，单位 uA
rptRule6	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin6	否	0-20000，阈值最小值，单位 uA
rptMax6	否	0-20000，阈值最大值，单位 uA
rptRule7	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin7	否	0-20000，阈值最小值，单位 uA

rptMax7	否	0-20000，阈值最大值，单位 uA
rptRule8	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin8	否	0-20000，阈值最小值，单位 uA
rptMax8	否	0-20000，阈值最大值，单位 uA

示例：

```
{
    "msgType": "getAiConfig",
    "data":""
}
{
    "msgType": "getAiConfigAck",
    "data": {
        "enRpt": "0",
        "cyc": "10",
        "rptRule1": "2",
        "rptMin1": "4000",
        "rptMax1": "4000",
        "rptRule2": "0",
        "rptMin2": "4000",
        "rptMax2": "4000",
        "rptRule3": "2",
        "rptMin3": "4000",
        "rptMax3": "4000",
        "rptRule4": "0",
        "rptMin4": "4000",
        "rptMax4": "4000",
        "rptRule5": "0",
        "rptMin5": "4000",
        "rptMax5": "4000",
        "rptRule6": "0",
        "rptMin6": "4000",
        "rptMax6": "4000",
        "rptRule7": "0",
        "rptMin7": "4000",
        "rptMax7": "4000",
        "rptRule8": "2",
        "rptMin8": "4000",
        "rptMax8": "4000"
    }
}
```

注：指令下发后立即生效。

5.3.修改 AI 配置

请求帧格式：

字段	是否必须	描述
msgType	是	setAiConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
enRpt	否	1：开启周期上报　0：关闭周期上报
cyc	否	0-65535：DI 状态上报周期，单位 秒
rptRule1	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin1	否	0-20000，阈值最小值，单位 uA
rptMax1	否	0-20000，阈值最大值，单位 uA
rptRule2	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin2	否	0-20000，阈值最小值，单位 uA
rptMax2	否	0-20000，阈值最大值，单位 uA
rptRule3	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin3	否	0-20000，阈值最小值，单位 uA
rptMax3	否	0-20000，阈值最大值，单位 uA
rptRule4	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin4	否	0-20000，阈值最小值，单位 uA
rptMax4	否	0-20000，阈值最大值，单位 uA
rptRule5	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin5	否	0-20000，阈值最小值，单位 uA
rptMax5	否	0-20000，阈值最大值，单位 uA
rptRule6	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin6	否	0-20000，阈值最小值，单位 uA

rptMax6	否	0-20000，阈值最大值，单位 uA
rptRule7	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin7	否	0-20000，阈值最小值，单位 uA
rptMax7	否	0-20000，阈值最大值，单位 uA
rptRule8	否	0：关闭阈值上报 1：阈值内 2：阈值外
rptMin8	否	0-20000，阈值最小值，单位 uA
rptMax8	否	0-20000，阈值最大值，单位 uA

响应帧格式：

字段	是否必须	描述
msgType	是	setAiConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setAiConfig",
    "data": {
        "enRpt": "0",
        "cyc": "10",
        "rptRule1": "2",
        "rptMin1": "4000",
        "rptMax1": "4000",
        "rptRule2": "0",
        "rptMin2": "4000",
        "rptMax2": "4000",
        "rptRule3": "2",
        "rptMin3": "4000",
        "rptMax3": "4000",
        "rptRule4": "0",
        "rptMin4": "4000",
        "rptMax4": "4000",
        "rptRule5": "0",
        "rptMin5": "4000",
        "rptMax5": "4000",
        "rptRule6": "0",
        "rptMin6": "4000",
        "rptMax6": "4000",
```

```
            "rptRule7": "0",
            "rptMin7": "4000",
            "rptMax7": "4000",
            "rptRule8": "2",
            "rptMin8": "4000",
            "rptMax8": "4000"
        }
    }
    {
        "msgType": "setAiConfigAck",
        "data":"0"
    }

    {
        "msgType":"setAiConfig",
        "data": {
            "enRpt": "0",
            "cyc": "10",
            "rptRule1": "2",
            "rptMin1": "4000",
            "rptMax1": "4000",
            "rptRule2": "0",
            "rptMin2": "4000",
            "rptMax2": "4000",
            "rptRule3": "2",
            "rptMin3": "4000",
            "rptMax3": "4000",
            "rptRule4": "0",
            "rptMin4": "4000",
            "rptMax4": "4000",
            "rptRule5": "0",
            "rptMin5": "4000",
            "rptMax5": "4000",
            "rptRule6": "0",
            "rptMin6": "4000",
            "rptMax6": "4000",
            "rptRule7": "0",
            "rptMin7": "4000",
            "rptMax7": "4000",
            "rptRule8": "6",
            "rptMin8": "4000",
            "rptMax8": "4000"
        }
    }
```

```
{
    "msgType": "setAiConfigAck",
    "data":"1"
}
```

注：指令下发后立即生效。

5.4.AI 主动上报

主动上报帧格式：

字段	是否必须	描述
msgType	是	aiValueRpt
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
AI1	接口数量决定	0-20000 AI1 输入电流值，单位 uA
AI2	接口数量决定	0-20000 AI2 输入电流值，单位 uA
AI3	接口数量决定	0-20000 AI3 输入电流值，单位 uA
AI4	接口数量决定	0-20000 AI4 输入电流值，单位 uA
AI5	接口数量决定	0-20000 AI5 输入电流值，单位 uA
AI6	接口数量决定	0-20000 AI6 输入电流值，单位 uA
AI7	接口数量决定	0-20000 AI7 输入电流值，单位 uA
AI8	接口数量决定	0-20000 AI8 输入电流值，单位 uA

示例：

```
{
    "msgType": "aiValueRpt",
    "data": {
        "AI1": "4000",
        "AI2": "5000",
        "AI3": "6000",
        "AI4": "7000",
        "AI5": "8000",
        "AI6": "9000",
        "AI7": "10000",
```

```
            "AI8": "20000"
        }
    }
```

注：主动上报指令服务器无需回复。

6.AO 接口

AO 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getAoValue	服务器->设备	读取 AO 输出值
getAoValueAck	设备->服务器	读取 AO 输出值的回复
setAoConfig	服务器->设备	设置 AO 输出值
setAoConfigAck	设备->服务器	设置 AO 输出值的回复

6.1.获取 AO 输出

请求帧格式：

字段	是否必须	描述
msgType	是	getAoValue
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getAoValueAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
AO1	硬件接口数量决定	4000-20000， AO1 输出值，单位 uA
AO2	硬件接口数量决定	4000-20000， AO2 输出值，单位 uA

示例：

```
{
    "msgType":"getAoValue",
    "data":""
}
{
    "msgType": "getAoValueAck",
    "data": {
        "AO1": "20000",
        "AO2": "20000"
    }
}
```

注：指令下发后立即生效。

6.2.设置 AO 输出

请求帧格式：

字段	是否必须	描述
msgType	是	setAoConfig
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
AO1	否	4000-20000， AO1 输出值，单位 uA
AO2	否	4000-20000， AO1 输出值，单位 uA

响应帧格式：

字段	是否必须	描述
msgType	是	setAoConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setAoValue",
    "data": {
        "AO1": "4000-20000",
```

```
            "AO2": "4000-20000"
        }
    }
    {
        "msgType": "setAoValueAck",
        "data":"0"
    }

    {
        "msgType":"setAoValue",
        "data": {
            "AO1": "11000",
            "AO2": "1000"
        }
    }
    {
        "msgType": "setAoValueAck",
        "data":"1"
    }
```


注：指令下发后立即生效。

7.RS485

RS485 在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
setRs485Value	服务器->设备	服务器通过 485 发送数据
setRs485ValueAck	设备->服务器	设备 485 接收数据主动上报
rs485ValueRpt	设备->服务器	设备 485 接收数据主动上报
getRs485Config	服务器->设备	请求 RS485 配置信息
getRs485ConfigAck	设备->服务器	请求 RS485 配置信息的回复
setRs485Config	服务器->设备	修改 RS485 配置信息
setRs485ConfigAck	设备->服务器	修改 RS485 配置信息的回复

7.1.向 RS485 发送数据

请求帧格式：

字段	是否必须	描述
msgType	是	setRs485Value
data	是	16 进制的字符串，最大不得超过 1024 字节

响应帧格式：

字段	是否必须	描述
msgType	是	setRs485ValueAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setRs485Value",
    "data":"6801020304050668110433333333331416"
}
```

{
"msgType": "setRs485ValueAck",
"data":"0"
}

7.2.RS485 主动上报

主动上报帧格式：

字段	是否必须	描述
msgType	是	rs485ValueRpt
data	是	16 进制的字符串，最大不得超过 1024 字节

示例：

{
"msgType":"rs485ValueRpt",
"data":"6801020304050668110433333333 1416"
}

7.3.获取 RS485 配置

请求帧格式：

字段	是否必须	描述
msgType	是	getRs485Config
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getRs485ConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
br	否	115200　波特率

wl	否	8/9 数据位
sb	否	1/1.5/2 停止位
parity	否	NONE/ODD/EVEN 奇偶校验位
pkg1	否	16 进制的字符串，最大不得超过 32 字节
len1	否	指令长度
cyc1	否	指令周期
pkg2	否	16 进制的字符串，最大不得超过 32 字节
len2	否	指令长度
cyc2	否	指令周期
pkg3	否	16 进制的字符串，最大不得超过 32 字节
len3	否	指令长度
cyc3	否	指令周期
pkg4	否	16 进制的字符串，最大不得超过 32 字节
len4	否	指令长度
cyc4	否	指令周期

示例：

```
{
    "msgType": "getRs485Config",
    "data":""
}
{
    "msgType": "getRs485ConfigAck",
    "data": {
        "br": "115200",
        "wl": "8",
        "sb": "1",
        "parity": "NONE",
        "pkg1": "1234567890",
        "len1": "5",
        "cyc1": "30",
        "pkg2": "1234567890",
        "len2": "5",
        "cyc2": "30",
        "pkg3": "1234567890",
        "len3": "5",
```

```
            "cyc3": "30",
            "pkg4": "1234567890",
            "len4": "5",
            "cyc4": "30"
        }
}
```

注：指令下发后立即生效。

7.4.修改 RS485 配置

请求帧格式：

字段	是否必须	描述
msgType	是	setRs485Config
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
br	否	115200　波特率
wl	否	8/9 数据位
sb	否	1/1.5/2 停止位
parity	否	NONE/ODD/EVEN 奇偶校验位
pkg1	否	16 进制的字符串，最大不得超过 32 字节
len1	否	指令长度
cyc1	否	指令周期
pkg2	否	16 进制的字符串，最大不得超过 32 字节
len2	否	指令长度
cyc2	否	指令周期
pkg3	否	16 进制的字符串，最大不得超过 32 字节
len3	否	指令长度
cyc3	否	指令周期
pkg4	否	16 进制的字符串，最大不得超过 32 字节

len4	否	指令长度
cyc4	否	指令周期

响应帧格式：

字段	是否必须	描述
msgType	是	setRs485ConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：


```
{
    "msgType":"setRs485Config",
    "data": {
        "br": "115200",
        "wl": "8",
        "sb": "1",
        "parity": "NONE",
        "pkg1": "1234567890",
        "len1": "5",
        "cyc1": "30",
        "pkg2": "1234567890",
        "len2": "5",
        "cyc2": "30",
        "pkg3": "1234567890",
        "len3": "5",
        "cyc3": "30",
        "pkg4": "1234567890",
        "len4": "5",
        "cyc4": "30"
    }
}
{
    "msgType": "setRs485ConfigAck",
    "data":"0"
}

{
    "msgType":"setRs485Config",
    "data": {
        "br": "115200",
        "wl": "8",
```

```
            "sb": "1",
            "parity": "NONE",
            "pkg1": "1234567890",
            "len1": "10",
            "cyc1": "30",
            "pkg2": "1234567890",
            "len2": "10",
            "cyc2": "30",
            "pkg3": "1234567890",
            "len3": "10"
            "cyc3": "30",
            "pkg4": "1234567890",
            "len4": "10"
            "cyc4": "30"
        }
}
{
    "msgType": "setRs485ConfigAck",
    "data":"1"
}
```

注：指令下发后立即生效。

8.逻辑

逻辑包含本地逻辑和设备间逻辑，在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getLogicConfig	服务器->设备	服务器读取逻辑配置信息
getLogicConfigAck	设备->服务器	服务器读取逻辑配置信息的回复
setLogicConfig	设备->服务器	服务器修改逻辑配置信息
setLogicConfigAck	服务器->设备	服务器修改逻辑配置信息的回复

8.1.读取逻辑配置

请求帧格式：

字段	是否必须	描述
msgType	是	getLogicConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getLogicConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
rule1	否	参数 1　0 关闭逻辑　1 正向跟随　2 反向跟随 3 模拟跟随　4 大于等于　5 小于等于 参数 2　0-255,设备间逻辑的 modbus 地址 参数 3　0-65535 输入寄存器地址,10 进制表示
rule2	否	
rule3	否	

rule4	否	参数 4　1 继电器输出　　2 模拟量输出 参数 5　0-65535 输出寄存器地址,10 进制表示 参数 6　1 断开 2 闭合 3 翻转 继电器输出方式 参数 7　比较阈值 参数 8　输出阈值
rule5	否	
rule6	否	
rule7	否	
rule8	否	

示例：

```
{
    "msgType":"getLogicConfig",
    "data":""
}
{
    "msgType": "getLogicConfigAck",
    "data": {
        "rule1": "0,1,0001,1,0001,1,4001,4002",
        "rule2": "0,1,0001,1,0001,1,4001,4002",
        "rule3": "0,1,0001,1,0001,1,4001,4002",
        "rule4": "0,1,0001,1,0001,1,4001,4002",
        "rule5": "0,1,0001,1,0001,1,4001,4002",
        "rule6": "0,1,0001,1,0001,1,4001,4002",
        "rule7": "0,1,0001,1,0001,1,4001,4002",
        "rule8": "0,1,0001,1,0001,1,4001,4002"
    }
}
```

注：指令下发后立即生效。

8.2.修改逻辑配置

请求帧格式：

字段	是否必须	描述
msgType	是	setLogicConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
rule1	否	参数 1　0 关闭逻辑　1 正向跟随　2 反向跟随 3 模拟跟随　4 大于等于　5 小于等于 参数 2　0-254,设备间逻辑的 modbus 地址 参数 3　0-65535 输入寄存器地址,10 进制表示 参数 4　1 继电器输出　　2 模拟量输出 参数 5　0-65535 输出寄存器地址,10 进制表示 参数 6　1 断开 2 闭合 3 翻转 继电器输出方式 参数 7　比较阈值 参数 8　输出阈值
rule2	否	
rule3	否	
rule4	否	
rule5	否	
rule6	否	
rule7	否	
rule8	否	

响应帧格式：

字段	是否必须	描述
msgType	是	setLogicConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setLogicConfig",
    "data": {
        "rule1": "0,1,0001,1,0001,1,4001,4002",
        "rule2": "0,1,0001,1,0001,1,4001,4002",
        "rule3": "0,1,0001,1,0001,1,4001,4002",
        "rule4": "0,1,0001,1,0001,1,4001,4002",
        "rule5": "0,1,0001,1,0001,1,4001,4002",
        "rule6": "0,1,0001,1,0001,1,4001,4002",
        "rule7": "0,1,0001,1,0001,1,4001,4002",
        "rule8": "0,1,0001,1,0001,1,4001,4002"
    }
}
{
    "msgType": "setLogicConfigAck",
    "data":"0"
}
```

注：指令下发后立即生效。

9.系统参数

系统参数配置在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getDeviceConfig	服务器->设备	服务器读取系统参数配置信息
getDeviceConfigAck	设备->服务器	服务器读取系统参数配置信息的回复
setDeviceConfig	设备->服务器	服务器修改系统参数配置信息
setDeviceConfigAck	服务器->设备	服务器修改系统参数配置信息的回复

9.1.获取系统参数

请求帧格式：

字段	是否必须	描述
msgType	是	getDeviceConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getDeviceConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
devAddr	否	1-254 modbus 通信地址码
devID	否	纵横产品唯一标识符，长度固定为 16 字节
devPW	否	最大 20 字节，产品密码，连接纵横云平台使用
vertion	否	0-65535，固件版本号

rptDrec	否	数据上报方式，0-65535
groupMode	否	0 关闭 1 打开
groupType	否	1（A） 2（B）
groupSN	否	最大 20 字节，组名称
groupPW	否	最大 20 字节，组密码
clock1	否	参数 1 1 开启闹钟 0 关闭闹钟 参数 2 时间-时 0-23 参数 3 时间-分 0-59 参数 4 时间-秒 0-59 参数 5 动作分类 1 控制 DO 2 重启设备 参数 6 动作主体 1(DO1)...8(DO8) 参数 7 动作内容 1 断开 2 闭合
clock2	否	
clock3	否	
clock4	否	
clock5	否	
clock6	否	
clock7	否	
clock8	否	

示例：

```
{
    "msgType":"getDeviceConfig",
    "data":""
}
{
    "msgType": "getDeviceConfigAck",
    "data": {
        "devAddr": "1",
        "devID": "4921200723009898",
        "devPW": "123456",
        "vertion": "1024",
        "rptDrec": "256",
        "groupMode": "0",
        "groupType": "1",
        "groupSN": "1AEQ1231313132",
        "groupPW": "123456",
        "clock1": "1,7,30,00,1,1,1",
        "clock2": "1,8,30,00,1,1,1",
        "clock3": "1,9,30,00,1,2,1",
        "clock4": "1,10,30,00,1,2,1",
        "clock5": "1,11,30,00,1,4,1",
        "clock6": "1,12,30,00,1,1,1",
```

```
            "clock7": "1,13,30,00,1,2,1",
            "clock8": "1,14,30,00,1,2,1"
        }
    }
```

注：指令下发后立即生效。

9.2.修改系统参数

请求帧格式：

字段	是否必须	描述
msgType	是	setDeviceConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
devAddr	否	1-254 modbus 通信地址码
devID	否	纵横产品唯一标识符，长度固定为 16 字节
devPW	否	最大 20 字节，产品密码，连接纵横云平台使用
vertion	否	0-65535，固件版本号
rptDrec	否	数据上报方式，0-65535
groupMode	否	0 关闭 1 打开
groupType	否	1（A） 2（B）
groupSN	否	最大 20 字节，组名称
groupPW	否	最大 20 字节，组密码
clock1	否	参数 1 1 开启闹钟 0 关闭闹钟
clock2	否	参数 2 时间-时 0-23
clock3	否	参数 3 时间-分 0-59
clock4	否	参数 4 时间-秒 0-59
clock5	否	参数 5 动作分类 1 控制 DO 2 重启设备
clock6	否	参数 6 动作主体 1(DO1)...8(DO8)

clock7	否	参数 7　动作内容　1 断开 2 闭合
clock8	否	

响应帧格式：

字段	是否必须	描述
msgType	是	setDeviceConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：


```
{
    "msgType":"setDeviceConfig",
    "data": {
        "devAddr": "1",
        "devPW": "123456",
        "vertion": "1024",
        "rptDrec": "256",
        "groupMode": "0",
        "groupType": "1",
        "groupSN": "1AEQ1231313132",
        "groupPW": "123456",
        "clock1": "1,7,30,00,1,1,1",
        "clock2": "1,8,30,00,1,1,1",
        "clock3": "1,9,30,00,1,2,1",
        "clock4": "1,10,30,00,1,2,1",
        "clock5": "1,11,30,00,1,4,1",
        "clock6": "1,12,30,00,1,1,1",
        "clock7": "1,13,30,00,1,2,1",
        "clock8": "1,14,30,00,1,2,1"
        }
}
{
    "msgType": "setDeviceConfigAck",
    "data":"0"
}
```


注：指令下发后立即生效。

10.网络基础参数

网络基础参数在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getLteNormalConfig	服务器->设备	服务器读取 LTE 基础参数信息
getLteNormalConfigAck	设备->服务器	服务器读取 LTE 基础参数的回复
setLteNormalConfig	设备->服务器	服务器修改 LTE 基础参数信息
setLteNormalConfigAck	服务器->设备	服务器修改 LTE 基础参数的回复

10.1.读取网络基础参数

请求帧格式：

字段	是否必须	描述
msgType	是	getLteNormalConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getLteNormalConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
ccid	否	最大 20 字节，卡号
qcsq	否	最大 40 字节，信号值
apnAddr	否	最大 44 字节，APN，默认为空
apnName	否	最大 44 字节，APN-NAME，默认为空
apnPass	否	最大 44 字节，APN-PWD，默认为空

superCmd	否	0 关闭 1 打开
gpsCyc	否	0-65535 GPS 上报周期，单位 S，0 为关闭上报
gps	否	最大 100 字节，GPS 位置信息

示例：

```
{
{
    "msgType":"getLteNormalConfig",
    "data":""
}
{
    "msgType": "getLteNormalConfigAck",
    "data": {
        "ccid": "860523354421544",
        "qcsq": "LTE,68,58,148",
        "apnAddr": "NET",
        "apnName": "",
        "apnPass": "",
        "superCmd": "0",
        "gpsCyc": "0",
        "gps": "1-1-1915"
    }
}
```

注：指令下发后立即生效。

10.2.修改网络基础参数

请求帧格式：

字段	是否必须	描述
msgType	是	setLteNormalConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
ccid	否	最大 20 字节，卡号
qcsq	否	最大 40 字节，信号值

apnAddr	否	最大 44 字节，APN，默认为空
apnName	否	最大 44 字节，APN-NAME，默认为空
apnPass	否	最大 44 字节，APN-PWD，默认为空
superCmd	否	0 关闭 1 打开
gpsCyc	否	0-65535 GPS 上报周期，单位 S，0 位关闭上报
gps	否	最大 100 字节，GPS 位置信息

响应帧格式：

字段	是否必须	描述
msgType	是	setLteNormalConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setLteNormalConfig",
    "data": {
        "apnAddr": "NET",
        "apnName": "",
        "apnPass": "",
        "superCmd": "1",
        "gpsCyc": "1",
        "gps": "1-1-1915"
    }
}
{
    "msgType": "setLteNormalConfigAck",
    "data":"0"
}
```

注：指令下发后立即生效。

11.网络 SOCKET 参数

网络 SOCKET 参数在 JSON 协议交互中，根据功能点不同，分为如下几个：

msgType	数据走向	描述

getLteSocket1Config	服务器->设备	服务器读取 SOCKET1 配置
getLteSocket1ConfigAck	设备->服务器	服务器读取 SOCKET1 配置的回复
setLteSocket1Config	设备->服务器	服务器修改 SOCKET1 配置
setLteSocket1ConfigAck	服务器->设备	服务器修改 SOCKET1 配置的回复
getLteSocket2Config	服务器->设备	服务器读取 SOCKET2 配置
getLteSocket2ConfigAck	设备->服务器	服务器读取 SOCKET2 配置的回复
setLteSocket2Config	设备->服务器	服务器修改 SOCKET2 配置
setLteSocket2ConfigAck	服务器->设备	服务器修改 SOCKET2 配置的回复
getLteSocket3Config	服务器->设备	服务器读取 SOCKET3 配置
getLteSocket3ConfigAck	设备->服务器	服务器读取 SOCKET3 配置的回复
setLteSocket3Config	设备->服务器	服务器修改 SOCKET3 配置
setLteSocket3ConfigAck	服务器->设备	服务器修改 SOCKET3 配置的回复
getLteSocket4Config	服务器->设备	服务器读取 SOCKET4 配置
getLteSocket4ConfigAck	设备->服务器	服务器读取 SOCKET4 配置的回复
setLteSocket4Config	设备->服务器	服务器修改 SOCKET4 配置
setLteSocket4ConfigAck	服务器->设备	服务器修改 SOCKET4 配置的回复

11.1.读取 SOCKET 配置

请求帧格式：

字段	是否必须	描述
msgType	是	getLteSocket1Config
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getLteSocket1ConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
enable	否	0 禁用 1 启用
mode	否	1 TCPC 4 MQTT
desIp	否	最大 66 字节，字符串
desPort	否	0-65535
regMode	否	0 关闭注册包 1 云转发 2 自定义 3ID 4CCID
regPos	否	1 连接发送 2 数据携带 3 以上两种
regPkg	否	16 进制字符串，最大不超过 200 字节
hbtMode	否	1 启用 2 关闭
hbtCyc	否	0-65535 心跳周期，单位 S
hbtPkg	否	16 进制字符串，最大不超过 40 字节

示例：

```
{
    "msgType":"getLteSocket1Config",
    "data":""
}
{
    "msgType": "getLteSocket1ConfigAck",
    "data": {
        "enable": "1",
        "mode": "1",
        "desIp": "www.iotrouter.com",
        "desPort": "55000",
        "regMode": "1",
        "regPos": "1",
        "regPkg": "492149214921",
        "hbtMode": "1",
        "hbtCyc": "10",
        "hbtPkg": "492149214921"
    }
}
```

注：指令下发后立即生效。

11.2.修改 SOCKET 配置

请求帧格式：

字段	是否必须	描述
msgType	是	setLteSocket1Config
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
enable	否	0 禁用 1 启用
mode	否	1 TCP 4 MQTT
desIp	否	最大 66 字节，字符串
desPort	否	0-65535
regMode	否	0 关闭注册包 1 云转发 2 自定义 3ID 4CCID
regPos	否	1 连接发送 2 数据携带 3 以上两种
regPkg	否	16 进制字符串，最大不超过 200 字节
hbtMode	否	1 启用 2 关闭
hbtCyc	否	0-65535 心跳周期，单位 S
hbtPkg	否	16 进制字符串，最大不超过 40 字节

响应帧格式：

字段	是否必须	描述
msgType	是	setLteSocket1ConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setLteSocket1Config",
    "data": {
        "enable": "1",
```

```
            "mode": "1",
            "desIp": "www.iotrouter.com",
            "desPort": "55000",
            "regMode": "1",
            "regPos": "2",
            "regPkg": "492149214921",
            "hbtMode": "1",
            "hbtCyc": "10",
            "hbtPkg": "492149214921"
        }
}
{
    "msgType": "setLteSocket1ConfigAck",
    "data":"0"
}
```


注：指令下发后立即生效。

12.MQTT

MQTT 参数在 JSON 协议交互中，根据功能点不同，分为如下几个 msgType：

msgType	数据走向	描述
getLteMqttConfig	服务器->设备	服务器读取 MQTT 通信参数
getLteMqttConfigAck	设备->服务器	服务器读取 MQTT 通信参数的回复
setLteMqttConfig	设备->服务器	服务器修改 MQTT 通信参数
setLteMqttConfigAck	服务器->设备	服务器修改 MQTT 通信参数的回复

12.1.获取 MQTT 参数

请求帧格式：

字段	是否必须	描述
msgType	是	getLteMqttConfig
data	是	空

响应帧格式：

字段	是否必须	描述
msgType	是	getLteMqttConfigAck
data	是	data 帧格式

data 帧格式:

字段	是否必须	描述
keepAlive	否	30-1200 单位 S
clear	否	目前只支持 1
clientID	否	最大 60 字节，字符串
userName	否	最大 60 字节，字符串
passWord	否	最大 60 字节，字符串

subTopic	否	最大 100 字节，字符串
pubTopic	否	最大 100 字节，字符串

示例：

```
{
    "msgType":"getLteMqttConfig",
    "data":""
}
{
    "msgType": "getLteMqttConfigAck",
    "data": {
        "keepAlive": "30",
        "clear": "1",
        "clientID": "sadfgiaiodjfkasdf",
        "userName": "4921190608225632",
        "passWord": "zhe19141915",
        "subTopic": "/sub/topic/121123wdqwe",
        "pubTopic": "/pub/topic/121123wsad"
    }
}
```

注：指令下发后立即生效。

12.2.修改 MQTT 参数

请求帧格式：

字段	是否必须	描述
msgType	是	setLteMqttConfig
data	是	data 帧格式

data 帧格式：

字段	是否必须	描述
keepAlive	否	30-1200 单位 S
clear	否	目前只支持 1
clientID	否	最大 60 字节，字符串
userName	否	最大 60 字节，字符串

passWord	否	最大 60 字节，字符串
subTopic	否	最大 100 字节，字符串
pubTopic	否	最大 100 字节，字符串

响应帧格式：

字段	是否必须	描述
msgType	是	setLteMqttConfigAck
data	是	0：成功 1：参数错误 2：字段错误

示例：

```
{
    "msgType":"setLteMqttConfig",
    "data": {
        "keepAlive": "30",
        "clear": "1",
        "clientID": "sadfgiaiodjfkasdf",
        "userName": "4921190608225632",
        "passWord": "zhe19141915",
        "subTopic": "/sub/topic/121123wdqwe",
        "pubTopic": "/pub/topic/121123wsad"
    }
}
{
    "msgType": "setLteMqttConfigAck",
    "data":"0"
}
```

注：指令下发后立即生效。