

IEC104 转 Modbus 网关

使用说明书

型号：SG-IEC104-TCP

版本：V1.0.0



天津滨海新区三格电子科技有限公司

版本信息

日期	版本号	修改内容	备注
2025/06/11	V1.0.0	建立	

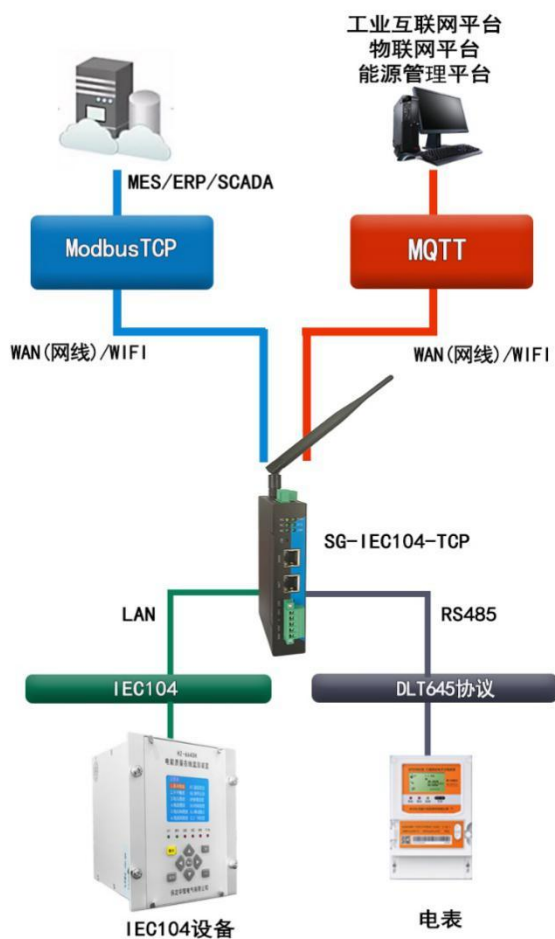
目录

版本信息	2
目录	3
第一章 产品概述	5
第二章 硬件说明	6
2.1. 产品外观	6
2.2. 产品接口	6
2.2.1. 天线接口	6
2.2.2. WAN 口	7
2.2.3. LAN 口	7
2.2.4. 电源端子	7
2.2.5. Reload 按键	7
2.2.6. COM1	7
2.3. 指示灯	8
2.4. 基本参数	8
2.5. 尺寸图	9
2.6. 标准配件	10
第三章 配置软件使用说明	10
3.1. 设备管理	10
3.2. 网络配置	12
3.3. 时间配置	14
3.4. IEC104 采集配置	15
3.4.1. 添加 IEC104	15

3.4.2.	参数配置	15
3.4.3.	数据点配置	16
3.4.4.	证书生成	18
3.5.	DLT645（电表）采集配置说明	19
3.5.1.	添加 DLT645 电表协议	19
3.5.2.	配置参数	20
3.5.3.	数据项配置	21
3.6.	Modbus TCP 服务配置	22
3.6.1.	添加 Modbus TCP 服务	22
3.6.2.	配置数据点寄存器地址	23
3.7.	MQTT 配置	25
3.7.1.	添加 MQTT	25
3.7.2.	配置参数	26
3.7.3.	脚本测试	27
3.7.4.	标签配置	31
3.7.5.	默认 MQTT 报文格式说明	31
3.7.6.	自定义报文格式	33
第四章	数据采集实例	40
4.1.	IEC104 数据采集实例	40
4.2.	DLT645（电表）数据采集实例	46
第五章	售后及联系方式	50

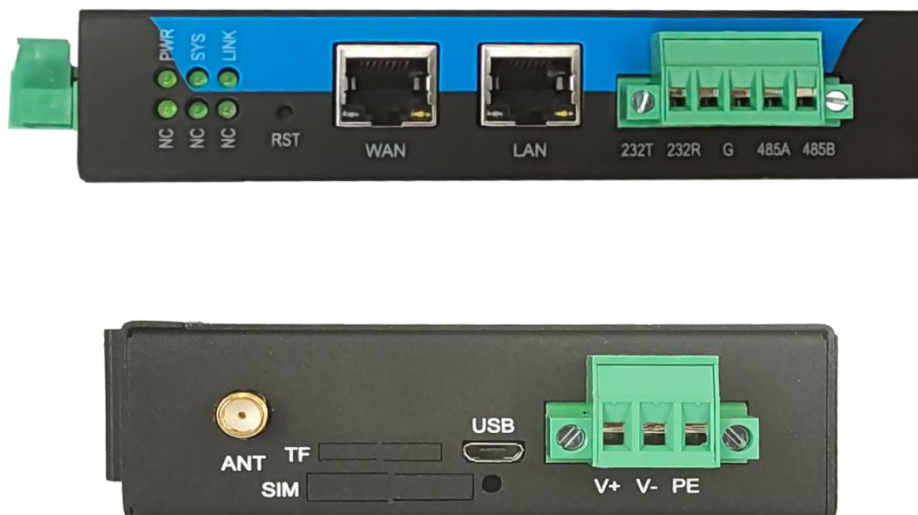
第一章 产品概述

IEC104 转 Modbus 网关型号 SG-IEC104-TCP，是三格电子推出的工业级网关（以下简称网关），主要用于 IEC104 数据采集、DLT645-1997/2007 数据采集，可接 IEC104 设备、电表等，将采集到的数据转换为 Modbus TCP 协议数据或 MQTT 连接云平台。IEC104 实现客户端功能，可接入 IEC104 网络，对接 IEC104 设备。Modbus TCP 实现从站功能，可接 Modbus TCP 主站支持多连接。



第二章 硬件说明

2.1. 产品外观



2.2. 产品接口

- WAN 口：用于连接 ModbusTCP 主站设备。
- LAN 口：用来接 IEC104 从站设备。
- 电源端子接口：网关供电。
- Reload 按键：长按 3~10s 设备恢复出厂设置。
- 天线接口：WIFI 天线。
- COM1：端子接法通用串口 RS485/RS232。

2.2.1. 天线接口

设备提供一个 WIFI 天线接口，接 2.4G WIFI 天线。

2.2.2. WAN 口

设备提供 1 个 WAN 口，可用于连接 ModbusTCP 主站设备。

2.2.3. LAN 口

设备提供 1 个 LAN 口，连接 IEC104 从站。

2.2.4. 电源端子

V+、V-为设备供电口，支持 12~36V DC 供电，PE 可接大地。

2.2.5. Reload 按键

长按 3~10s 设备恢复出厂设置，设备恢复出厂设置 LAN 口默认 IP 为 192.168.1.37。

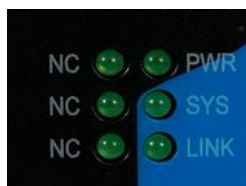
2.2.6. COM1

插拔式接线端子，支持一路 RS485/RS232，不能同时使用。

引脚序号	功能
1	RS232_TX
2	RS232_RX
3	GND
4	RS485_A
5	RS485_B

2.3. 指示灯

- Power 灯：常量表示供电正常。
- Sys 灯：设备工作正常时闪烁。
- Link 灯：MQTT 全部连接到服务器时常亮，部分连接闪烁，全部没有连接时关闭。
- NC：预留。
- NC：预留。
- NC：预留。

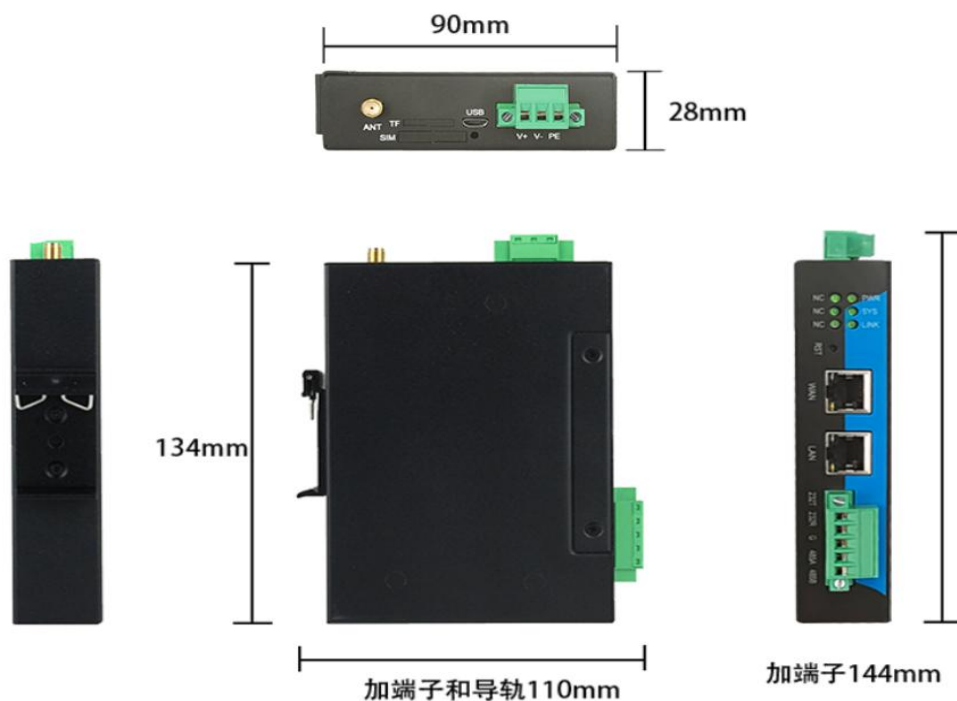


2.4. 基本参数

分类	参数	数值
硬件参数	供电电压	DC 12~36V
	功率	5W
	网口	RJ45、10/100M
	WIFI	2.4G
	串口波特率	9600~115200bps

	IP 获取方式	LAN 口静态 IP，WAN 口默认 DHCP。 LAN/WAN/WIFI 需配置不同网段。
其他	尺寸（mm）	110*144*27（不带端子 90*133*27）
	工作温度	-20~65℃
	存储温度	-40~105℃
	工作湿度	5%~95% RH（无凝露）
	存储湿度	5%~95% RH（无凝露）

2.5. 尺寸图

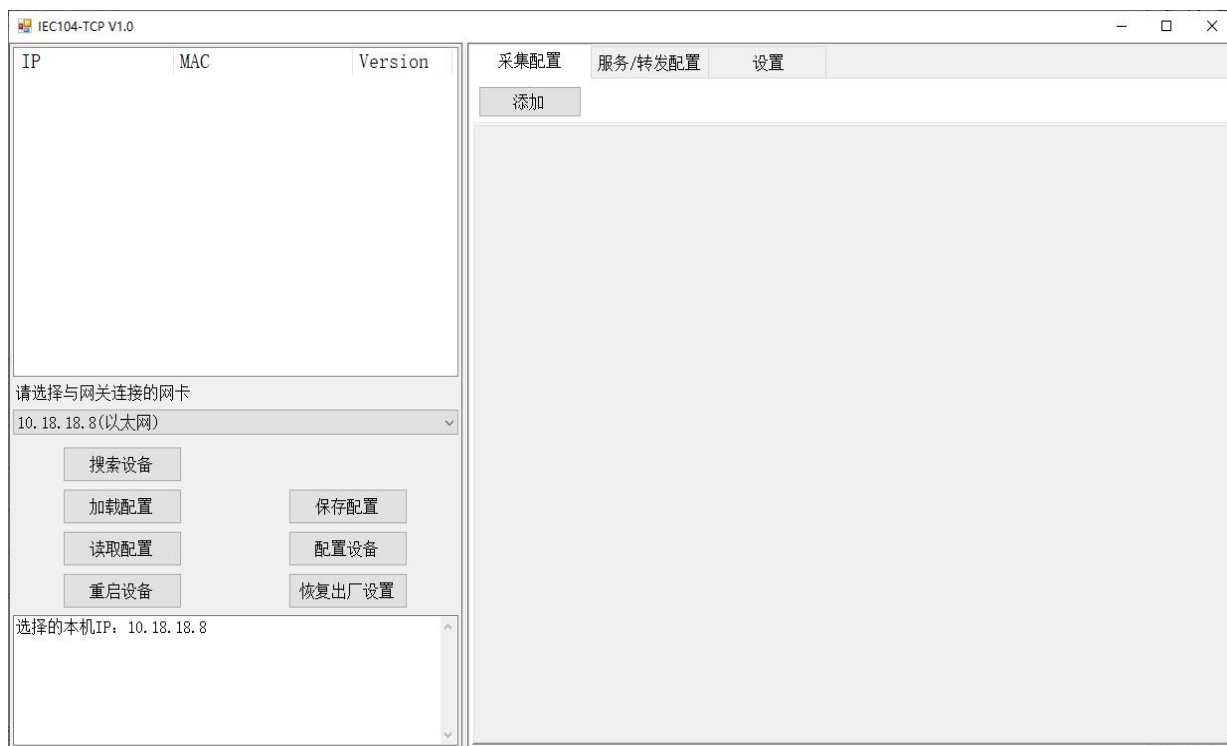


2.6. 标准配件

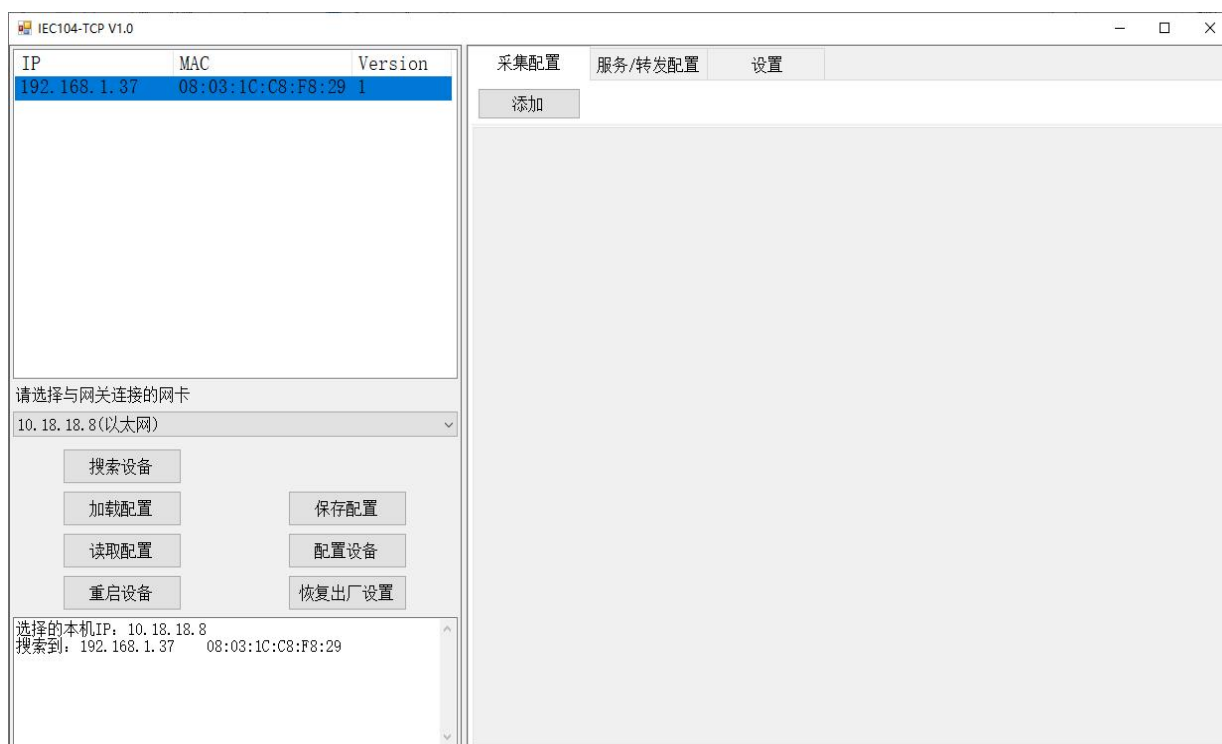
- SG-IEC104-TCP x1
- WIFI 天线 x1（选配）

第三章 配置软件使用说明

3.1. 设备管理



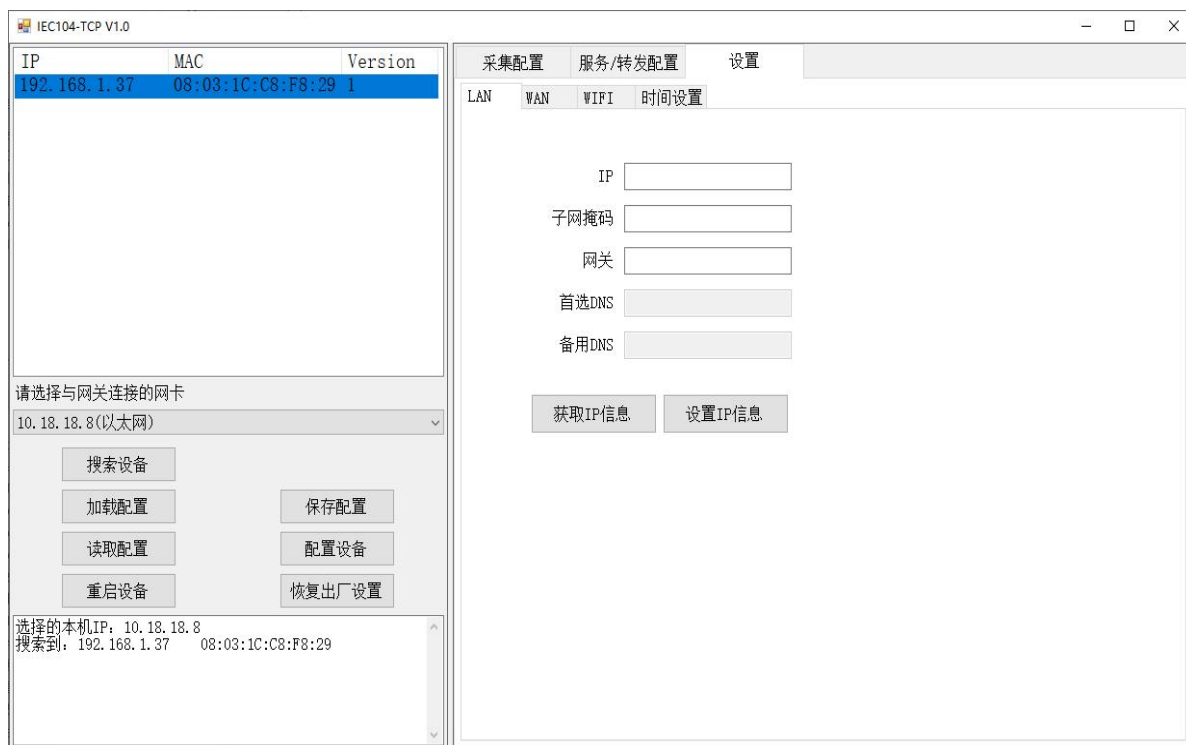
选择电脑端与网关连接的网卡。点击搜索设备可查看搜索到的网关 IP。



选中要配置的网关，点击读取配置可将网关中配置的参数信息显示到右侧参数信息中。点击配置设备可将当前修改的配置同步到网关。可操作重启网关和恢复出厂设置。

保存配置：将当前配置保存到配置文件。加载配置：从配置文件中加载配置信息。

3.2. 网络配置



左侧选中设备，右侧切换到设置页，可配置 LAN、WAN、WIFI 参数。

➤ LAN



LAN 口可配置 IP 子网掩码。点击获取 IP 信息可获得 LAN 口的 IP 信息。点击设置 IP 信

息，将当前配置的 LAN 口 IP 信息更新到网关。

➤ WAN



LAN WAN WIFI 时间设置

☐ 静态IP ☒ 动态IP

IP

子网掩码

网关

首选DNS

备用DNS

获取IP信息 设置IP信息

WAN 口可配置动态 IP 和静态 IP。点击获取 IP 信息可获取 WAN 口配置信息，点击设置 IP 信息将当前 WAN 口配置更新到网关。

➤ WIFI

LAN

WAN

WIFI

时间设置

☐ 开启

☐ 关闭

SSID

密码

☐ 静态IP

☒ 动态IP

IP

子网掩码

网关

首选DNS

备用DNS

获取IP信息

设置IP信息

WIFI 支持 station 模式用于联网。

WIFI 可设置 SSID 和密码，配置动态 IP 和静态 IP。点击获取 IP 信息可获得 WIFI 配置信息，点击设置 IP 信息将当前 WIFI 配置更新到网关。

3.3. 时间配置

LAN

WAN

WIFI

时间设置

时间

获取时间

同步本机时间

NTP服务器

获取

设置

获取时间可获取网关当前时间。同步本机时间是将当前计算机时间同步到网关。

可获取和设置 NTP 服务器。网关断电不保存时间，网关不连接公网时可配置内网 NTP 服务器，以确保网关时间正确，网关会自动从配置的 NTP 服务器获取时间。连接公网情况下不需要配置。

3.4. IEC104 采集配置

3.4.1. 添加 IEC104



在采集配置页点击添加



协议处可选择 IEC104、DLT645 协议。设备名称必须设置且不能重复。

3.4.2. 参数配置



配置 IEC104 设备 IP 地址、端口、COA 地址、数据采集间隔（毫秒）等。

参数设置

K	12	w	8
t0	10	t1	15
t2	10	t3	20

TLS加密 ☐

根证书 📁 选则文件

客户端证书 📁 选则文件

客户端密钥 📁 选则文件

客户端密钥密码

验证服务器证书 ☐

服务器证书 📁 选则文件

确定 取消

参数设置中可配置 TLS 加密机证书文件。

3.4.3. 数据点配置

104 ✖

IP	192.168.1.8	超时时间	1000	COA	1
端口	2404	采集间隔	3000	参数设置	

添加 清空 导入 导出

序号	名称/标签	IOA地址	ASDU类型	数据类型	删除
0			M_SP_NA_1 ▾	bool	✖
1			▾		✖

点击添加可添加数据点、清空删除所有数据点，导入可从 csv 文件导入，导出将当前数据点配置导出到 csv 文件。

名称/标签必须填写，且不能重复。IOA 地址为 IEC104 数据点地址。

下表为支持 IEC104 的类型及对应的数据类型。监控方向可采集数据，控制方向只能写入数据无法采集。

监控方向		控制方向	
M_SP_NA_1	bool	C_SC_NA_1	bool
M_SP_TA_1	bool	C_DC_NA_1	int8
M_DP_NA_1	int8	C_RC_NA_1	int8
M_DP_TA_1	int8	C_SE_NA_1	float
M_ST_NA_1	int8	C_SE_NB_1	int16
M_ST_TA_1	int8	C_SE_NC_1	float
M_BO_NA_1	uint32	C_BO_NA_1	uint32
M_BO_TA_1	uint32	C_SC_TA_1	bool
M_ME_NA_1	float	C_DC_TA_1	int8
M_ME_TA_1	float	C_RC_TA_1	int8
M_ME_NB_1	int16	C_SE_TA_1	float
M_ME_TB_1	int16	C_SE_TB_1	int16
M_ME_NC_1	float	C_SE_TC_1	float
M_ME_TC_1	float	C_BO_TA_1	uint32
M_IT_NA_1	int32		
M_IT_TA_1	int32		
M_EP_TA_1	int8		
M_EP_TB_1	int8		
M_EP_TC_1	int8		
M_PS_NA_1	uint32		
M_ME_ND_1	float		
M_SP_TB_1	bool		
M_DP_TB_1	int8		
M_ST_TB_1	int8		
M_BO_TB_1	uint32		
M_ME_TD_1	float		
M_ME_TE_1	int16		
M_ME_TF_1	float		
M_IT_TB_1	int32		
M_EP_TD_1	int8		
M_EP_TE_1	int8		

M_EP_TF_1	int8		
-----------	------	--	--

M_PS_NA_1 成组单点遥信 bit0~15 为 STn 状态位，bit16~31 为 CDn 变位检出标志位。

IEC104 数据通过 MQTT 上报服务器时按照采集时间间隔上报。如通过控制方向写数据会单独将对应数据点数据上报服务器。

3.4.4. 证书生成

支持 TLS 加密，支持 crt、key、pem 格式。

以下为 openssl 生成证书文件示例。

生成根证书

1. 生成根证书的私钥和证书签发请求(CSR):

```
openssl req -newkey rsa:2048 -nodes -keyout root.key -out root.csr
```

2. 自签名根证书:

```
openssl x509 -req -days 3650 -in root.csr -signkey root.key -out root.crt
```

此步骤生成的 root.crt 为要配置的根证书。

生成服务器证书

3. 生成服务器证书的私钥和 CSR:

```
openssl req -newkey rsa:2048 -nodes -keyout server.key -out server.csr
```

此步骤生成的 server.key 为服务器证书秘钥。

4. 使用根证书和私钥为服务器证书签名:

```
openssl x509 -req -days 3650 -in server.csr -CA root.crt -CAkey root.key -CAcreateserial -out server.crt
```

此步骤生成的 server.crt 为服务器证书。如需要验证服务器证书，需要配置此证书。

生成客户端证书

5. 生成客户端证书的私钥和 CSR:

```
openssl req -newkey rsa:2048 -nodes -keyout client.key -out client.csr
```

此步骤生成的 client.key 为客户端证书秘钥。

6. 使用根证书和私钥为客户端证书签名:

```
openssl x509 -req -days 3650 -in client.csr -CA root.crt -CAkey root.key -CAcreateserial -out client.crt
```

此步骤生成的 client.crt 为客户端证书。

3.5. DLT645（电表）采集配置说明

3.5.1. 添加 DLT645 电表协议



The image shows a dialog box titled "协议选择" (Protocol Selection). It contains two input fields: "设备名称" (Device Name) with the value "645" and "协议" (Protocol) with a dropdown menu showing "DLT645". At the bottom, there are two buttons: "确定" (Confirm) and "取消" (Cancel).

选择 DLT645 协议。

3.5.2. 配置参数

波特率	2400	停止位	1	通信超时 时间毫秒	1000
数据位	8	校验位	EVEN	采集时间 间隔毫秒	3000
添加电表					
1 ✖					
协议版本:		DLT645/2007			
通讯地址/电表号:		AAAAAAAAAAAA		☒ 全选	

波特率：按照电表实际波特率选择，一搬默认为 2400

数据位：按照实际选择，一搬默认为 8

停止位按照实际选择，一搬默认为 1

校验位：按照实际选择，一搬默认为 EVEN 偶校验

通信超时时间单位毫秒

采集时间间隔单位毫秒

3.5.3. 数据项配置

添加电表

1
✖

协议版本: DLT645/2007 ▼

通讯地址/电表号: AAAAAAAAAAAA ☒ 全选

序号	选择	名称/标签	数据项名称
0	☒		正向有功费率4电能
1	☒		正向有功费率3电能
2	☒		正向有功费率2电能
3	☒		正向有功费率1电能
4	☒		组合有功费率4电能
5	☒		组合有功费率3电能
6	☒		组合有功费率2电能
7	☒		组合有功费率1电能
8	☒		组合无功2总电能
9	☒		组合无功1总电能

名称/标签必须填写，不同电表之间也不能重复。

根据实际需要选择需要采集的数据项。电表可能不支持部分数据项，不支持的建议取消。

添加电表，可同时添加多个电表，电表通讯波特率、数据位、停止位、校验位必须一致。

通讯地址/电表号: AAAAAAAAAAAA |

按照电表实际通讯地址/电表号填写。若不知道电表的地址，且一条 485 链路上只有一个电表，那么可以填写通配地址 12 个 A(AAAAAAAAAAAAAA)。

协议版本: DLT645/2007 ▼

根据电表支持的协议版本选择 DLT645/2007 或 DLT645/1997。

3.6. Modbus TCP 服务配置

3.6.1. 添加 Modbus TCP 服务



服务/转发配置页，点击添加



选择 ModbusTCP，设置名称。ModbusTCP 协议只能添加一个，使用默认端口 502 不可配置。

3.6.2. 配置数据点寄存器地址

1

保持寄存器 输入寄存器 线圈 离散输入

导入数据点

清空

序号	设备名称	变量/标签名	数据类型	Modbus地址	长度	删除
----	------	--------	------	----------	----	----

支持保持寄存器、输入寄存器、线圈和离散输入 4 种寄存器。其中线圈和离散输入只能导入 bool 类型数据。切换到对应的寄存器页，点击导入数据点。

序号	选择	数据点名称	数据类型	地址信息
0	☐	A	bool	M_SP_NA_1 1
1	☐	B	int8	M_DP_TA_1 2

左侧选中设备后，右侧可选择数据点。把需要导入的数据点全部勾选。

保持寄存器 输入寄存器 线圈 离散输入

导入数据点 清空

序号	设备名称	变量/标签名	数据类型	Modbus地址	长度	删除
0	104	A	bool	0	1	×
1	104	B	int8	1	1	×
2	645	a1	float	2	2	×
3	645	a2	float	4	2	×
4	645	a3	float	6	2	×

设置 Modbus 偏移地址，需注意地址不能有重叠。长度列为对应数据类型映射到 Modbus

占用的寄存器数量。在通过 Modbus 读取数据时最大地址不能超过配置的地址。数据使用 Big-endian 模式。

根据数据采集间隔更新 Modbus 寄存器。

3.7. MQTT 配置

3.7.1. 添加 MQTT



The image shows a dialog box titled "协议选择" (Protocol Selection). It contains two input fields: "名称" (Name) with the value "mqttl" and "协议" (Protocol) with a dropdown menu showing "MQTT". At the bottom, there are two buttons: "确定" (Confirm) and "取消" (Cancel).

设置名称，选择 MQTT 协议。

3.7.2. 配置参数

IP/域名	192.168.1.8
端口	1883
客户端ID	1234567890
用户名	username
密码	password
Qos	1
报文格式	☒ 默认 ☐ 自定义 测试脚本
上报主题	
上报JS	
写值主题	
写值JS	

配置 MQTT 服务器 IP，支持域名。配置端口客户端 ID、用户名密码等。

报文格式支持默认和自定义格式。自定义格式需用户自行编译 JavaScript 转换格式脚本。

上报主题为数据上报的 MQTT 主题。

写值主题为 MQTT 服务器通过网关向设备写数据的主题。网关会订阅此主题接收服务器下发的消息。

设备每采集一次通过 MQTT 上报一次数据。

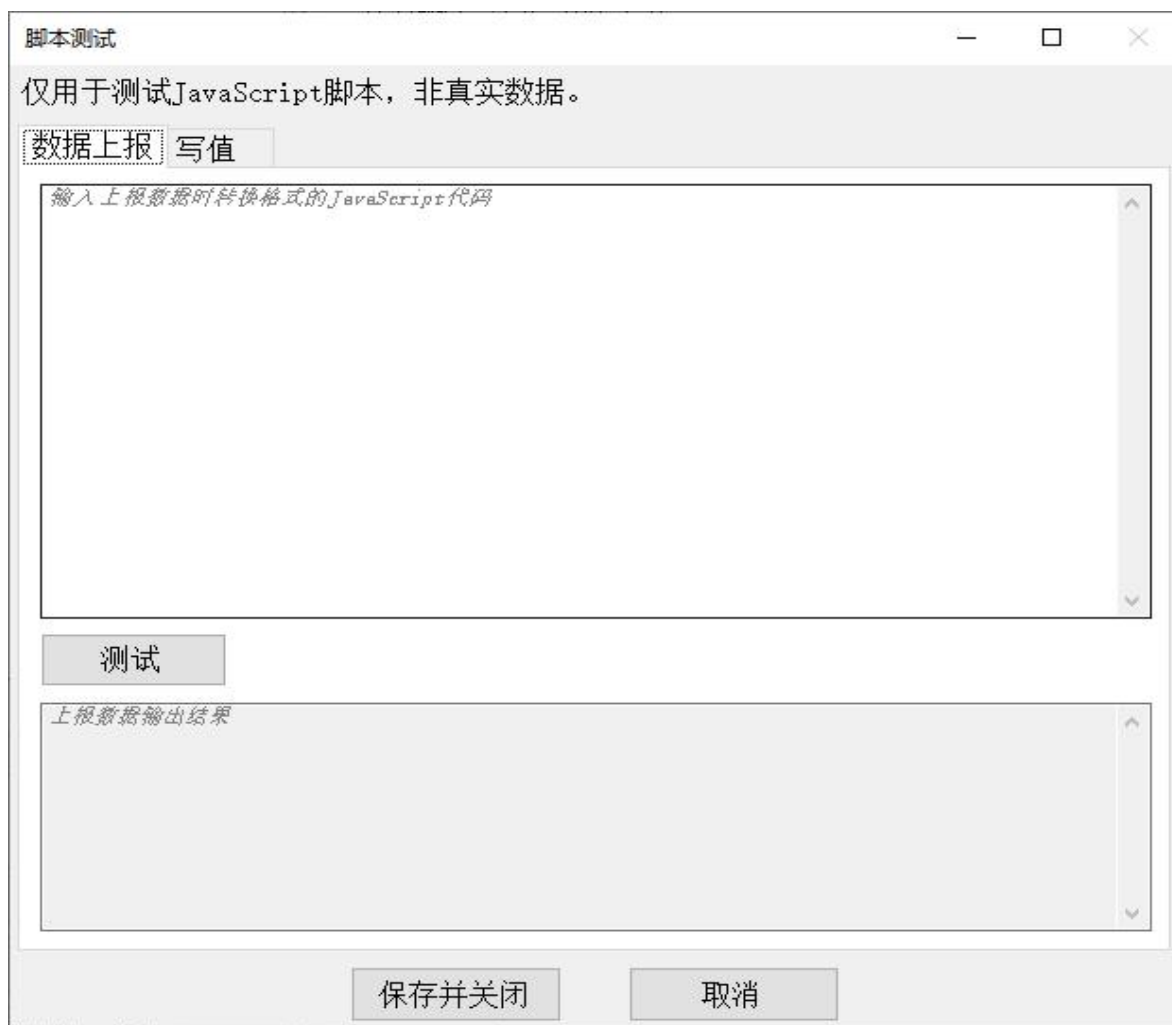
3.7.3. 脚本测试

选择自定义报文时，支持简单的脚步测试。

报文格式 ☐ 默认

☒ 自定义

 **测试脚本**



脚本测试

仅用于测试JavaScript脚本，非真实数据。

数据上报 写值

输入上报数据时转换格式的JavaScript代码

测试

上报数据输出结果

保存并关闭 取消

脚本测试中的数据为模拟数据非真实数据。分为数据上报和写值测试。

脚本测试
 —
□
×

仅用于测试JavaScript脚本，非真实数据。

数据上报
 写值

```

var json = {};
json.tag = payload.deviceName;    // 设备名称对应tag
json.stationid = payload.stationid; // 标签stationid对应stationid
var time = payload.time;

var datas = [];    // 存放转换后的数据点信息
var len = payload.datas.length; // 数据点数量
for (let i = 0; i < len; i++) {
    var data = payload.datas[i];

    if(data.valid==1){        //数据是否有效
        var a = {};
        a.rm = data.name;    // 数据点名称
        a.t = time;          // 时间
        a.v = data.value;    // 值
        a.type=data.dataType; // 数据类型
        datas.push(a);
    }
}
json.datas = datas;
JSON.stringify(json,null,4);    //转Json，序列化。格式化json 使用
    
```

测试

```

{
  "tag": "deviceName123",
  "stationid": "5",
  "datas": [
    {
      "rm": "name1",
      "t": "2025-06-05 10:13:36",
      "v": 1234,
      "type": "int32"
    },
    " " " " "
  ]
}
    
```

保存并关闭

取消

数据上报页输入数据上报转换脚本，点击测试可查看转换后的结果。

脚本测试

— □ ×

仅用于测试JavaScript脚本，非真实数据。

数据上报 写值

请输入写值时的Json

写值时转换脚本

测试

输出结果，需要与默认格式一致。

保存并关闭 取消

写值页需输入 MQTT 服务器下发 Json 格式和转换脚本。点击测试后查看转换结果，需确认转换结果与默认 MQTT 报文格式中写值格式是否一致，转换后格式必须保持一致。

脚本测试

仅用于测试JavaScript脚本，非真实数据。

数据上报 写值

```

{
  "deviceName": "dev_nm1",
  "datas": [
    {
      "nm": "nm1",
      "v": "0"
    },
    {
      "nm": "nm2",
      "v": "0"
    }
  ]
}

```

```

var json = {};
var obj = JSON.parse(payload); // 反序列化，将MQTT Json转为对象

var datas = []; // 存放数据点
var len = obj.datas.length; // 写值时数据点数量
json.deviceName = obj.deviceName; // PLC名称

for (let i = 0; i < len; i++) {
  var data = obj.datas[i];

  var a = {};
  a.name = data.nm; // 数据点名称
  a.value = data.v; // 数据点写入的值
}

```

测试

```

{
  "deviceName": "dev_nm1",
  "datas": [
    {
      "name": "nm1",
      "value": "0"
    },
    {
      "name": "nm2",
      "value": "0"
    }
  ]
}

```

保存并关闭 取消

输出结果需人工核对格式是否与默认 MQTT 报文格式中写值格式一致。

3.7.4. 标签配置



服务/转发配置页添加标签。

标签

	标签名	值
标签1		
标签2		
标签3		
标签4		
标签5		

标签仅对MQTT有效

确定

取消

最多设置 5 个标签。标签由所有 MQTT 共有，会自动添加到 MQTT 上报数据中。不使用标签全部留空即可。

3.7.5. 默认 MQTT 报文格式说明

➤ 上报数据格式：

```
{
  "datas": [ // 数据点列表
    {
```

```
    "name": "nm1",    //数据点名称
    "value": 1        //值
  },
  {
    "name": "nm2",
    "value": 96.32
  },
  {
    "name": "nm3",
    "value": "abc123456"
  },
],
"deviceName": "dev_nm1",    //设备名称
"sn": "test123456789",    //设备 SN 号
"time": "2024-01-01 00:00:00",    //北京时间
"TAG1": "tag1",    //标签，以实际配置的标签名和值为准。没有配置标签时此处不会添加标签。
"TAG2": "tag2",    //标签
"TAG3": "tag3",    //标签
"TAG4": "tag4",    //标签
"TAG5": "tag5",    //标签
}
```

格式如上，按照配置的设备分开上报数据。deviceName 为新建设备时的名称，datas 内为所有读取到的数据点信息，配置的数据点无效时将不会上报。datas 内 name 为配置数据点的名称，value 为实际读取到的值。value 除数据类型为 string 类型外，在 Json 中都使用 Number 的形式。

➤ 写值数据格式:

```
{  
  "deviceName":"dev_nm1",    //设备名称  
  "datas":[ // 要写值的数据点列表  
    {  
      "name":"nm1", //数据点名称  
      "value":"88"  //值  
    },  
    {  
      "name":"nm2",  
      "value":"abc888"  
    }  
  ]  
}
```

格式如上，需按照设备分开写值，不同设备写值需分开向设备发送。deviceName 为设置的设备名称，datas 为要写值的数据点列表，可单个数据点写值，也可多个数据点同时写值。datas 内那么为设置的数据点名称，value 为要写入的值，写值时统一使用 string 形式。数据错误时不能正确写入。

3.7.6. 自定义报文格式

自定义报文格式使用 JavaScript 脚本方式实现格式转换。

➤ 数据上报格式转换

数据上报时 JavaScript 脚本可操作对象为 payload。

payload 属性如下:

```
payload.sn          //设备 SN 号  
payload.deviceName  //设备名称
```

```
payload.time           //北京时间 "2024-01-01 00:00:00"
payload.timestamp      //UTC 时间戳, 毫秒
payload.TAG1           //标签以实际配置的标签名为准。最多 5 个标签
                        //如配置的标签名为 deviceId, 则对应为 payload.deviceId
payload.datas[]        //数据点列表, 数组
payload.datas.length   //数据点数组长度
// 数据点属性如下
payload.datas[0].name  //数据点名称
payload.datas[0].addr  //数据点地址
payload.datas[0].dataType //数据点数据类型
payload.datas[0].valid //数据是否有效, 0 无效 1 有效
payload.datas[0].value //数据
```

示例 1: 如需上报如下格式报文。

```
{
  "tag": "dev_nm1",      //对应设备名
  "stationid": "test",  //可添加一个名为 stationid 的标签
  "datas": [
    {
      "nm": "nm1",
      "t": "2024-05-28 14:02:40",
      "v": 0,
      "type": "int32"
    },
  ],
}
```

```
{
  "nm": "nm2",
  "t": "2024-05-28 14:02:40",
  "v": 0,
  "type": "int32"
},
{
  "nm": "nm3",
  "t": "2024-05-28 14:02:40",
  "v": 0,
  "type": "int32"
},
{
  "nm": "nm4",
  "t": "2024-05-28 14:02:40",
  "v": 0,
  "type": "int32"
}
]
```

可使用此 JavaScript 脚本进行转换格式。

```
var json = {};

json.tag = payload.deviceName; // 设备名称对应 tag

json.stationid = payload.stationid; // 标签 stationid 对应 stationid

var time = payload.time;
```

```
var datas = [];    // 存放转换后的数据点信息
var len = payload.datas.length; //数据点数量
for (let i = 0; i < len; i++) {
    var data = payload.datas[i];

    if(data.valid==1){    //数据是否有效
        var a = {};
        a.nm = data.name;    // 数据点名称
        a.t = time;        // 时间
        a.v = data.value;    // 值
        a.type=data.dataType; // 数据类型
        datas.push(a);
    }
}
json.datas = datas;
JSON.stringify(json); //转 Json，序列化,。格式化 json 使用 JSON.stringify(json, null, 4);
```

最后一行 `JSON.stringify(json);`必须填写且后面不能有 JavaScript 语句才能正确返回报文。

示例 2：如需要将 Json 数据部分转成 **Key-Value** 方式可参考以下转换脚本。

```
var json = {};
json.tag = payload.deviceName;    // 设备名称对应 tag
json.stationid = payload.stationid; // 标签 stationid 对应 stationid
var time = payload.time;

var datas = {};    // 存放转换后的数据点信息
```

```
var len = payload.datas.length; //数据点数量
for (let i = 0; i < len; i++) {
    var v = payload.datas[i];
    if(v.valid==1){ //数据是否有效
        datas[v.name]=v.value; //转换为 key-value 形式
    }
}
json.datas = datas;
JSON.stringify(json); //转 Json, 序列化,。格式化 json 使用 JSON.stringify(json, null, 4);
```

输出的格式如下

```
{
  "tag": "dev_nm1",
  "stationid": "test",
  "datas": {
    "nm1": 1234,
    "nm2": 1,
    "nm3": "123456",
    "nm4": 3.1415
  }
}
```

➤ 写值报文格式转换

写值时，必须将 Json 格式转换为默认 MQTT 格式中的写值数据格式。写值时 JavaScript 脚本可操作对象为 payload，payload 为 MQTT 接收到的 Json 文本。

示例 1：如 MQTT 下发格式格式为以下格式，进行转换。

```
{
```

```
"deviceName": "dev_nm1",
"datas": [
  {
    "nm": "nm1",
    "v": "0"
  },
  {
    "nm": "nm2",
    "v": "0"
  },
  {
    "nm": "nm3",
    "v": "0"
  }
]
```

可使用如下 JavaScript 脚本：

```
var json = {};
var obj = JSON.parse(payload); // 反序列化，将 MQTT Json 转为对象

var datas = []; // 存放数据点
var len = obj.datas.length; // 写值时数据点数量
json.deviceName = obj.deviceName; // PLC 名称

for (let i = 0; i < len; i++) {
```

```
var data = obj.datas[i];

var a = {};

a.name = data.nm;      // 数据点名称
a.value = data.v;      // 数据点写入的值
datas.push(a);
}
json.datas = datas;
JSON.stringify(json);    //序列化，转为 Json。格式化 json 使用 JSON.stringify(json, null, 4);
```

最后一行 `JSON.stringify(json);` 必须填写且后面不能有 JavaScript 语句才能正确返回转换后 Json。

示例 2：如 MQTT 下发格式格式为如下 key-value 格式进行转换。

```
{
  "deviceName": "dev_nm1",
  "datas": {
    "nm1": "0",
    "nm2": "0",
    "nm3": "0"
  }
}
```

可使用如下 JavaScript 脚本：

```
var json = {};
var obj = JSON.parse(payload); // 反序列化，将 MQTT Json 转为对象

var datas = [];                // 存放数据点
json.deviceName = obj.deviceName; // PLC 名称
```

```
Object.entries(obj.datas).forEach(([key, value]) => { //key value 为对象的属性和值
    var a={name:key, value:value} //key 对应 name, value 对 value 值
    datas.push(a);
});
json.datas = datas;
JSON.stringify(json); //序列化, 转为 Json。格式化 json 使用 JSON.stringify(json, null, 4);
```

第四章 数据采集实例

4.1. IEC104 数据采集实例

使用 IEC104 Server Simulator 软件模拟 IEC104 设备。

1、连接设备

将网关与电脑连接。网关 IP: 192.168.1.37, 电脑 IP: 192.168.1.8

2、采集配置

IP
超时时间
COA

端口
采集间隔

参数设置

添加

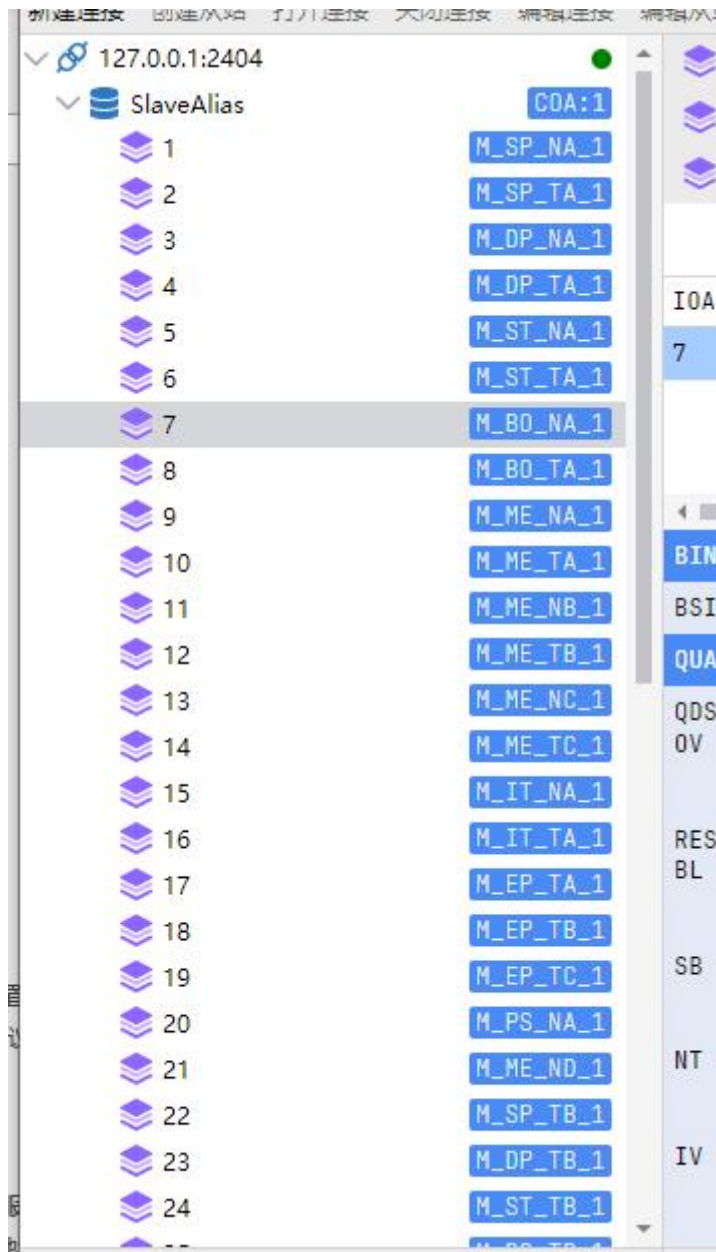
清空

导入

导出

序号	名称/标签	IOA地址	ASDU类型	数据类型	删除
1		1	M_SP_NA_1		×
2		2	M_SP_TA_1		×
3		3	M_DP_NA_1		×
4		4	M_DP_TA_1		×
5		5	M_ST_NA_1		×
6		6	M_ST_TA_1		×
7		7	M_BO_NA_1		×
8		8	M_BO_TA_1		×
9		9	M_ME_NA_1		×
10		10	M_ME_TA_1		×
11		11	M_ME_NB_1		×
12		12	M_ME_TB_1		×
13		13	M_ME_NC_1		×
14		14	M_ME_TC_1		×
15		15	M_ME_NA_1		×

3、IEC104 Server Simulator 软件配置



4、配置 Modbus TCP 和 MQTT

保持寄存器 输入寄存器 线圈 离散输入

导入数据点

清空

序号	设备名称	变量/标签名	数据类型	Modbus地址	长度	删除
0	2	1	bool	0	1	×
1	2	2	bool	1	1	×
2	2	3	int8	2	1	×
3	2	4	int8	3	1	×
4	2	5	int8	4	1	×
5	2	6	int8	5	1	×
6	2	7	uint32	6	2	×
7	2	8	uint32	8	2	×
8	2	9	float	10	2	×
9	2	10	float	12	2	×
10	2	11	int16	14	1	×
11	2	12	int16	15	1	×
12	2	13	float	16	2	×
13	2	14	float	18	2	×
14	2	15	int32	20	2	×
15	2	16	int32	22	2	×
16	2	17	int8	24	1	✓

IP/域名	192.168.1.8
端口	1883
客户端ID	
用户名	b
密码	
Qos	1
报文格式	☒ 默认 ☐ 自定义
	测试脚本
上报主题	report2
上报JS	
写值主题	write
写值JS	

5、数据监控

通过 MQTT.fx 监控 MQTT 上传到服务器的数据。

```
report2
05-06-2025 11:17:05.40625805
142
QoS 0

"value" : 0
}, {
  "name" : "2",
  "value" : 0
}, {
  "name" : "3",
  "value" : 2
}, {
  "name" : "4",
  "value" : 0
}, {
  "name" : "5",
  "value" : -24
}, {
  "name" : "6",
  "value" : -55
}, {
  "name" : "7",
  "value" : 1029
}, {
  "name" : "8",
  "value" : 0
}, {
  "name" : "9",
  "value" : 0.849971
}, {
```

通过 Modbus poll 查看数据。

Mbpolli

Tx = 30818: Err = 19949: ID = 1: F = 03: SR = 1000ms

	Alias	00000	Alias	00010	Alias	00020
0		0		0.849971		0
1		1		--		0
2		1		0.955992		0
3		0		--		0
4		-29		453		0
5		-60		0		0
6		1029		5836.5		0
7		--		--		0
8		0		0		0
9		0		0		0

4.2. DLT645（电表）数据采集实例

- 1、网关与电表通过 485 连接。





2、配置电表

波特率

停止位

通信超时时间

数据位

校验位

采集时间间隔

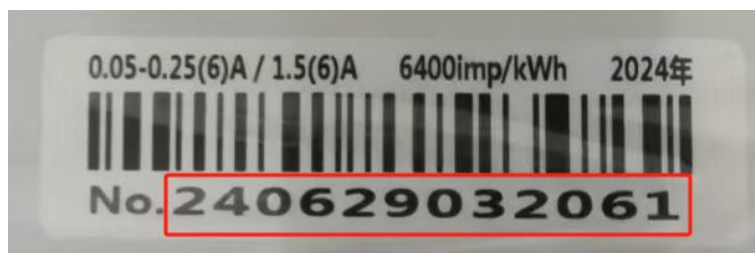
1
✖

协议版本:

通讯地址/电表号: ☒ 全选

序号	选择	名称/标签	数据项名称
34	☒		总功率因数
35	☒		A相功率因数
36	☒		B相功率因数
37	☒		C相功率因数
38	☒	A相电流	A相电流
39	☒	B相电流	B相电流
40	☒	C相电流	C相电流
41	☒	A相电压	A相电压
42	☒	B相电压	B相电压
43	☒	C相电压	C相电压

选择电表协议版本。按照电表上的表号填写电表号。



电表可能不支持部分数据项采集，不支持的数据不会通过 MQTT 上传到服务器。**不支持的数据项建议取消选中**，避免影响采集。

MQTT.fx 查看 MQTT 上报服务器数据。

```
report2
06-06-2025 14:21:27.51687738 1599 QoS
{
  "name": "正向有功费率4电能",
  "value": 0
}, {
  "name": "正向有功费率3电能",
  "value": 0.03
}, {
  "name": "A相电流",
  "value": 800
}, {
  "name": "B相电流",
  "value": 800
}, {
  "name": "C相电流",
  "value": 0
}, {
  "name": "A相电压",
  "value": 0.1
}, {
  "name": "B相电压",
  "value": 0.2
}, {
  "name": "C相电压",
  "value": 237.3
}, {
  "deviceName": "645",
}
```

第五章 售后及联系方式

公司网址: www.tj-sange.com www.sange-cbm.com

售前购买咨询: 176-0260-2061 (同微信)

售后技术电话: 022-2210-6681

公众账号: 获取产品使用视频和更多资讯。

