

PLC 设备跟 HTTP 协议系统平台对接的 JSON 格式配置举例

智能网关 IGT-DSER 集成了多种 PLC 的原厂协议，方便实现各种 PLC、智能仪表通过 HTTP 协议与 MES 等各种系统平台通讯对接。PLC 内不用编写程序，设备不用停机，通过网关的参数配置软件([下载地址](#))配置 JSON 文件的字段与 PLC 寄存器地址等参数即可。



智能网关 IGT-DSER 支持 POST/GET/PUT 等多种方法，可同时作为 HTTP 协议的客户端和服务端。作为客户端通讯时将 JSON 文件提交给 HTTP 的服务端，如果服务端有返回的 JSON，网关进行解析后将数据写入到 PLC，实现 PLC 与 HTTP 服务端双向通讯；作为服务端时根据客户端 URL 中的路径查找所配置的数据，打包成 JSON 文件后返回给客户端。[相关案例](#)

网关支持触发、周期、混合等多种工作模式：周期模式是按设置的周期时间提交请求和解析返回的文件；触发模式可配置一个或多个触发条件，网关根据触发条件的数据变化进行文件提交和下载解析；也可以两种模式同时配置运行。

本文例采用西门子 PLC，通过参数软件(下载地址)的'功能'->'HTTP-GET/POST'，进入参数配置页面，以下是数据配置表自动生成的 JSON 文件的对应。JSON 文件内的数据可以自由增减，字符可自由修改，举例只是为了描述各种 JSON 文件的结构实现。MQTT 协议的发布/订阅也是同样的对应方式。具体如下：

1. 多个 Key:Value 的同级组合：

序号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX 数据	tag1	DB	1	00012	DINT32	网口2	0	☐	
0002	XX 数据	tag2	DB	1	00016	REAL32	网口2	0	☐	
0003	XX 数据	tag3	DB	1	00020	CHAR[n]	网口2	0	☐	

```
{  
  "tag1": 12345678,  
  "tag2": 12345.678,  
  "tag3": "ABCD1234"  
}
```

2. 包含子对象:

序号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX 标题	clientID	Script	0	0	CHAR[n]	网口2	0	☐	IGT-DSER
0002	XX 数组对象	datas	Script	0	0	CHAR[n]	网口2	0	☐	
0003	XX 数据	tag1	DB	2	00120	UINT16	网口2	0	☐	
0004	XX 数据	tag2	DB	2	00122	BYTE	网口2	0	☐	
0005	XX 数据	tag3	M	0	00012.0	BOOL	网口2	0	☐	

```

{
    "clientID": "IGT-DSER",
    "datas": {
        "tag1": 55555,
        "tag2": 222,
        "tag3": 1
    }
}

```

3. 增加时间戳，多种格式可以切换：

序号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初值
0001	日期时间	time	Script	0	0	CHAR[n]	网口2	0	☐	IGT-DSER
0002	XX 标题	clientID	Script	0	0	CHAR[n]	网口2	0	☐	
0003	XX 数组对象	datas	Script	0	0	CHAR[n]	网口2	0	☐	
0004	XX 数据	tag1	DB	6	00020	UINT32	网口2	0	☐	
0005	XX 数据	tag2	DB	6	00042	CHAR[n]	网口2	0	☐	

```

{
  "time": 1573198662,
  "clientID": "IGT-DSER",
  "datas": {
    "tag1": 98762345,
    "tag2": "P02101已完成"
  }
}

```

时间戳(单位秒,10位长度)

YYYY-MM-DD hh:mm:ss

时间戳(单位秒,10位长度)

时间戳(单位毫秒,13位长度)

4. 包含多个子对象:

序号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	日期时间	time	Script	0	0	CHAR[n]	网口2	0	☐	IGT-DSER
0002	XX标题	clientID	Script	0	0	CHAR[n]	网口2	0	☐	
0003	XX数组对象	datas	Script	0	0	CHAR[n]	网口2	0	☐	
0004	XX数据	tag1	DB	12	00000	DINT32	网口2	0	☐	
0005	XX数据	tag2	DB	12	00004	REAL32	网口2	0	☐	
0006	XX数组对象	product	Script	0	0	CHAR[n]	网口2	0	☐	
0007	XX数据	tag3	DB	15	00120	CHAR[n]	网口2	0	☐	
0008	XX数据	tag4	DB	15	00160	CHAR[n]	网口2	0	☐	

```

{
    "time": 1573198662,
    "clientID": "IGT-DSER",
    "datas": {
        "tag1": 2837465,
        "tag2": 562312.668
    },
    "product": {
        "tag3": "P020102-01",
        "tag4": "012x60"
    }
}

```

5. 对象数组结构:

序 号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX数组对象	datas	Script	15	0	CHAR[n]	网口2	0	☐	
0002	XX数据	tag1[0]	DB	12	00000	DINT32	网口2	0	☐	
0003	XX数据	tag2[0]	DB	12	00004	REAL32	网口2	0	☐	
0004	XX数据	tag1[1]	DB	15	00000	DINT32	网口2	0	☐	
0005	XX数据	tag2[1]	DB	15	00004	REAL32	网口2	0	☐	

```
{
  "datas": [
    {
      "tag1": 2837465,
      "tag2": 562312.668
    },
    {
      "tag1": 6734625,
      "tag2": 345267.232
    }
  ]
}
```

6. 子对象+对象数组:

序 号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX数组对象	product	Script	15	0	CHAR[n]	网口2	0	☐	
0002	XX数据	name	DB	120	00000	CHAR[n]	网口2	0	☐	
0003	XX数据	count	DB	120	00000	DINT32	网口2	0	☐	
0004	XX数组对象	datas	Script	15	0	CHAR[n]	网口2	0	☐	
0005	XX数据	tag1[0]	DB	12	00000	DINT32	网口2	0	☐	
0006	XX数据	tag2[0]	DB	12	00004	REAL32	网口2	0	☐	
0007	XX数据	tag1[1]	DB	15	00000	DINT32	网口2	0	☐	
0008	XX数据	tag2[1]	DB	15	00004	REAL32	网口2	0	☐	

```
{
  "product": {
    "name": "PN2205",
    "count": 65678
  },
  "datas": [
    {
      "tag1": 2837465,
      "tag2": 562312.668
    },
    {
      "tag1": 6734625,
      "tag2": 345267.232
    }
  ]
}
```

7. 子对象内再增加子对象：

序 号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX标题	clientID	Script	0	0	CHAR[n]	网口2	0	☐	IGT-DSER
0002	XX数组对象	datas	Script	0	0	CHAR[n]	网口2	0	☐	
0003	XX数据	tag21	DB	15	00000	CHAR[n]	网口2	0	☐	
0004	XX数据	tag22	DB	15	00000	CHAR[n]	网口2	0	☐	
0005	XX数据	params	Script	15	0	CHAR[n]	网口2	0	☐	\$(subjson)
0006	数值读取	subjson	DB	6	00012	DINT32	网口2	0	☐	#{ "tag31": ## ,
0007	数值读取	subjson	DB	6	00016	REAL32	网口2	0	☐	#{ "tag32": ## }}

```
{
  "clientID": "IGT-DSER",
  "datas": {
    "tag21": "P020102-01",
    "tag22": "012x60",
    "params": {
      "tag31": 12345678,
      "tag32": 12345.677734
    }
  }
}
```

8. 外加方括号：

序 号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX数据	tag11	DB	12	00000	DINT32	网口2	0	☐	
0002	XX数据	tag12	DB	12	00004	REAL32	网口2	0	☐	
0003	XX数据	tag13	DB	15	00000	DINT32	网口2	0	☐	
0004	XX数据	tag14	DB	15	00004	REAL32	网口2	0	☐	

```
[
  {
    "tag11": 2837465,
    "tag12": 562312.668,
    "tag13": 6734625,
    "tag14": 345267.232
  }
]
```

9. 数组形式整体组合：

序 号	字段类别	字段名称	数据区域	DB号码	数据地址	数据类型	通讯端口	IP/站号/组	变化判断	数据处理/初始值
0001	XX数据	tag1	DB	12	00000	DINT32	网口2	1	☐	
0002	XX数据	tag2	DB	12	00004	REAL32	网口2	1	☐	
0003	XX数据	tag1	DB	15	00000	DINT32	网口2	2	☐	
0004	XX数据	tag2	DB	15	00004	REAL32	网口2	2	☐	

```
[
  {
    "tag1": 2837465,
    "tag2": 562312.668
  },
  {
    "tag1": 6734625,
    "tag2": 345267.232
  }
]
```

图上'字段类别'栏目中所对应的'XX', 在 HTTP 协议里对应'上报'和'下载', 在 MQTT 协议里对应'发布'和'订阅'。另外智能网关也支持边缘计算脚本编程([手册下载](#)), 通过脚本程序可以实现任意格式的 JSON、XML, 以及逗号分隔符等各种非标准格式的文件机构, 以及数据的运算处理。

西门子 PLC 为 S7-1500 系列, IP:192.168.2.11, 与 IGT-DSEr 智能网关以太网通讯, 配置的网络参数, 如下图:

网口1.接口类型	有线网卡RJ45接口	
网口1.IP地址	192.168.2.5	网关网口1的自身的网络参数:
网口1.子网掩码	255.255.255.0	本案例该网口连接PLC, IP设置为PLC的同网段;
网口1.默认网关	192.168.3.1	默认网关一般设置为连接数据服务器的路由器的IP地址;
网口1.PLC设备IP地址(0-254)	11	PLC参数: PLC的IP地址只需设置最后一段, 前三段与网关的IP相同;
网口1.PLC网络端口与协议	102	西门子的网络通讯端口是102;
网口2.接口类型	有线网卡RJ45接口	
网口2.IP地址	192.168.3.5	网关网口2的自身的网络参数:
网口2.子网掩码	255.255.255.0	本案例该网口连接服务器;
网口2.默认网关	192.168.3.1	默认网关一般设置为连接数据服务器的路由器的IP地址;
网口2.PLC设备IP地址(0-254)		
网口2.PLC网络端口与协议	不连接PLC, 无需设置;	
网口3.接口类型	端口关闭	
网络服务.接口类型	上位机通讯	
网络服务.端口1.网络端口	102	西门子S7协议服务端口:
网络服务.端口1.通讯协议	0_透明传输/IP耦合隔离	支持OPC、组态软件数据通讯;
网络服务.端口1.对应输出接口	网口1	支持高级语言数据通讯;
网络服务.端口1.超时判定时间	3000	
网络服务.端口2.网络端口	502	ModbusTCP协议服务端口:
网络服务.端口2.通讯协议	1_ModbusTCP协议	一般设置为502, 通过'功能'->'Modbus地址表'编辑地址;
网络服务.端口2.对应输出接口	网口1	支持OPC、组态软件数据通讯;
网络服务.端口2.超时判定时间	3000	

设置好网络参数, 按上图配置 JSON 格式, 即可完成 PLC 跟 HTTP 系统平台的对接。