

物联网数据平台对接说明

| 版本 | 时间 | 更新说明 |
|------|------------|-------------------------------------|
| 1.01 | 2018-06-12 | |
| 1.02 | 2018-08-11 | 1、更新部分文字描述 2、增加目录 3、增加 4.2.12 |

目录

| | |
|-------------------------|----|
| 物联网数据平台对接说明 | 0 |
| 一、对接方式 | 2 |
| 1.1 数据推送方式（推荐） | 2 |
| 1.2 数据查询方式 | 2 |
| 1.3 设备远程控制 | 2 |
| 二、全局说明 | 2 |
| 2.1 适用范围 | 2 |
| 2.2 安全认证 | 2 |
| 2.2.1 数据推送和查询 | 2 |
| 2.2.2 设备远程控制 | 3 |
| 2.2.3 测试服务器 | 4 |
| 三、数据接口 | 5 |
| 3.1 数据推送和查询 | 5 |
| 3.1.1 抄表数据推送 | 5 |
| 3.1.2 获取单个设备的实时数据 | 7 |
| 3.1.3 获取多个设备的实时数据 | 9 |
| 3.1.4 获取历史数据 | 10 |
| 3.2 告警推送和查询 | 11 |
| 3.2.1 告警数据推送 | 11 |
| 3.2.2 获取数据列表 | 12 |
| 3.3 设备事件推送和查询 | 13 |
| 3.3.1 设备事件推送 | 13 |
| 3.3.2 获取数据列表 | 14 |
| 四、远程控制接口 | 15 |
| 4.1 参数说明 | 15 |
| 4.2 远程控制接口 | 16 |
| 4.2.1 添加 | 16 |
| 4.2.2 修改 | 17 |
| 4.2.3 删除 | 18 |
| 4.2.4 获取实体 | 19 |
| 4.2.5 获取实体列表 | 20 |
| 4.2.6 取消命令 | 21 |
| 4.2.7 重试命令 | 22 |
| 4.2.8 批量添加 | 22 |
| 4.2.9 批量修改 | 24 |
| 4.2.10 批量删除、取消、重试 | 26 |
| 4.2.11 获取命令详情 | 27 |
| 4.3 物联网表远程控制数据结构 | 30 |
| 4.3.1 数据项值 | 30 |
| 4.3.2 命令类型 | 30 |
| 3.5.3 数据结构(C#) | 31 |

一、对接方式

1.1 数据推送方式（推荐）

平台会在收到设备数据后，推送到应用方。此种方式下数据只保存在应用方；平台在推送不成功的时候会缓存 7 天的数据，待应用恢复正常时重新推送。

此方式需要应用方实现指定的几个接口用于接收数据，并自己设计数据库保存数据。

1.2 数据查询方式

平台收到设备数据后，会将数据保存在平台的数据库中。此种方式下，需要对接应用通过平台提供的接口查询数据。

1.3 设备远程控制

平台提供设备控制命令发送接口，通过调用此接口实现对设备的远程控制，如阀门开关、调价等。

二、全局说明

2.1 适用范围

本接口用于向第三方应用提供抄表数据；接口采用 WEBAPI 方式，http 协议 RESTful 风格，json 格式传输数据，编码格式为 UTF8。

2.2 安全认证

2.2.1 数据推送和查询

采用 key 认证方式，访问接口时需要将 Key 放置在 http 请求头中，例如：
POST /api/Data/MeterData/GetHistoryList

content-type: application/json
company_no: 1001
data_api_key: 496dlbac094f11e79d9c0a0027000010

| 名称 | 说明 | 备注 |
|--------------|-----------------------|----|
| company_no | 项目公司编号，4 个数字字符，由管理员分配 | |
| data_api_key | 数据接口访问 key | |

2.2.2 设备远程控制

采用口令验证（token）方式。首先需要使用用户登陆接口（api/Conf/Account/Login）获取口令（token），然后将口令放置在 http 请求头中。例如：

第一步：登录

请求 post api/Conf/Account/Login?loginName={loginName}&pwd={pwd}

| 名称 | 说明 | 备注 |
|-----------|---------------------|----|
| loginName | 登录用的管理员用户名 | |
| pwd | 登录密码，将密码做 MD5 加密后传入 | |

返回

| 名称 | 说明 | 备注 |
|----------------|----------------|----|
| token | 返回的口令 | |
| expiration | 口令过期时间 | |
| admin_id | 当前登录管理员的 ID 号 | |
| company_no | 当前登录管理员所属的项目公司 | |
| admin_name | 管理员用户名 | |
| admin_remark | 备注 | |
| admin_ext_data | 其他数据 | |

返回样例

```
{  
  "Data": {  
    "token":  
    "1C4810F517A173B68C28F75F0302D28A2F16C8523219AEA21D129E637CC7B31BF5F8  
A88E6A903891542D60BD23FD6A21A0E90CB65CD4805CB50D048C604650605B84F77DC  
7CDEFF00071F5DF512BB1A4FC4F8A0D865FB78AC789506E1075869D1C2C06D9A2ABD3  
0E78343E80E71A234866BD9FF980240F856ECEAA5B7C51AC76623B68ED44CDFE17C13
```

```

67A76C802CE478D424B4595C99CE8224B6EA9E8D037F84B8C26B16FF40EC498B5BD89
6614C3214B437CC3107F396BE2180EB269F5F5B1",
    "expiration": "2017-05-17T11:17:31.6895023+08:00",
    "admin_id": 1,
    "company_no": "1001",
    "admin_name": "admin",
    "admin_remark": "test",
    "admin_ext_data": "{\\"a\\": \\"b\\"}"
  },
  "State": "success",
  "Message": ""
}

```

第二步. 获取数据

```

var settings = {
  "async": true,
  "crossDomain": true,
  "url": "http://localhost:5192/api/Conf/Constants/CompanyInfo",
  "method": "POST",
  "headers": {
    "content-type": "application/json",
    "token":
"DFD5752191B83A356483476DDF8D0637C82CFF279FB625906D266A594828832310D5
5712B445333560C789052689DF7D783F7F761C9A3BE0ECA8E724CB052161ECF5ED09F
3DD7093E827CD2FB7AB4328949E033EE79FE68D1219F14386B33D32569BD77455C657
9CE47B9B75F846F7AB0A794115E0458EE923834B8672160765D2E8031553E869D3798
A8F082207BE2A4D146646722A543958F5E836F820820C644C4F607D7318656C765733
3B8C05C91C646C317DE46E5218385A24E6DE6EB6"
  }
}

$.ajax(settings).done(function (response) {
  console.log(response);
});

```

2.2.3 测试服务器

数据接口

Url 地址: http://120.55.75.63:2102

认证 KEY:

company_no: 1001

data_api_key: 496dlbac094f11e79d9c0a0027000010

远程控制接口

Url 地址: http://120.55.75.63:2101

用户名: admin

密码: 123456

三、数据接口

3.1 数据推送和查询

当采用推送方式时，应用方必须按照下列接口标准进行开发；当对接方式为查询时，由平台提供下列接口。

当应用方开发接口时，3.1.1 是必须实现的，其他可根据需要实现。

3.1.1 抄表数据推送

请求: POST {{baseurl}}/api/Data/MeterData/Add

Body 参数:

| 名称 | 数据类型 | 说明 |
|------------------|---------------|------------------------|
| meter_comm_no | nvarchar(50) | 通信号，物联网表 14 位，无线表 10 位 |
| device_type | int | 设备类型 |
| device_type_name | nvarchar(50) | 设备类型名称 |
| factory_code | nvarchar(50) | 厂家代码 |
| meter_area_no | nvarchar(50) | 分区号 |
| meter_define_no1 | nvarchar(50) | 用户自定义编号 1 |
| meter_define_no2 | nvarchar(50) | 用户自定义编号 2 |
| meter_define_no3 | nvarchar(50) | 用户自定义编号 3 |
| meter_define_no4 | nvarchar(50) | 用户自定义编号 4 |
| data_type | Int | 数据类型 (0-实时数据 1-冻结数据) |
| data_type_name | nvarchar(50) | 数据类型名称 |
| acquisition_time | Datetime | 采集时间/冻结时间 |
| receive_time | Datetime | 数据接收时间 |
| std_sum | Decimal(18,3) | 标况总累计量 |
| work_sum | Decimal(18,3) | 工况总累计量 |
| std_flow | Decimal(18,3) | 标况瞬时流量 |
| work_flow | Decimal(18,3) | 工况瞬时流量 |
| temperature | Decimal(18,3) | 温度 |

| | | |
|-----------------|---------------|----------------|
| pressure | Decimal(18,3) | 压力 |
| magatt_times | Int | 磁攻击次数 |
| this_month_sum | Decimal(18,3) | 当月累计用量(来自表内) |
| last_month_sum | Decimal(18,3) | 上月累计用量(来自表内) |
| this_cycle_sum | Decimal(18,3) | 本周周期累计用量(来自表内) |
| last_cycle_sum | Decimal(18,3) | 上周周期累计用量(来自表内) |
| remain_money | Decimal(18,3) | 剩余金额 |
| remain_volume | Decimal(18,3) | 剩余量 |
| curr_price | Decimal(18,3) | 当前价格 |
| meter_state | Int | 表状态 |
| meter_stat_emsg | nvarchar(max) | 表状态解析信息 |
| rsssi | Int | 信号强度 |
| battery_voltage | Decimal(18,3) | 电池电压 |
| battery_level | Int | 电池电量 |
| ext_data | JSON | 其他数据 |

ext_data

```

//rtu 状态值
public int? rtu_state { get; set; } = 0;
//rtu 状态信息
public string rtu_state_msg { get; set; } = "";
//表底数
public decimal? base_volume { get; set; } = 0;
//累购量
public decimal? total_buy_value { get; set; } = 0;
//表计状态数据:
//开户标志, true-已开户、false-未开户
public bool? open_state { get; set; } = false;
//阀门状态, true-开阀、false-关阀
public bool? valve_open { get; set; } = false;
//低电等级, 0-正常, 1-一级低电, 2-二级低电
public int? power_level { get; set; } = 0;
//远程关阀, 无线阀控, true-有、false-无
public bool? remote_valve_off { get; set; } = false;
//拆表关阀, true-有、false-无
public bool? apart_valve_off { get; set; } = false;
//大流量关阀, true-有、false-无
public bool? max_flow_valve_off { get; set; } = false;
//小流量关阀, true-有、false-无
public bool? min_flow_valve_off { get; set; } = false;
//死表关阀, true-有、false-无
public bool? dead_valve_off { get; set; } = false;
//磁攻击, true-有、false-无

```

```

public bool? magnetic_attack { get; set; } = false;
//外电情况, true-有、false-无
public bool? ext_power { get; set; } = false;
//表计费类型, 0-阶梯费率 1-计量表
public int? calc_type { get; set; } = 0;
//脉冲故障, 干黄故障, true-有、false-无
public bool? pulse_fault { get; set; } = false;
//电池断电, 扣电, true-有、false-无
public bool? battery_cut { get; set; } = false;
//强制开阀, true-有、false-无
public bool? force_valve_on { get; set; } = false;
//RTU 状态数据
//门禁状态 (RTU), true-门开、false-门关
public bool? rtu_door_open { get; set; } = false;
//市电状态 (RTU), true-有、false-无
public bool? rtu_power { get; set; } = false;
//围栏异常报警 (RTU), true-有、false-无
public bool? rtu_fence_alarm { get; set; } = false;
//振动报警 (RTU), true-有、false-无
public bool? rtu_vibrate_alarm { get; set; } = false;

```

返回:

| 名称 | 类型 | 说明 | 备注 |
|---------|--------|---------------------|------------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Data | String | 返回新的 data_guid 值 | |

返回样例:

```

{
  "Data": "LA6owvN3SECEP2zL2fkYxg==",
  "State": "success",
  "Message": ""
}

```

3.1.2 获取单个设备的实时数据

请求: post {{baseurl}}/api/Data/MeterData/GetRealModel/{meter_comm_no}

URL 参数:

| 名称 | 数据类型 | 说明 |
|---------------|--------------|--------------------------|
| meter_comm_no | nvarchar(50) | 通信号, 物联网表 14 位, 无线表 10 位 |

样例: {{baseurl}}/api/Data/MeterData/GetRealModel/1234567890

返回:

| 名称 | 类型 | 说明 | 备注 |
|---------|--------|---------------------|---------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Data | Object | 成功返回实体对象, 失败返回 null | |

Data:

| 名称 | 数据类型 | 说明 |
|------------------|------------------------|--------------------------|
| data_guid | BINARY(16) PRIMARY KEY | uuid 数据标识 |
| company_no | nvarchar(10) | 项目公司编号, 4 个数字字符 |
| meter_comm_no | nvarchar(50) | 通信号, 物联网表 14 位, 无线表 10 位 |
| device_type | int | 设备类型 |
| device_type_name | nvarchar(50) | 设备类型名称 |
| factory_code | nvarchar(50) | 厂家代码 |
| meter_area_no | nvarchar(50) | 分区号 |
| meter_define_no1 | nvarchar(50) | 用户自定义编号 1 |
| meter_define_no2 | nvarchar(50) | 用户自定义编号 2 |
| meter_define_no3 | nvarchar(50) | 用户自定义编号 3 |
| meter_define_no4 | nvarchar(50) | 用户自定义编号 4 |
| data_type | Int | 数据类型 (0-实时数据 1-冻结数据) |
| data_type_name | nvarchar(50) | 数据类型名称 |
| acquisition_time | Datetime | 采集时间/冻结时间 |
| receive_time | Datetime | 数据接收时间 |
| std_sum | Decimal(18, 3) | 标况总累计量 |
| work_sum | Decimal(18, 3) | 工况总累计量 |
| std_flow | Decimal(18, 3) | 标况瞬时流量 |
| work_flow | Decimal(18, 3) | 工况瞬时流量 |
| temperature | Decimal(18, 3) | 温度 |
| pressure | Decimal(18, 3) | 压力 |
| magatt_times | Int | 磁攻击次数 |
| this_month_sum | Decimal(18, 3) | 当月累计用量(来自表内) |

| | | |
|-----------------|---------------|----------------|
| last_month_sum | Decimal(18,3) | 上月累计用量(来自表内) |
| this_cycle_sum | Decimal(18,3) | 本周周期累计用量(来自表内) |
| last_cycle_sum | Decimal(18,3) | 上周周期累计用量(来自表内) |
| remain_money | Decimal(18,3) | 剩余金额 |
| remain_volume | Decimal(18,3) | 剩余量 |
| curr_price | Decimal(18,3) | 当前价格 |
| meter_state | Int | 表状态 |
| meter_stat_emsg | nvarchar(max) | 表状态解析信息 |
| rss_i | Int | 信号强度 |
| battery_voltage | Decimal(18,3) | 电池电压 |
| battery_level | Int | 电池电量 |
| ext_data | JSON | 其他数据 |

3.1.3 获取多个设备的实时数据

请求: Post

```
{{baseurl}}/api/Data/MeterData/GetRealList2?startrow={startrow}&rowcount={rowcount}
```

url 参数 (可选参数)

| 名称 | 数据类型 | 说明 |
|----------|------|--------------|
| startrow | Int | 起始记录, 从 0 开始 |
| rowcount | Int | 要返回的记录数量 |

body 参数 (查询条件)

| 名称 | 数据类型 | 说明 |
|----------------|----------|--|
| meter_comm_nos | string[] | 通讯编号集合 |
| acq_start_time | datetime | 采集起始时间 |
| acq_end_time | datetime | 采集结束时间 |
| rec_start_time | datetime | 接收起始时间 |
| rec_end_time | datetime | 接收结束时间 |
| orderby | string | 指定要进行排序的字段名称, 可用的字段名称为: meter_comm_no, acquisition_time, receive_time, meter_define_no1, meter_define_no3 |
| sequence | string | 排序方式 asc-升序, desc-降序 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|----|------|----|----|
|----|------|----|----|

| | | | |
|------------|----------|----------------|------------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Totalcount | int | 符合查询条件的总记录数量 | |
| startrow | Int | 本次起始记录, 从 0 开始 | |
| rowcount | Int | 本次返回的记录数量 | |
| Data | Object[] | 本次返回的实体对象的列表 | |

```
{
  "Data": [
    { Data 1 ..... }, { Data 2..... }
  ],
  "Totalcount": 2,
  "Startrow": 0,
  "Rowcount": 1,
  "State": "success",
  "Message": ""
}
```

3.1.4 获取历史数据

请求 post

```
{{baseurl}}/api/Data/MeterData/GetHistoryList2?startrow={startrow}&rowcount={rowcount}
```

url 请求参数

| 名称 | 数据类型 | 说明 |
|----------|------|--------------|
| startrow | Int | 起始记录, 从 0 开始 |
| rowcount | Int | 要返回的记录数量 |

样例

```
{{baseurl}}/api/Data/MeterData/GetHistoryList2?startrow=0&rowcount=10
```

body 请求参数 (查询条件) 样例

| 名称 | 数据类型 | 说明 |
|---------------|--------|---------|
| meter_comm_no | string | 通讯编号 |
| data_type | Int | 数据类型 |
| year | Int | 采集时间, 年 |

| | | |
|--------------|-----|---------|
| month | Int | 采集时间，月 |
| start_day | Int | 起始时间，日 |
| start_hour | Int | 起始时间，小时 |
| start_minute | Int | 起始时间，分钟 |
| end_day | Int | 起始时间，日 |
| end_hour | Int | 截止时间，小时 |
| end_minute | Int | 截止时间，分钟 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|------------|----------|---------------|------------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Totalcount | int | 符合查询条件的总记录数量 | |
| startrow | Int | 本次起始记录，从 0 开始 | |
| rowcount | Int | 本次返回的记录数量 | |
| Data | Object[] | 本次返回的实体对象的列表 | |

3.2 告警推送和查询

3.2.1 告警数据推送

请求： {{baseurl}}/post api/Data/DeviceAlarm/Add

Body 参数：

| 名称 | 数据类型 | 说明 |
|----------------|---------------|-----------------|
| device_comm_no | nvarchar(50) | 设备通信编号 |
| device_type | nvarchar(50) | 设备类型 1-表计 2-RTU |
| alarm_time | datetime | 告警时间 |
| receive_time | datetime | 接收时间 |
| alarm_code | int | 告警代码 |
| alarm_content | nvarchar(500) | 告警内容 |
| alarm_data | json | 告警数据 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|---------|--------|-----------------|---------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Data | Int | 返回新的 alarm_id 值 | |

3.2.2 获取数据列表

请求: post

```
{{baseurl}}/api/Data/DeviceAlarm/GetList2?startrow={startrow}&rowcount={rowcount}
```

url 请求参数

| 名称 | 数据类型 | 说明 |
|----------|------|--------------|
| startrow | Int | 起始记录, 从 0 开始 |
| rowcount | Int | 要返回的记录数量 |

样例 api/Data/DeviceAlarm/GetList2?startrow=0&rowcount=10

body 请求参数 (查询条件) 样例

| 名称 | 数据类型 | 说明 |
|----------------|--------|-----------------|
| device_comm_no | string | 通讯编号 |
| device_type | Int | 设备类型 1-表计 2-RTU |
| year | Int | 采集时间, 年 |
| month | Int | 采集时间, 月 |
| start_day | Int | 起始时间, 日 |
| start_hour | Int | 起始时间, 小时 |
| start_minute | Int | 起始时间, 分钟 |
| end_day | Int | 起始时间, 日 |
| end_hour | Int | 截止时间, 小时 |
| end_minute | Int | 截止时间, 分钟 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|---------|--------|--------|---------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |

| | | | |
|------------|----------|---------------|--|
| Totalcount | int | 符合查询条件的总记录数量 | |
| startrow | Int | 本次起始记录，从 0 开始 | |
| rowcount | Int | 本次返回的记录数量 | |
| Data | Object[] | 本次返回的实体对象的列表 | |

Data:

| 名称 | 数据类型 | 说明 |
|----------------|---------------|-----------------|
| alarm_id | Int | 流水号 |
| company_no | nvarchar(10) | 项目公司编号，4 个数字字符 |
| device_comm_no | nvarchar(50) | 设备通信编号 |
| device_type | nvarchar(50) | 设备类型 1-表计 2-RTU |
| alarm_time | datetime | 告警时间 |
| receive_time | datetime | 接收时间 |
| alarm_code | int | 告警代码 |
| alarm_content | nvarchar(500) | 告警内容 |
| alarm_data | json | 告警数据 |

3.3 设备事件推送和查询

3.3.1 设备事件推送

请求 post {{baseurl}}/api/Data/DeviceEvent/Add

Body 参数

| 名称 | 数据类型 | 说明 |
|----------------|---------------|-----------------|
| device_comm_no | nvarchar(50) | 设备通信编号 |
| device_type | nvarchar(50) | 设备类型 1-表计 2-RTU |
| event_time | datetime | 事件时间 |
| receive_time | datetime | 接收时间 |
| event_code | int | 事件代码 |
| event_content | nvarchar(500) | 事件内容 |
| event_data | json | 事件数据 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|----|------|----|----|
|----|------|----|----|

| | | | |
|---------|--------|-----------------|---------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |
| Data | Int | 返回新的 event_id 值 | |

3.3.2 获取数据列表

请求: post

{{baseurl}}/api/Data/DeviceEvent/GetList2?startrow={startrow}&rowcount={rowcount}

url 请求参数

| 名称 | 数据类型 | 说明 |
|----------|------|--------------|
| startrow | Int | 起始记录, 从 0 开始 |
| rowcount | Int | 要返回的记录数量 |

样例 api/Data/DeviceEvent/GetList2?startrow=0&rowcount=10

body 参数 (查询条件)

| 名称 | 数据类型 | 说明 |
|----------------|--------|-----------------|
| device_comm_no | string | 通讯编号 |
| device_type | Int | 设备类型 1-表计 2-RTU |
| year | Int | 采集时间, 年 |
| month | Int | 采集时间, 月 |
| start_day | Int | 起始时间, 日 |
| start_hour | Int | 起始时间, 小时 |
| start_minute | Int | 起始时间, 分钟 |
| end_day | Int | 起始时间, 日 |
| end_hour | Int | 截止时间, 小时 |
| end_minute | Int | 截止时间, 分钟 |

返回

| 名称 | 数据类型 | 说明 | 备注 |
|---------|--------|--------|---------------------------|
| State | String | 操作状态 | success - 成功, failed - 失败 |
| Message | String | 操作状态信息 | |

| | | | |
|------------|----------|---------------|--|
| Totalcount | int | 符合查询条件的总记录数量 | |
| startrow | Int | 本次起始记录，从 0 开始 | |
| rowcount | Int | 本次返回的记录数量 | |
| Data | Object[] | 本次返回的实体对象的列表 | |

Data:

| 名称 | 数据类型 | 说明 |
|----------------|--------------------------------|-----------------|
| event_id | Int AUTO_INCREMENT PRIMARY KEY | 流水号 |
| company_no | nvarchar(10) | 项目公司编号，4 个数字字符 |
| device_comm_no | nvarchar(50) | 设备通信编号 |
| device_type | nvarchar(50) | 设备类型 1-表计 2-RTU |
| event_time | datetime | 事件时间 |
| receive_time | datetime | 接收时间 |
| event_code | int | 事件代码 |
| event_content | nvarchar(500) | 事件内容 |
| event_data | json | 事件数据 |

四、远程控制接口

由平台提供，应用调用此接口实现对设备的远程控制，如阀门开关、调价等。

4.1 参数说明

| 名称 | 数据类型 | 说明 |
|----------------|--------------------------------|-------------------|
| ctrl_id | Int AUTO_INCREMENT PRIMARY KEY | 事物流水号 |
| upper_ctrl_id | int | 上级系统事物流水号 |
| company_no | nvarchar(10) | 项目公司编号，4 个数字字符 |
| processor_name | nvarchar(50) | 处理程序标识 |
| ctrl_define_no | nvarchar(50) | 自定义命令编号 |
| target_comm_no | nvarchar(50) | 目标通信编号 |
| target_devtype | int | 目标设备类型 1-表计 2-RTU |

| | | |
|-------------------|----------------|-------------------------------|
| ctrl_time | datetime | 任务生成时间 |
| ctrl_timeout | datetime | 任务超时时间 |
| ctrl_state | int | 任务状态 |
| ctrl_state_msg | nvarchar(50) | 任务状态说明 |
| ctrl_feedback_msg | nvarchar(2000) | 反馈信息 |
| ctrl_type | nvarchar(200) | 任务类型 |
| ctrl_data | json | 任务数据 |
| return_data | json | 返回的响应数据 |
| sync_flag | int | 同步标志, 0-不需要同步, 1-等待同步, 2-同步完成 |
| sync_time | Datetime | 下次同步时间 |
| sync_msg | nvarchar(50) | 同步信息 |

任务状态

0—等待发送
 1—已经发送
 2—已经应答
 3—任务成功
 4—任务失败
 5—任务被取消
 6—任务超时
 7—发送数据失败
 99—其他错误

4.2 远程控制接口

4.2.1 添加

请求 `post {{baseurl}}/api/Conf/CtrlData/Add`

body 请求参数样例

```
{
  "processor_name": "iotmeter",
  "ctrl_define_no": "1234567890",
  "target_comm_no": "1234567890",
  "target_devtype": 1,
  "ctrl_time": "2017-04-17",
  "ctrl_timeout": "2017-04-18",
  "ctrl_type": "iotmeter.openvalve",
  "ctrl_data": "{}"
}
```

返回

| 名称 | 说明 | 备注 |
|---------|----------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 返回新的 ctrl_id 值 | |

样例:

```
{
  "Data": 1,
  "State": "success",
  "Message": ""
}
```

4.2.2 修改

请求 `post {{baseurl}}/api/Conf/CtrlData/Update`

body 请求参数样例

```
{
  "ctrl_id":1,
  "upper_ctrl_id":0,
  "company_no":"1001",
  "processor_name":"iotmeter",
  "ctrl_define_no":"1234567890",
  "target_comm_no":"1234567890",
  "target_devtype":1,
  "ctrl_time":"2017-04-17",
  "ctrl_timeout":"2017-04-18",
  "ctrl_state":0,
  "ctrl_state_msg":"等待发送",
  "ctrl_feedback_msg": "",
  "ctrl_type":"iotmeter.openvalve",
  "ctrl_data":"{}",
  "return_data":"{}",
  "sync_flag":0,
  "sync_time":"2017-04-18",
  "sync_msg":"syncmsg"
}
```

返回

| 名称 | 说明 | 备注 |
|---------|------------------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回 ctrl_id 值, 失败返回 0 | |

样例:

```
{
  "Data": 1,
  "State": "success",
  "Message": ""
}
```

4.2.3 删除

请求 post {{baseurl}}/api/Conf/CtrlData/Delete/{ctrl_id}

样例 api/Conf/CtrlData/Delete/1

返回

| 名称 | 说明 | 备注 |
|---------|------------------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回 ctrl_id 值, 失败返回 0 | |

样例:

//成功

```
{
  "Data": 1,
  "State": "success",
  "Message": ""
}
```

//失败

```
{
  "Data": 0,
  "State": "failed",
  "Message": "找不到要删除的数据"
}
```

4.2.4 获取实体

请求: `post {{baseurl}}/api/Conf/CtrlData/GetModel/{ctrl_id}`

样例: `api/Conf/CtrlData/GetModel/2`

返回

| 名称 | 说明 | 备注 |
|---------|---------------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回实体对象, 失败返回 null | |

样例:

```
{
  "Data": {
    "ctrl_id": 2,
    "upper_ctrl_id": 0,
    "company_no": "1001",
    "processor_name": "iotmeter",
    "ctrl_define_no": "1234567890",
    "target_comm_no": "1234567890",
    "target_devtype": 1,
    "ctrl_time": "2017-04-17T10:12:10",
    "ctrl_timeout": "2017-04-18T00:00:00",
    "ctrl_state": 0,
    "ctrl_state_msg": "等待发送",
    "ctrl_feedback_msg": "",
    "ctrl_type": "iotmeter.openvalve",
    "ctrl_data": "{}",
    "return_data": "{}",
    "sync_flag": 0,
    "sync_time": "2017-04-18T00:00:00",
    "sync_msg": "syncmsg"
  },
  "State": "success",
  "Message": ""
}
```

4.2.5 获取实体列表

请求 post

{{baseurl}}/api/Conf/CtrlData/GetList?startrow={startrow}&rowcount={rowcount}

url 请求参数

| 名称 | 数据类型 | 说明 |
|----------|------|-------------|
| startrow | Int | 起始记录，从 0 开始 |
| rowcount | Int | 要返回的记录数量 |

样例 {{baseurl}}/api/Conf/CtrlData/GetList?startrow=0&rowcount=2

body 请求参数（查询条件） 样例

```
{
  "upper_ctrl_id": 0,
  "processor_name": "iotmeter",
  "ctrl_define_no": "1234567890",
  "target_comm_no": "1234567890",
  "target_devtype": 1,
  "ctrl_state": 0,
  "ctrl_type": "iotmeter.openvalve",
  "ctrl_time_from": "2017-04-17",
  "ctrl_time_to": "2017-04-18"
}
```

返回

| 名称 | 说明 | 备注 |
|---------|-----------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 返回实体对象的列表 | |

样例:

```
{
  "Data": [
    {
      "ctrl_id": 2,
      "upper_ctrl_id": 0,
      "company_no": "1001",
      "processor_name": "iotmeter",

```

```

        "ctrl_define_no": "1234567890",
        "target_comm_no": "1234567890",
        "target_devtype": 1,
        "ctrl_time": "2017-04-17T10:12:10",
        "ctrl_timeout": "2017-04-18T00:00:00",
        "ctrl_state": 0,
        "ctrl_state_msg": "等待发送",
        "ctrl_feedback_msg": "",
        "ctrl_type": "iotmeter.openvalve",
        "ctrl_data": "{}",
        "return_data": "{}",
        "sync_flag": 0,
        "sync_time": "2017-04-18T00:00:00",
        "sync_msg": "syncmsg"
    }
],
"Totalcount": 1,
"Startrow": 0,
"Rowcount": 1,
"State": "success",
"Message": ""
}

```

4.2.6 取消命令

请求 post {{baseurl}}/api/Conf/CtrlData/Cancel/{ctrl_id}

样例 {{baseurl}}/api/Conf/CtrlData/Cancel/1

返回

| 名称 | 说明 | 备注 |
|---------|------------------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回 ctrl_id 值, 失败返回 0 | |

样例:

```

//成功
{
    "Data": 1,
    "State": "success",
    "Message": ""
}

```

```
//失败
{
  "Data": 0,
  "State": "failed",
  "Message": "找不到要修改的数据"
}
```

4.2.7 重试命令

请求 `post {{baseurl}}/api/Conf/CtrlData/Retry/{ctrl_id}`

样例 `{{baseurl}}/api/Conf/CtrlData/Retry/1`

返回

| 名称 | 说明 | 备注 |
|---------|------------------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回 ctrl_id 值, 失败返回 0 | |

样例:

```
//成功
{
  "Data": 1,
  "State": "success",
  "Message": ""
}
//失败
{
  "Data": 0,
  "State": "failed",
  "Message": "找不到要修改的数据"
}
```

4.2.8 批量添加

请求 `post {{baseurl}}/api/Conf/CtrlData/AddMulti`

body 请求参数样例

```
[
    {
        "processor_name": "iotmeter",
        "ctrl_define_no": "",
        "target_comm_no": "1234567890",
        "target_devtype": 1,
        "ctrl_time": "2017-04-17",
        "ctrl_timeout": "2017-04-18",
        "ctrl_type": "iotmeter.openvalve",
        "ctrl_data": "{}"
    },
    {
        "processor_name": "iotmeter",
        "ctrl_define_no": "",
        "target_comm_no": "1234567890",
        "target_devtype": 1,
        "ctrl_time": "2017-04-17",
        "ctrl_timeout": "2017-04-18",
        "ctrl_type": "iotmeter.openvalve",
        "ctrl_data": "{}"
    }
]
```

返回

| 名称 | 说明 | 备注 |
|---------|-------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 返回新的 操作结果列表 | |

操作结果列表项

| 名称 | 说明 | 备注 |
|----------------|------------|--------------------------|
| ctrl_id | 返回 ctrl_id | >0 成功 |
| target_comm_no | 通信号 | |
| result | 操作结果 | bool 型 true-成功, false-失败 |
| message | 错误信息 | |

样例:

```
{
    "Data": [
        {
            "ctrl_id": 8,
```



```

        "target_comm_no": "1234567890",
        "result": false,
        "message": ""
    },
    {
        "ctrl_id": 9,
        "target_comm_no": "1234567890",
        "result": false,
        "message": ""
    }
],
"Totalcount": 2,
"Startrow": 0,
"Rowcount": 2,
"State": "success",
"Message": ""
}

```

4.2.9 批量修改

请求 post {{baseurl}}/api/Conf/CtrlData/UpdateMulti

body 请求参数样例

```

[
    {
        "ctrl_id": 8,
        "upper_ctrl_id": 0,
        "company_no": "1001",
        "processor_name": "iotmeter",
        "ctrl_define_no": "",
        "target_comm_no": "1234567890",
        "target_devtype": 1,
        "ctrl_time": "2017-06-12T14:59:00",
        "ctrl_timeout": "2017-04-18T00:00:00",
        "ctrl_state": 0,
        "ctrl_state_msg": "等待发送",
        "ctrl_feedback_msg": "",
        "ctrl_type": "iotmeter.openvalve",
        "ctrl_data": "{}",
        "return_data": "{}",
        "sync_flag": 0,
        "sync_time": "2017-06-12T14:59:00",
        "sync_msg": ""
    }
]

```

```

    },
    {
        "ctrl_id": 9,
        "upper_ctrl_id": 0,
        "company_no": "1001",
        "processor_name": "iotmeter",
        "ctrl_define_no": "",
        "target_comm_no": "1234567890",
        "target_devtype": 1,
        "ctrl_time": "2017-06-12T14:59:00",
        "ctrl_timeout": "2017-04-18T00:00:00",
        "ctrl_state": 0,
        "ctrl_state_msg": "等待发送",
        "ctrl_feedback_msg": "",
        "ctrl_type": "iotmeter.openvalve",
        "ctrl_data": "{}",
        "return_data": "{}",
        "sync_flag": 0,
        "sync_time": "2017-06-12T14:59:00",
        "sync_msg": ""
    }
]

```

返回

| 名称 | 说明 | 备注 |
|---------|-------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 返回新的 操作结果列表 | |

操作结果列表项

| 名称 | 说明 | 备注 |
|----------------|------------|--------------------------|
| ctrl_id | 返回 ctrl_id | >0 成功 |
| target_comm_no | 通信号 | |
| result | 操作结果 | bool 型 true-成功, false-失败 |
| message | 错误信息 | |

样例:

```

{
    "Data": [
        {
            "ctrl_id": 8,

```

```

        "target_comm_no": "1234567890",
        "result": true,
        "message": ""
    },
    {
        "ctrl_id": 9,
        "target_comm_no": "1234567890",
        "result": true,
        "message": ""
    }
],
"Totalcount": 2,
"Startrow": 0,
"Rowcount": 2,
"State": "success",
"Message": "操作完成"
}

```

4.2.10 批量删除、取消、重试

删除 post {{baseurl}}/api/Conf/CtrlData/DeleteMulti

取消 post {{baseurl}}/api/Conf/CtrlData/CancelMulti

重试 post {{baseurl}}/api/Conf/CtrlData/RetryMulti

body 请求参数样例

[8,9] //ctrl_id 列表

返回

| 名称 | 说明 | 备注 |
|---------|-------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 返回新的 操作结果列表 | |

操作结果列表项

| 名称 | 说明 | 备注 |
|----------------|------------|--------------------------|
| ctrl_id | 返回 ctrl_id | >0 成功 |
| target_comm_no | 通信号 | |
| result | 操作结果 | bool 型 true-成功, false-失败 |
| message | 错误信息 | |

样例:

```
{
  "Data": [
    {
      "ctrl_id": 8,
      "target_comm_no": "1234567890",
      "result": true,
      "message": ""
    },
    {
      "ctrl_id": 9,
      "target_comm_no": "1234567890",
      "result": true,
      "message": ""
    }
  ],
  "Totalcount": 2,
  "Startrow": 0,
  "Rowcount": 2,
  "State": "success",
  "Message": "操作完成"
}
```

4.2.11 获取命令详情

请求 post {{baseurl}}/api/Conf/CtrlData/GetDetial/{ctrl_id}

样例 {{baseurl}}/api/Conf/CtrlData/GetDetial/1

返回

| 名称 | 说明 | 备注 |
|---------|-----------------|---------------------------|
| State | 操作状态 | success - 成功, failed - 失败 |
| Message | 操作状态信息 | |
| Data | 成功返回控制命令的描述文本信息 | |

样例:

```
{
  "Data": "事物流水号:7 </br>处理程序标识:iotmeter </br>自定义命令  
编号: </br>通信编号:1234567890 </br>设备类型:RTU </br>任务类  
型:iotmeter.openvalve </br>生成时间:2017-06-12 14:59:00 </br>超时时  
间:2017-04-18 00:00:00 </br>任务状态:6 (任务超时) </br>反馈信息:  
</br>任务数据:</br> {} </br>响应数据:</br> {} </br>",
  "State": "success",
```

```
    "Message": ""
}
```

4.2.12 命令执行状态上报

本接口由应用方实现，平台调用此接口将命令执行的结果上报给应用方。

请求 POST {{baseurl}}/api/Conf/CtrlData/Report

Body 参数（数组）：

| 名称 | 数据类型 | 说明 |
|-------------------|----------------|---------|
| ctrl_id | Int | 流水号 |
| meter_comm_no | string | 通信号 |
| ctrl_time | datetime | 任务生成时间 |
| ctrl_timeout | datetime | 任务超时时间 |
| ctrl_type | string | 任务类型 |
| ctrl_state | int | 任务状态 |
| ctrl_state_msg | nvarchar(50) | 任务状态说明 |
| ctrl_feedback_msg | nvarchar(2000) | 反馈信息 |
| return_data | json | 返回的响应数据 |
| | | |

body 请求参数样例

```
[
  {
    "ctrl_id":1,
    "meter_comm_no": "1234567890",
    "ctrl_time": "2018-08-11 10:37:56",
    "ctrl_timeout": "2018-08-15 12:37:56",
    "ctrl_type": "iotmeter.openvalve",
    "ctrl_state": 3,
    "ctrl_state_msg": "任务成功",
    "ctrl_feedback_msg": "",
    "return_data": "{}"
  },
  {
    "ctrl_id":1,
    "meter_comm_no": "1234567890",
    "ctrl_time": "2018-08-11 10:37:56",
    "ctrl_timeout": "2018-08-15 12:37:56",
```

```

        "ctrl_type": "iotmeter.openvalve",
        "ctrl_state": 3,
        "ctrl_state_msg": "任务成功",
        "ctrl_feedback_msg": "",
        "return_data": "{}"
    }
]

```

返回

| 名称 | 数据类型 | 说明 |
|---------|----------|----------------------------------|
| State | string | 操作状态 success-成功, failed-失败 |
| Message | string | 操作状态信息 |
| Data | Object[] | 返回操作结果列表 |

操作结果列表项

| 名称 | 数据类型 | 备注 |
|---------------|--------|---------------------------|
| ctrl_id | int | 流水号 |
| meter_comm_no | string | 通信号 |
| result | bool | 操作结果 true-成功, false-失败 |
| message | string | 错误信息 |

样例:

```

{
    "Data": [
        {
            "ctrl_id": 8,
            "meter_comm_no": "1234567890",
            "result": false,
            "message": ""
        },
        {
            "ctrl_id": 9,
            "meter_comm_no": "1234567890",
            "result": false,
            "message": ""
        }
    ],
}

```

```

    "State": "success",
    "Message": ""
}

```

4.3 物联网表远程控制数据结构

4.3.1 数据项值

| 数据项 | 取值 |
|----------------|--------------|
| processor_name | "iotmeterv1" |
| target_devtype | 1 |

4.3.2 命令类型

| 命令类型 | ctrl_type 取值 | ctrl_data 取值 | return_data 返回 |
|------------|------------------------------------|---------------------|----------------------|
| 预付费设置价格 | "iotmeterv1.prepay.setprice" | ladder_price | "{}" |
| 预付费读取价格 | "iotmeterv1.prepay.getprice" | "{}" | ladder_price |
| 预付费设表内余额 | "iotmeterv1.prepay.setremain" | remain_param | remain_param |
| 预付费远程充值 | "iotmeterv1.prepay.remotecharge" | remote_charge_param | remote_charge_return |
| 预付费查远程充值情况 | "iotmeterv1.prepay.getchargestate" | "{}" | remote_charge_state |
| 预付费远程开户 | "iotmeterv1.prepay.openaccount" | open_account_param | open_account_return |
| 开阀 | "iotmeterv1.common.setvalveon" | "{}" | "{}" |
| 关阀 | "iotmeterv1.common.setvalveoff" | "{}" | "{}" |
| 设置累计量 | "iotmeterv1.common.setcumulant" | cumulant_param | "{}" |
| 解除阀门强制状态 | "iotmeterv1.common.releaseforce" | "{}" | "{}" |

4.5.3 数据结构(C#)

```
/// <summary>
/// 阶梯价格数据结构
/// </summary>
public class ladder_price
{
    //价格 1
    public decimal price1 { get; set; }
    //阶梯量 1
    public decimal ladder1 { get; set; }
    //价格 2
    public decimal price2 { get; set; }
    //阶梯量 2
    public decimal ladder2 { get; set; }
    //价格 3
    public decimal price3 { get; set; }
    //阶梯量 3
    public decimal ladder3 { get; set; }
    //价格 4
    public decimal price4 { get; set; }
    //阶梯量 4
    public decimal ladder4 { get; set; }
    //价格 5
    public decimal price5 { get; set; }
    //周期起始时间
    public DateTime cycle_start { get; set; }
    //周期月数
    public int cycle_month { get; set; }
}

/// <summary>
/// 表内余额
/// </summary>
public class remain_param
{
    /// <summary>
    /// 剩余金额/量
    /// </summary>
    public double remain_value { get; set; }
}

/// <summary>
```



```

/// 远程充值
/// </summary>
public class remote_charge_param
{
    //当前充值量/金额
    public double charge_value { get; set; }
    //累计充值量/金额
    public double total_value { get; set; }
    //充值次数
    public int charge_times { get; set; }
}

/// <summary>
/// 远程充值返回
/// </summary>
public class remote_charge_return
{
    //充值成功标志: 0xaa: 充值成功, 0xbb: 充值失败
    public int success { get; set; }
    //充值失败原因
    public int reason { get; set; }
    //计费方式
    public int calc_type { get; set; }
    //剩余金额/量
    public double remain { get; set; }
}

/// <summary>
/// 查充值情况
/// </summary>
public class remote_charge_state
{
    //卡充值次数
    public int card_charge_times { get; set; }
    //IC 最近一次充值金额/量
    public double card_charge_value { get; set; }
    //IC 累计充值金额/量
    public double card_total_value { get; set; }
    //远程购买次数
    public int remote_charge_times { get; set; }
    //远程最近一次充值金额/量
    public double remote_charge_value { get; set; }
    //远程累计充值金额/量
    public double remote_total_value { get; set; }
}

```

```

}

public class open_account_param
{
    //当前充值量/金额
    public double charge_value { get; set; } = 0;
    //累计充值量/金额
    public double total_value { get; set; } = 0;
    //价格 1
    public double price1 { get; set; } = 0;
    //阶梯量 1
    public UInt32 ladder1 { get; set; } = 0;
    //价格 2
    public double price2 { get; set; } = 0;
    //阶梯量 2
    public UInt32 ladder2 { get; set; } = 0;
    //价格 3
    public double price3 { get; set; } = 0;
    //阶梯量 3
    public UInt32 ladder3 { get; set; } = 0;
    //价格 4
    public double price4 { get; set; } = 0;
    //阶梯量 4
    public UInt32 ladder4 { get; set; } = 0;
    //价格 5
    public double price5 { get; set; } = 0;
    //周期起始时间
    public DateTime cycle_start_time { get; set; } = DateTime.Now;
    //周期月数
    public int cycle_month { get; set; } = 0;
}

public class open_account_return
{
    //成功标志: 0: 成功, 1 失败
    public int success { get; set; } = 0;
    //失败原因
    public int reason { get; set; } = 0;
    //失败原因描述
    public string reason_msg { get; set; } = "";
    //剩余金额/量
    public double remain { get; set; } = 0;
}

```

```
public class cumulant_param
{
    //标况累计量
    public double std_sum { get; set; }
    //工况累计量
    public double work_sum { get; set; }
}
```