拓强电气智能电表第二套接口规范 **version 2.0**

> 本接口支持对电能表/水表的实时操作，本站数据库只存储采集器和表设备信息。
>
> 不支持查询历史数据。不支持本公司App和自助充值终端。

一：接口开发原则

- 双方以 **HTTP + Json** 方式通讯\
- 电能表/水表作为物联网设备，受网络质量影响会产生掉线和高延迟的情况，所有对电能表/水表操作的接口必须以异步方式进行，异步调用需由请求方实现接收通知的API接口。不需要通过操作电能表/水表的请求可同步返回，重复请求或已经有结果的请求也会同步返回。
- API 服务器只记录采集器号和电能表/水表表号，读表,写表,开户,充值等所有操作记录和数据全部不记录。
- 为了减少或避免出现一些不可预知的异常，双方接口需实现以下特性：

  可靠性。 由于双方服务器处于不同网段，网络可能出现异常或任意一方服务器宕机会导致消息不可达，请求失败。需设计消息重发机制，比如间隔 5 秒,30 秒 ,1 分钟, 5分钟等。接口请求方在发送请求前应先持久化再请求API服务器， API服务器回复消息前也会先持久化。（不重要的消息或可以同步完成消息可以不持久化）

  幂等性。每次操作时带一个唯一操作ID ，服务器会过滤重复操作ID的请求，不携带操作ID的请求多次操作也会返回相同结果。

  高效性。支持批量操作，对同一接口的多次请求可以合并到同一次请求中。

安全性。对请求体数据进行签名，根据请求内容拼接双方约定的随机字符串计算哈希值，附加在消息内容里。

- 应用不按照本规范要求集成导致财务受损或数据丢失的，本公司概不负责。

二： 后台的使用

2.1 授权配置

系统后台 `主菜单>接口授权>授权配置` 页面

- 授权码：账号授权识别方式
- 接口模式：
  - 开发模式不验证签名
  - 生产模式签名验证，可防止通信数据被篡改，正式使用时请切换到生产模式
- 随机字符串： 待签名数据尾部拼接此随机字符串后计算哈希值
- 默认电表型号： 通过**电能表添加**接口<见 4.3.3>添加电表时如果未指定电表型号`model`字段，则默认取此处指定的型号，如果有多款型号的电表应在添加电能表接口`model`字段处指定。电表类型前的数字即为型号ID。例如此处`单相远程电表`型号ID为 `154`
- 默认水表型号：通过**水表添加**接口<见 4.3.5>添加水表时如果未指定水表型号`model`字段，则默认取此处指定的型号，如果有多款型号的水表应在添加水表接口`model`字段处指定。水表类型前的数字即为型号ID。例如此处`MBus有限水表`型号ID为 `1210`

2.2 Api请求记录

系统后台 `主菜单>接口授权>Api请求记录` 页面

此页面可查看所有异步通知Api请求记录。可查看异步任务参数，执行日志，通知成功与否，通知次数，下次通知时间，任务执行日志等详细信息。

三： 通用数据交换格式

3.1 请求格式

GET和POST方式都支持，建议查询时用GET ，有修改操作用POST

```
{
    auth_code:"abcd...",  //string 授权码 （必填）
    request_content:"请求内容",//string 格式化的json字符串（必填）
    timestamp: 1521688597 ,//int 发起请求时的服务器时间戳 （必填）
    notify_url:"www.baidu.com",//string （可带GET参数） (同步的请求不需要 ，有异步通知时需要)
    sign:"efg...",//string 签名 (必填)
}
```

- `request_content` 字段内容请使用json格式化为string类型。

- http协议请求头 `Content-Type` 支持 `application/json` 和 `application/x-www-form-urlencoded` 两种

3.2 同步请求时的返回数据格式。

```
{
    status:"SUCCESS/ERROR_TYPE", // string 请求状态 (必填)
    error_msg:"", //string 请求错误时原因描述 （仅错误时需要）
    response_content:"回复内容",//string 格式化的json字符串（必填）
    timestamp: 1521688597 ,//int API服务器时间戳 （必填）
    sign:"efg...",//string 签名 (必填)
}
```

- `status` 不是 `SUCCESS` 时表示服务不可用，比如授权失效（`AUTH_FAIL`），系统维护(`SYSTEM_ERROR`),服务器故障(`SYSTEM_ERROR`)，签名异常(`SIGN_ERROR`)，请求参数错误（`REQUEST_ERR`）等，不涉及具体业务逻辑，具体业务逻辑在 `response_content` 字段处理。此时不会对请求做任何处理，也不会持久化请求内容。
- 在Http请求返回的状态码不为 `200` 或 `status` 为 `SYSTEM_ERROR` 时应该启动重发机制，过一段时间再发。一般间隔时间设置几秒到几分钟递增，直到超过限制次数。

3.3 异步通知时的数据格式。

```
{
    response_content:"回复内容",//string 格式化的json字符串（必填）
    timestamp: 1521688597 ,//int 服务器时间戳 （必填）
    sign:"efg...",//string 签名 (必填)
}
```

- 跟 **3.2** 相比只少了2个字段。 通知接收时 只需要回复 `SUCCESS` 即可。回复 `FAIL` 或 返回的http状态码`statusCode`不是 `200` 过一段时间继续发送此通知，直到成功或达到限制次数。

3.4 `sign` 字段签名规则

对除了 `sign` 字段的数据按照字段排序，取各个字段内容拼接字符串，再加上双方约定随机字符串（网站以后会将开放API的配置功能），计算 `md5` 哈希值，调试阶段不验证签名（后台可配置）。

四：同步操作数据接口

4.1 `request_content` 通用格式。（支持批量处理）

```
[{
    ....  //其他参数....
},
{...}
,....]
```

4.2 `response_content` 字段通用格式。（支持批量处理）

```
[{
    status:"SUCCESS/FAIL",//string 状态类型枚举。正常情况下是成功，其他状态再约定 (必填)
    error_msg:"", //string 错误类型描述，只在发生错误时存在

    ....  //其他参数....
},
{...}
,....]
```

4.3 设备和档案操作接口

以下所有接口只针对 `request_content` 和 `response_content` 部分字段单条操作来讲，所有的功能支持批量操作。

4.3.1 采集器添加,删除

```
/Api_v2/collector/add
/Api_v2/collector/delete

请求:
    {
        cid: "12345678901" //string 采集器号  11/12位整数
    }
回复:
    {
        cid: "12345678901",
        status:"SUCCESS/FAIL",
        online:true/false, //bool 当前是否在线 仅状态成功时存在
    }
```

- 如果采集器号已存在，也会返回 `SUCCESS`
- 采集器数量有限制 （一般等于采购数量 + 1）， 请合理利用添加和删除操作

4.3.2 采集器档案查询(可查询已添加的采集器，以及采集器在线状态)

```
/Api_v2/collector/query

//如果请求体为空 即 "response_content" 字段内容是""或"[]" 即可查询所有采集器

请求:
    {
        cid: "12345678901" //string 采集器号  11/12位整数
    }
回复:
    {
        cid: "12345678901",
        status:"SUCCESS/FAIL",
        online:true/false, //bool 当前是否在线 仅状态成功时存在
        connect_time : "2018-01-01 00:00:00", //如果采集器在线 connect_time 参数有效 ， 表示上次连接时间
        disconnect_time: "2018-01-01 00:00:00" //如果采集器离线 disconnect_time 参数有效 ， 表示上次断线时间
    }
```

4.3.3 电能表添加,删除

```
/Api_v2/ele_meter/add
/Api_v2/ele_meter/delete

请求:
    {
        cid: "12345678901" ，//string 采集器号11/12位整数
        address:"123456789012",  //string 12/14位电能表表号（我们这边一般称作表地址）
        model:"",//string 产品型号 （可不填）
    }
回复:
    {
        status:"SUCCESS/FAIL",
        cid:"12345678901", //string 采集器号11/12位整数
        address:"123456789012" //string 12/14位电能表表号
        model:"***",//string 产品型号
    }
```

- 如果表号已存在，也会返回 `SUCCESS`
- `model` 表示产品型号，后台可配置缺省值，如果请求方服务器有多款型号的产品则必须包含此参数，不同型号电能表通讯协议不同。
- PS: **add**操作 可用于更新产品型号。
- 电能表数量有限制 （一般等于采购数量 + 1）， 请合理利用添加和删除操作

4.3.4 电能表档案查询

```
/Api_v2/ele_meter/query

//如果请求体为空 即 "response_content" 字段内容是""或"[]" 即可查询所有采集器

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
    }
回复:
    {
        status:"SUCCESS/FAIL",
        cid:"12345678901",
        address:["123456789012","123456789013"] //返回所有已经添加成功的表号
    }
```

4.3.5 水表添加,删除

```
/Api_v2/water_meter/add
/Api_v2/water_meter/delete

请求:
    {
        cid: "12345678901" , //string 采集器号11/12位整数
        address:"12345678901234",  //string 14位水表表号（我们这边一般称作表地址）
        model:"",//string 产品型号 （可不填）
    }
回复:
    {
        status:"SUCCESS/FAIL",
        cid:"12345678901", //string 采集器号11/12位整数
        address:"12345678901234" //string 14位水表表号
        model:"***",//string 产品型号
    }
```

- 如果表号已存在，也会返回 `SUCCESS`
- `model` 表示产品型号，后台可配置缺省值，如果请求方服务器有多款型号的产品则必须包含此参数，不同型号水表通讯协议不同。
- PS: **add**操作 可用于更新产品型号。
- 水表数量有限制 （一般等于采购数量 + 1） ， 请合理利用添加和删除操作

4.3.6 水表档案查询

```
/Api_v2/water_meter/query

//如果请求体为空 即 "response_content" 字段内容是""或"[]" 即可查询所有采集器

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
    }
回复:
    {
        status:"SUCCESS/FAIL",
        cid:"12345678901",
        address:["123456789012","123456789013"] //返回所有已经添加成功的表号
    }
```

4.3.7 操作状态查询 (查询异步操作的状态)

```
/Api_v2/request/status

请求:
```

```
    {
        opr_id: "" ，//string 要查询的操作ID
    }
回复:
    {
        opr_id:"" ,//string 要查询的操作ID
        status: <见 5.1.1>,
        data:[], //object[] status 为 SUCCESS 时，且有抄表数据时此字段有效。表示抄上来的数据
        err_msg:'', //string status 为失败时 ，字段有效 ， 表示失败时的错误信息
        params:{} ,//object 请求时附带的参数
        resolve_time : "",//string 完成时间。
    }
```

- `opr_id` 找不到的时候 `status` 为 `NOTFOUND`

4.3.8 取消操作 (查询异步操作)

```
POST /Api_v2/request/cancel

请求:
    {
        opr_id: "" ，//string 要取消的操作ID
    }
回复:
    {
        opr_id:"" ,//string 要查询的操作ID
        status: "SUCCESS/FAIL/NOTFOUND",
        //SUCCESS 取消成功 ， FAIL 取消失败 ， NOTFOUND 请求未找到
        opr_status:'',// 取消失败时 返回当前操作状态 <参考 5.1.1>
    }
```

- 取消操作只能用于任务状态处于 `ACEEPTED` 和 `QUEUE`状态操作 ， 其他状态不允许取消。

五：异步操作数据接口

5.1 `request_content` 通用请求格式。（支持批量处理）

```
[{
    opr_id:"123456", //string 操作ID。 （必填）
    time_out: 30 * 60 , //int 超时时间 单位是秒 （必填 -1 表示不限制时间,ps：不推荐）
    must_online:true/false, //bool 是否必须在线 （必填）
    retry_times:1 ,//int 重试次数（0-3） 服务器向采集器下发数据时没有收到返回数据，则重新下发操作，最多几次
    type:123,//int 操作类型ID

    ....  //其他参数....
},
{...}
,....]
```

- 任务执行顺序根据收到请求时间正序执行,批量处理时根据数组下标正序执行。
- `time_out` 指的是操作有效时间 （单位是秒），计时是从首次收到该操作ID请求时开始，超过时间采集器未上线则通知请求方服务器或默认失效，这个时间不会非常精确 ,考虑到网络等因素，应至少预留 5 秒时间。
- 双方服务器应该以 `opr_id` 作为基本处理单位， `opr_id` 一致时请保证只处理第一次（多次发送或接受同一 `opr_id` 时不会造成重复处理。）。 所有对电能表操作应持久化到数据库或其他存储服务，宕机或重启服务时对这些操作状态进行检查或若未发送成功应重新发送到 API 服务器。
- `must_online` 为 `true` 时 ， 如果采集器当前不在线或操作过程中采集器掉线，则立即返回操作失败。
- `must_online` 为 `false` 时 ， 如果采集器当前不在线，则在 `time_out` 时间内如果采集器上线则执行操作，操作过程中采集器掉线也会继续等待上线直到超时。
- `retry_times` 指的是单次操作无返回时重试次数，提高重试次数能提高操作成功率，但如果硬件连接有问题会导致采集器信道长时间处于无意义等待状态直到超时。

- 采集器全双工通讯模式，电能表半双工通讯模式， 单个采集器下的所有电能表同一时间只能操作一个。（不同采集器间无限制，可同时操作）

5.1.1 异步操作 `status` 状态说明：

- `ACCEPTED` 表示操作请求已接受,
- `QUEUE` 表示 `opr_id` 的操作任务处于调度状态,
- `PROCESSING`表示正在处理中,正在下发指定或等待表端返回 ,
- `SUCCESS` 表示已完成，获得了正确的返回数据或完成了操作,
- `NOTSUPPORT` 该型号表不支持此功能或该接口不支持此功能 ,\
- `FAIL` 请求失败,参数缺失等问题
- `TIMEOUT` 表示超过设定的时间没有完成处理（跟请求时的 `time_out` 参数有关）,
- `CANCELED` 该操作已被取消
- `RESPONSE_TIMEOUT` 表端返回超时（无法与表端建立通信）,
- `RESPONSE_FAIL` 表端返回异常，表不支持此功能或表拒绝处理此命令

5.2 `response_content` 字段异步通知数据格式。（支持批量处理）

```
[{
    opr_id:"123456",//string 操作ID。 （必填）
    resolve_time: 1521688597 , //int 请求被处理时的服务器时间戳 (必填)
    status:见 <5.1.1> 节,//string 状态类型枚举。
    error_msg:"", //string 错误类型描述，只在发生错误时存在

    ....  //其他参数....
},
{...}
,....]
```

- 所有异步通知全部以POST方式提交
- 服务器端会对同一个回调地址的数据尝试合并发送。每次发送通知前会等待1\~3秒，收集同一地址数据，保证了对同一回调地址请求频率低于一秒一次。数据量越大，并发越高，合并越多，回调地址调用频率会越低（代价是延迟更高，最高延迟3秒）。

5.3 `response_content` 字段同步返回数据格式。（支持批量处理）

```
[{
    opr_id:"123456",//string 操作ID。 （必填）
    status:见 <5.1.1> 节,//string 状态类型枚举。
    error_msg:"", //string 错误类型描述，只在发生错误时存在
},
{...}
,....]
```

- 同步返回字段无需扩展。对同一 `opr_id` 的请求会返回该操作状态，如果操作已经结束会返回该操作结果。

5.4 设备操作接口

以下所有接口只针对 `request_content` 和 `response_content` 部分字段单条操作来讲。字段内容必须包含 `opr_id` 。请求数据先根据 `opr_id` 持久化再处理。 所有的功能支持批量操作。

5.4.1 抄电表数据

```
POST /Api_v2/ele_read

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        type:123,// int 抄表类型ID
    }
异步通知:
    {
        status:见 <5.1.1> 节,//状态
```

```
        data:[{type:123 , value:{},dsp:""}...],//结果 (仅status为 SUCCESS 时有效)
    }
```

- `type` 指的是抄表项目枚举 例如： **当前电量，剩余金额，ABC三相电压，ABC三相电流，功率，读电价，读报警金额** 等包括但不限于这些。
- 有些型号的电能表是可以一条命令读到多个参数的，会在返回数据里全部带上,所以 `data` 字段用数组格式。
- 返回时只包含`opr_id` ,其他请求时的参数不携带， 请根据 `opr_id` 自取 。
- `dsp` 描述字段是对数据的解释，可作为调试和展示数据，格式可能变动，不要在程序里解析！

5.4.2 设置电表参数

```
POST /Api_v2/ele_write

请求:
    {
        cid: "12345678901" ,//string 采集器号11位整数
        address: "123456789012", //需要抄表的表号
        params:{}, // object 写参数
        type: 246 ,// int 操作类型ID
    }
异步通知:
    {
        status:见 <5.1.1> 节,//状态
        data:[{type:246 , value:[] }...],//结果 (仅status为 SUCCESS 时有效)
    }
```

- `type` 指的是设置类型枚举 例如： **设置电价，设置一级报警金额，设置二级报警金额，设置互感器变比** 等。
- 返回只包含`opr_id` ,其他请求时的参数， 请根据 `opr_id` 自取。
- 基本参数同上 , 有些写操作是有实际数据返回的 详见 `data` 字段，跟读结果一致。
- `value` 包含单位和数值，数值可能是多组比如 电量信息[总,尖,峰,平,谷]

5.4.3 拉合闸

```
POST /Api_v2/ele_control

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        type: 789 ,// int 操作类型ID
    }
异步通知:
    {
        status:见 <5.1.1> 节,//状态
    }
```

- `type` 只支持2种 **拉闸** 和 **合闸** 操作类型， 具体ID 见测试工具配置文件或测试工具界面。

5.4.4 电表开户，充值，清零

```
POST /Api_v2/ele_security/open_acount (电表开户)
POST /Api_v2/ele_security/recharge (电表充值)
POST /Api_v2/ele_security/reset (电表清零)

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        params:{
                count : 123 , //int 次数
```

```
                account_id:234 ,  //long 户号
                money : 100.00,//decimal 充值或首次开户金额
               }
    }
异步通知:
    {
        params:{count: 124 , account_id: 234, money:100.00 },
        status:见 <5.1.1> 节,//状态
    }
```

- 三个接口，参数是一致的（清零不需要 `params` 参数）。 `count` 参数对电表操作时非常重要的， `count` 与电能表内不一致会导致操作失败（API服务器不维护这个参数。）开户必须是1， 充值必须从上次位置 + 1 （即 `count` + 1， 从 2 开始）,如果操作成功，会返回当前表内次数 `count`， 请求方应该更新并持久化这个参数。 清零后次数从1开始 。如果充值时 `account_id` 与开户时不一致，操作会失败。
- 操作失败时可尝试 **电表清零** 操作， 重置表内 次数和开户号， 然后重新开户。（PS： 清零操作不会影响表内计量数据，但是剩余金额会重置）
- 首次使用前需执行一次清零操作，相当于初始化。
- 如果电能表因欠费自动拉闸，充值后会自动合闸，无须发合闸命令

5.4.5 抄水表数据

```
POST /Api_v2/ele_read

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        type:42,// int 抄表类型ID
    }
异步通知:
    {
        status:见 <5.1.1> 节,//状态
        data:[{type:123 , value:{} , dsp:""}...],//结果 (仅status为 SUCCESS 时有效)
    }
```

- `type` 指的是抄表项目枚举 目前固定为 `42`
- `value` 返回参数有3个分别代表 **总用量，剩余量 ，总购量**
- 返回时只包含`opr_id` ,其他请求时的参数不携带， 请根据 `opr_id` 自取 。
- `dsp` 描述字段是对数据的解释，可作为调试和展示数据，格式可能变动，不要在程序里解析！

5.4.6 水表开关阀

```
POST /Api_v2/water_control

请求:
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        type: 789 ,// int 操作类型ID
    }
异步通知:
    {
        status:见 <5.1.1> 节,//状态
    }
```

- `type` 只支持2种 **开阀** 和 **关阀** 操作类型， 具体ID 见测试工具配置文件或测试工具界面。

5.4.7 水表清零

```
POST /Api_v2/water_security/reset (水表清零)
```

```
请求：
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
    }
异步通知：
    {
        status:见 <5.1.1> 节,//状态
    }
```

- 将水表恢复出厂设置，包括已用量和剩余量全部清零，充值次数恢复为0

5.4.8 水表充值

```
POST /Api_v2/water_security/recharge (水表充值)

请求：
    {
        cid: "12345678901" ,//string 采集器号11/12位整数
        address: "123456789012", //需要抄表的表号
        params:{
                count : 123 , //int 次数
                price: 1.20 ,  //decimal 水价 2位小数
                money : 100.00,//decimal 充值或首次开户金额
               }
    }
异步通知：
    {
        params:{count: 124 , price：1.2, money:100.00 },
        status:见 <5.1.1> 节,//状态
    }
```

- 首次使用前建议执行一次水表清零操作，相当于初始化。
- 充值操作有3个额外参数，`count` 表示次数，从 **1** 开始，以后每次充值须在原值上`+1` ,如果次数不匹配会导致充值失败，`price`参数为水价，用于计算购买量，并下发到设备中，`money`为充值金额。
- 操作失败时可尝试 **水表清零** 操作， 重置表内 次数和开户号， 然后重新开户。（PS： 清零操作会将水表恢复出厂设置）
- 如果水表因欠费自动关阀，充值后会自动开阀，无须发开阀命令