珠海派诺科技股份有限公司

文件编号：	
密级：	
项目 ID：	
总页数：	

云平台数据接口

产品接入规范

版本<0.1.0>

拟制		日期	2017-12-12
审核		日期	
批准		日期	

版本	日期	AMD	修订者	说明
0.0.1	2017-12-12		易建祥	建立
0.0.2	2017-01-09		易建祥	修改，AES 的 key 更改为 EnergyYunPilot@@
0.1.0	2019-01-08		李传政	重新规范接口调用流程，区分基础接口和数据接口
0.1.1	2019-01-24		李传政	增加正式和测试请求地址内容，规范返回结果定义。

1 系统概述

该文档编写为了规范智慧云平台与第三方平台数据交互方式，支持第三方平台通过接口调用平台数据。

2 接口要求

2.1 框架要求

为降低系统间的依赖性，降低系统耦合，采用 JSON 格式传输数据。

2.2 安全要求

为了安全起见，采用授权认证的方式，验证 Token，然后上传或者下载数据。为了避免频繁请求平台，需要控制同一个用户在一段时间内（10 分钟）访问次数，默认为 30 次，超过直接返回请求过于频繁。

2.3 共同约定

编码格式采用 UTF-8 格式。

2.4 数据类型

采用 JSON 传递参数和返回结果，数据类型：

String：字符、数字，包含汉字

NString：汉字、字符、数字，utf-8 编码

Number：只能是数字

接口通用返回字段：

参数名称	类型	长度	必填	说明
status	Number		必填	接口代码返回代码，具体见附录 1；eg.0
msg	String		必填	接口代码返回描述，包含代码简短描述。
data	List		必填	接口返回数据集合

2.5 请求地址

正式接口站点名称：http://120.77.169.124:2227/Energy

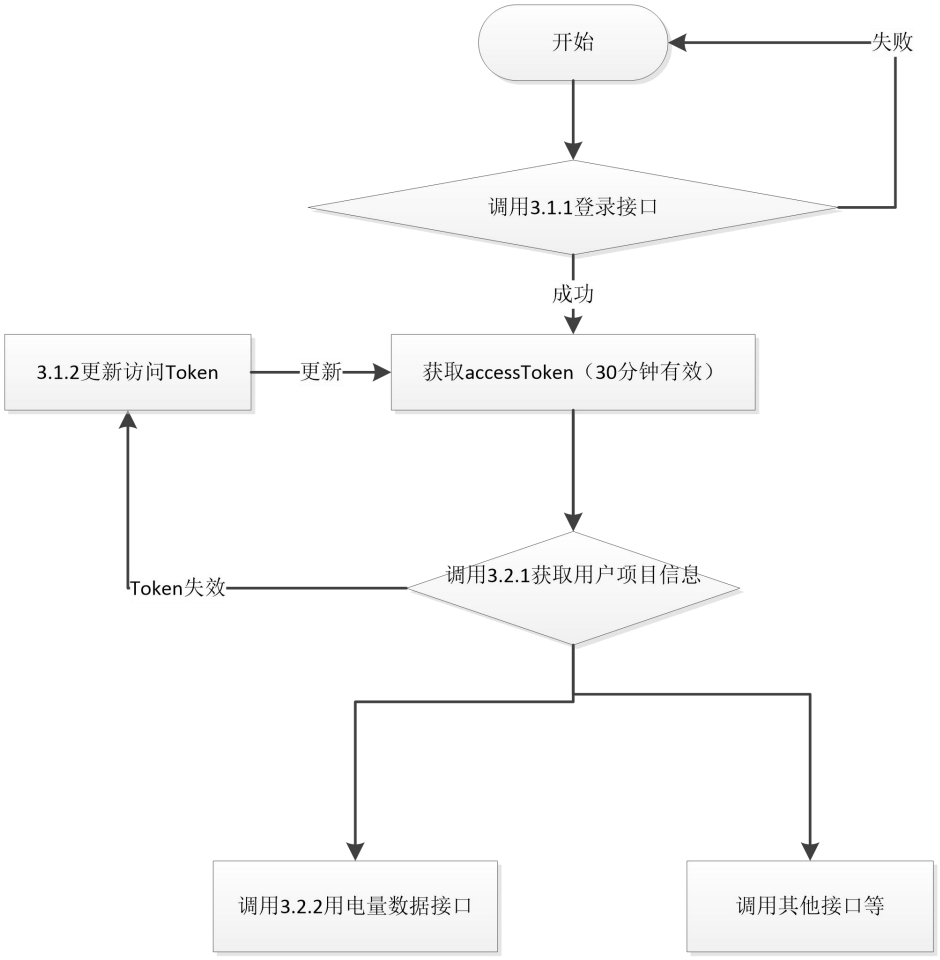
测试接口站点名称：http://125.88.36.149:7854/Energy

接口请求地址分为测试地址和正式地址，开发过程只能连接测试地址，二者请求地址只有站点名称的差别，事例如下：调用登陆接口，正式环境使用“http://120.77.169.124:2227/Energy/thirdparty/v1/UserLogin”地址请求，测试环境则替换为“http://125.88.36.149:7854/Energy/thirdparty/v1/UserLogin”，二者只有站点名称差别。

3 系统接入规范

3.1 基础接口

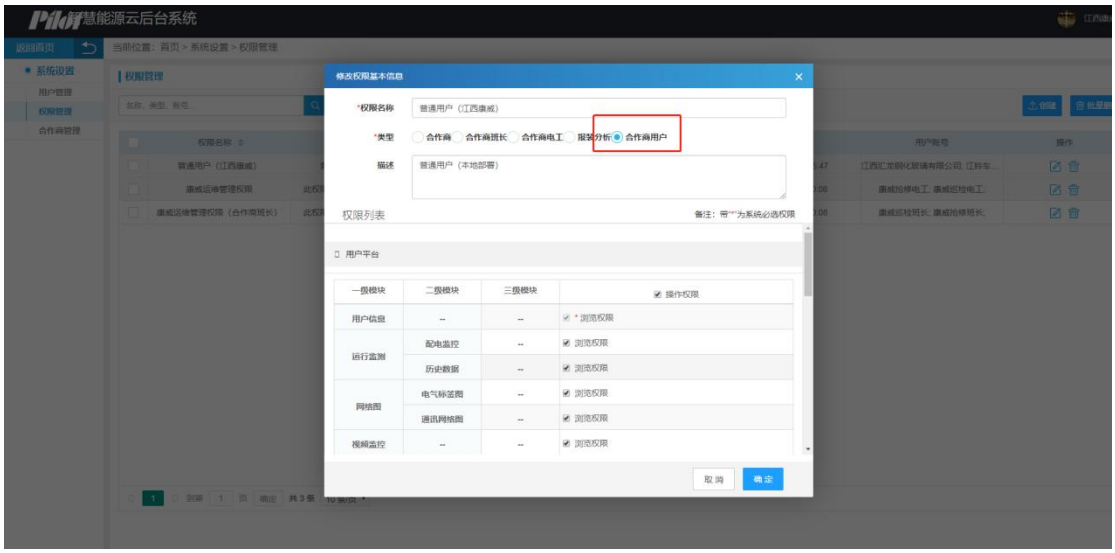
采用授权方式，提供获取、更新、销毁 Token 功能，为之后调用数据接口提供权限验证。请求流程如下：



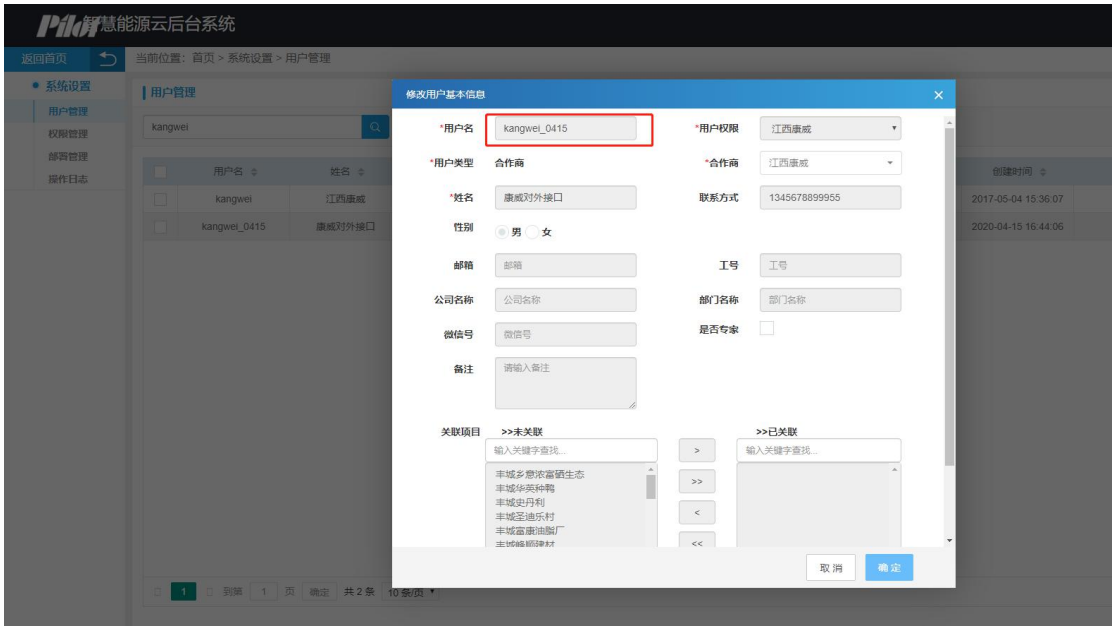
3.1.1 创建账号和密码

登录智慧云平台新后台

1. 创建用户角色



2. 创建用户账号



3.1.2 用户登录接口

3.1.2.1 接口定义

接口编号	0101	接口名称	UserLogin
接口描述	用户登录接口		
调用方式	POST		

调用说明	根据平台提供的用户名和密码，获取用户的编码、accessToken 和 refreshToken，其中密码加密 key 为固定值：“EnergyYunPilot@@”
Header 参数	Content-Type:application/json

Body 参数	{
	"userName":"test",
	"password":"123456" （可以用直接加密好的 c7Q/KM/3Zw/7Mc6zALXwPg==）
	}

站点名称	http://120.77.169.124:2227/Energy
请求地址	http://120.77.169.124:2227/Energy/thirdparty/v1/UserLogin

参数说明

参数名称	类型	长度	必填	说明
userName	String		必填	用户名，平台建立的用户名，如 testadmin
password	String		必填	对用户密码加密 AES 加密值，获取方法：将平台建立的时候设定的密码，如，12345,通过用 AES 加密所得，加密的 key 固定，加密解密的方法见附录 2。

返回数据

参数名称	类型	长度	必填	说明
userId	String		必填	用户唯一标识，如 199
accessToken	String		必填	访问 token，用于访问数据请求 API，有效期 30 分钟，过期需要重新获取。如 abcd14234
refreshToken	String		必填	更新 token，用于下次获取 accessToken 使用，有效期 1 小时。

3.1.2.2 调用举例：

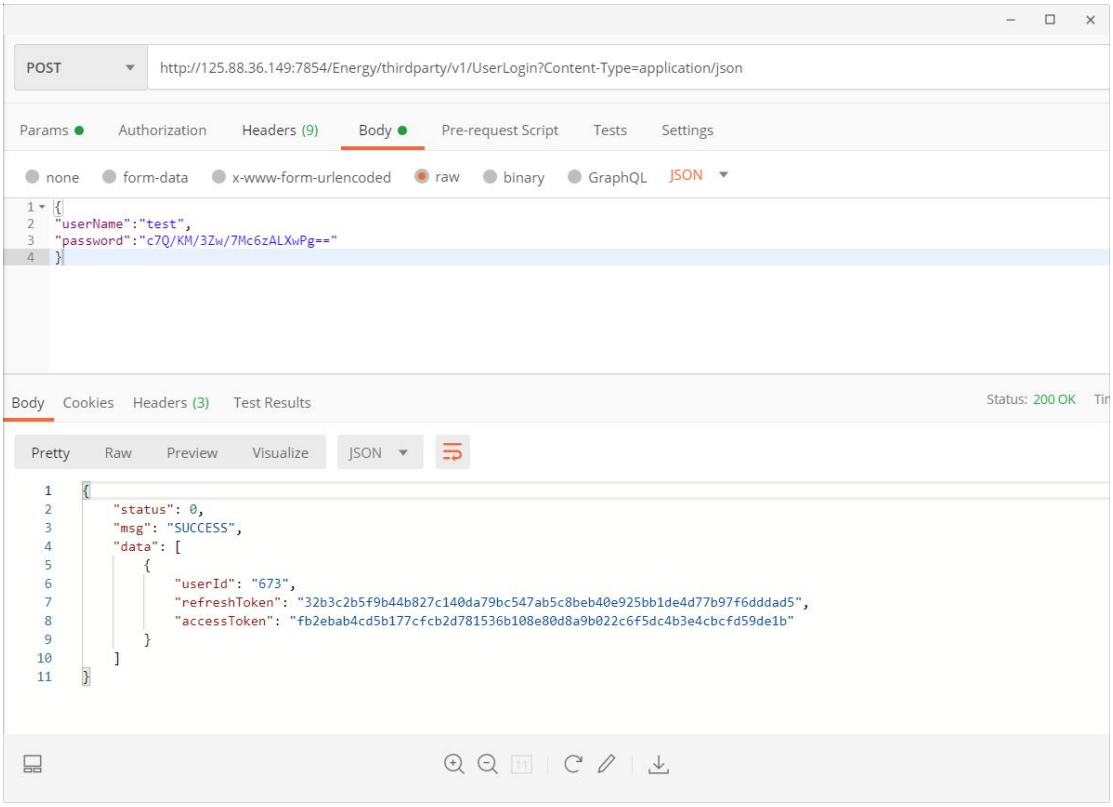
接口	**String** UserLogin**(String parmeter)**

参数值	{
	"userName":"test",
	"password":"4324xxsf234234"
	}

返回值	{ "data": [{

```
            "userId":199,
            "accessToken":"234234xsgsdgx",
            "refreshToken":"234237878sdgx"
        }
    ],
    "status": 0,  //接口调用结果代码
    "msg":"success"   //接口代码返回描述
}
```


post 参考示例，注意是 post 请求。



3.1.3 更新访问 Token

3.1.2.1 接口定义

接口编号	0102	接口名称	UpdateAccessToken
接口描述	更新访问 Token 接口		
调用方式	POST		
调用说明	通过登录接口获取的 refreshToken，获取新的 accessToken 和 refreshToken，每个 refreshToken 用完即失效。每次获取最新 accessToken 之后，原 accessToken 如还未达到 30 分钟失效时间，则默认 1 分钟之后失效。		

Header 参数	Content-Type:application/json

Body 参数	{
	"userId":199,
	"refreshToken":"234234xsgsdgx"
	}

站点名称	http://120.77.169.124:2227/Energy
请求地址	http://120.77.169.124:2227/Energy/thirdparty/v1/UpdateAccessToken

参数说明				
参数名称	类型	长度	必填	说明
userId	String		必填	平台返回的用户唯一标识，如 199
refreshToken	String		必填	平台返回的更新 token，如 abcd14234

返回数据				
参数名称	类型	长度	必填	说明
userId	String		必填	用户唯一标识，如 199
accessToken	String		必填	访问 token，用于访问数据请求 API，有效期 30 分钟，过期需要重新获取。如 abcd14234
refreshToken	String		必填	更新 token，用于下次获取 accessToken 使用，有效期 1 小时。

3.1.2.2 调用举例：

接口	**String** UpdateAccessToken**(String parmeter)**

参数值	{
	"userId":199,
	"refreshToken":"234234xsgsdgx"
	}

返回值	{
	"data": [
	{
	"userId":199,
	"accessToken":"234234xsgsdgx",
	"refreshToken":"23434244xsgsgx"
	}

	], "status": 0, //接口调用结果代码 "msg":"success" //接口代码返回描述 }

3.1.4 检查访问 Token 是否有效

3.1.3.1 接口定义

接口编号	0103	接口名称	CheckAccessToken
接口描述	检查访问 accessToken 是否有效		
调用方式	POST		
调用说明	用于检测 accessToken 是否有效		
Header 参数	Content-Type:application/json		

Body 参数	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/CheckAccessToken		

参数说明

参数名称	类型	长度	必填	说明
userId	String		必填	平台返回的用户唯一标识，如 199
accessToken	String		必填	平台返回的访问 token，如 abcd14234

返回数据

参数名称	类型	长度	必填	说明
active	Boolean		必填	是否有效，true:有效；false:无效；

3.1.4.2 调用举例：

接口	**String** CheckAccessToken**(String parmeter)**
参数值	{ "userId":199, "accessToken":"234234xsgsdgx"

	}

返回值	{
	"data": [
	{
	"active":false
	}
	],
	"status": 0, //接口调用结果代码
	"msg":"success" //接口代码返回描述
	}

3.1.5 退出登录

3.1.4.1 接口定义

接口编号	0104	接口名称	UserLogout
接口描述	退出登录，手动注销登录获取的 accessToken 和 refreshToken 信息		
调用方式	POST		
调用说明	退出登录，手动注销登录获取的 accessToken 和 refreshToken 信息		
Header 参数	Content-Type:application/json		

Body 参数	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/UserLogout		

参数说明				
参数名称	类型	长度	必填	说明
userId	String		必填	平台返回的用户唯一标识，如 199
accessToken	String		必填	平台返回的访问 token，如 abcd14234
返回数据				

参数名称	类型	长度	必填	说明
result	Boolean		必填	是否有效，true:成功；false:失败；

3.1.4.2 调用举例：

接口	**String** UserLogout**(String parmeter)**

参数值	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}

返回值	{
	"data": [
	{
	"result":true
	}
	],
	"status": 0, //接口调用结果代码
	"msg":"success" //接口代码返回描述
	}

3.2数据接口

3.2.1 获取用户站点信息

3.1.3.1 接口定义

接口编号	0201	接口名称	GetUserAllFactories
接口描述	获取用户站点数据		
调用方式	POST		
调用说明	根据接口 3.1.1 中提供的用户 id，token，获取该用户的所有站点信息，包含站点代码、站点名称		
Header 参数	Content-Type:application/json		

Body 参数	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetUserAllFactories		

参数说明				
参数名称	**类型**	**长度**	**必填**	**说明**
userId	String		必填	平台返回的用户唯一标识，如 199
accessToken	String		必填	平台返回的访问 token，如 abcd14234
返回数据				
参数名称	**类型**	**长度**	**必填**	**说明**
factoryId	String		必填	用户站点代码，如 199
factoryName	String		必填	站点名称， 如 珠海派诺科技

3.1.4.2 调用举例：

接口	**String** GetUserAllFactories**(String parmeter)**

参数值	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}

返回值	{
	"data": [
	{
	"factoryId":199,
	"factoryName":"珠海派诺科技"
	},
	{
	"factoryId":200,
	"factoryName":"珠海派诺科技 2"
	}
	],
	"status": 0, //接口调用结果代码
	"msg":"success" //接口代码返回描述
	}

3.2.2 获取项目能耗峰平谷数据

3.1.4.1 接口定义

接口编号	0202	接口名称	GetEnergyRateData
接口描述	获取项目能耗峰平谷数据		
调用方式	POST		
调用说明	获取项目能耗峰平谷数据,需要指定时间范围		
Header 参数	Content-Type:application/json		

Body 参数	{
	"userId":199,
	"accessToken":"234234xsgsdgx",
	"dataType":0,
	"factoryId":199,
	"beginTime":"2017-12-13",
	"endTime":"2017-12-14"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetEnergyRateData		

参数说明

参数名称	类型	长度	必填	说明
userId	String		必填	平台返回的用户唯一标识，如 199
accessToken	String		必填	平台返回的访问 token，如 abcd14234
dataType	Number		必填	请求数据类型，0：天。目前仅支持天类型。
factoryId	Number		必填	站点 id，如 199
beginTime	String		必填	开始时间，如 “2017-12-13”
endTime	String		必填	结束时间 ，如 “2017-12-14”,开始时间与结束时间不超过 7 天

返回数据

参数名称	类型	长度	必填	说明

deviceId	String		必填	用电设备编码，如 199
deviceName	String		必填	用电设备名称， 如 1#变压器
dataTime	String		必填	数据时间， 如 2017-12-13
total	Number		必填	总用电量， 如 8999.12
peak	Number		选填	峰用电量，如复费率未配置，则该值为空。 如 8999.12
flat	Number		选填	平用电量，如复费率未配置，则该值为空。 如 8999.12
valley	Number		选填	谷用电量，如复费率未配置，则该值为空。 如 8999.12
tip	Number		选填	尖用电量，如复费率未配置，则该值为空。 如 8999.12

3.1.4.2 调用举例：

接口	**String** GetEnergyRateData**(String parmeter)**

参数值	{
	"userId":199,
	"accessToken":"234234xsgsdgx",
	"dataType":0,
	"factoryId":199,
	"beginTime":"2019-01-13",
	"endTime":"2019-01-14"
	}

返回值	{ "data": [{ "deviceId":199, "deviceName":"珠海派诺科技", "dataTime":"2019-01-13", "total":8988.22, "peak":2288.22,

```
            "flat":4000,
            "valley":2700,
            "tip":null
        }
    ],
    "status": 0,  //接口调用结果代码
    "msg":"success"   //接口代码返回描述
}
```


3.3 网关信息请求

3.3.1 接口定义

接口编号	0301	接口名称	GetUserGateway
接口描述	用户所有的网关信息		
调用方式	POST		
调用说明	返回用户所有的网关信息		
Header 参数	Content-Type:application/json		

Body 参数	{
	"userId":199,
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetUserGateway		

参数说明				
参数名称	类型	长度	必填	说明
userId	String		必填	用户 id
accessToken	String		必填	平台返回的访问 token，如 abcd14234
返回数据				
参数名称	类型	长度	必填	说明

sn	String		必填	网关序列号，不区分大小写
name	String		必填	网关名称
factoryName	String		必填	项目名称
factoryShortName	String		必填	项目简称

3.3.2 调用举例

请求

```
{
    "sn":"xx0901230002",
    "accessToken":"234234xsgsdgx"
}
```

响应

```
{
    "status": 0,
    "msg": "success",
    "data": [{
        "sn": "xx0901230002",
        "name": "测试网关"
    },{
        "sn": "xx0901230003",
        "name": "测试网关"
    }]
}
```

3.4 配置信息请求

3.4.1 接口定义

接口编号	0401	接口名称	GetGateConfig
接口描述	获取指定网关的配置信息		
调用方式	POST		

调用说明	获取指定网关的配置信息
Header 参数	Content-Type:application/json

Body 参数	{
	"sn":199,
	"accessToken":"234234xsgsdgx"
	}

站点名称	http://120.77.169.124:2227/Energy
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetGateConfig

参数说明

参数名称	类型	长度	必填	说明
sn	String		必填	网关序列号，不区分大小写
accessToken	String		必填	平台返回的访问 token，如 abcd14234

返回数据

参数名称	类型	长度	必填	说明
sn	String		必填	网关序列号，不区分大小写
devs			必填	设备数组
devid	int		必填	设备 ID
channel	String			Xgate 通道号
devname	String		必填	设备名称
pts			必填	设备下的点位数组
ptid	int		必填	点 ID
type	int		必填	测量点类型 ID
name	String		必填	名称
pointTypeName	String			测量点类型名称
pointTypeUnit	String			测量点类型单位

3.4.2 调用举例

请求

```
{
    "sn":"xx0901230002",
"accessToken":"234234xsgsdgx"
}
```

响应

```
{
    "status": 0,
    "msg": "success",
    "data": [{
        "sn": "xx0901230002",
        "devs": [{
                "devid": 5,
                "channel": "1",
                "devname": "A2-2",
                "pts": [{
                        "ptid": 1001,
                        "type": 3,
                        "name": "CommStatus"
                    },
                    {
                        "ptid": 1002,
                        "type": 5,
                        "name": "A 相电压"
                    }
                ]
            },
            {
                "devid": 6,
                "channel": "2",
                "devname": "A5-3",
                "pts": [{
                    "ptid": 1006,
```

<table>
<tr><td></td><td>

<pre>
                "type": 7,
                "name": "有功电度"
            }]
        }
    ]
}]
}
</pre>

</td></tr>
</table>

3.5 实时信息请求

3.5.1 接口定义

接口编号	0501	接口名称	GetPointRealTimeData
接口描述	获取指定点位实时信息		
调用方式	POST		
调用说明	返回各点位最新数据。 如果某个点位的值品质无效，则填写 null		
Header 参数	Content-Type:application/json		

Body 参数	{
	"ptids": [199]
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetPointRealTimeData		

参数说明				
参数名称	类型	长度	必填	说明
ptids	List<int>		必填	点位 ID 列表，最多 10000 个点位
accessToken	String		必填	平台返回的访问 token，如 abcd14234
返回数据				
参数名称	类型	长度	必填	说明

pts			必填	点位数组
ptid	int		必填	点 ID
tm	String		必填	时间，格式：yyyy-MM-dd HH:mm:ss
value	double		必填	数值，无效时写 null

3.5.2 调用举例

请求

```
{
    "ptids":[1001,1002]
}
```

响应

```
{
    "status": 0,
    "msg": "success",
    "data": [{
        "pts": [{
                "ptid": 1001,
                "tm": "2019-01-02 13:22:32",
                "value": 5.32
            },
            {
                "ptid": 1002,
                "tm": "2019-01-03 13:22:32",
                "value": 1
            }
        ]
    }]
}
```

3.6 历史数据请求

3.6.1 接口定义

接口编号	0601	接口名称	GetPointHistoryData
接口描述	获取指定点位指定时间段的历史数据		
调用方式	POST		
调用说明	时间范围采用闭区间，需要包含结束时间的值； 查询的开始及结束时间必须在同一天； 如果某个点位某个时间点的值品质无效，则该记录不发送； 如果某个点位没有记录，则跳过该点位； **注意：** **如果请求的开始时间与结束时间相同，则每次请求允许最多 1000 个点位。** **如果请求的开始时间与结束时间不相同，则允许最多 20 个点位。**		
Header 参数	Content-Type:application/json		

Body 参数	{
	"begintime": "2019-01-02 00:00:00",
	"endtime": "2019-01-02 23:59:59",
	"ptids":[1001,1002,1003],
	"minuteinterval":15,
	"accessToken":"234234xsgsdgx"
	}		

站点名称	http://120.77.169.124:2227/Energy		
调用举例	http://120.77.169.124:2227/Energy/thirdparty/v1/GetPointHistoryData		

参数说明

参数名称	类型	长度	必填	说明
accessToken	String		必填	平台返回的访问 token，如 abcd14234
begintime	String		必填	开始时间，格式：yyyy-MM-dd HH:mm:ss
endtime	String		必填	结束时间，格式：yyyy-MM-dd HH:mm:ss
ptids	List<int>		必填	点位 ID 列表

minuteinterval	int		必填	查询分钟间隔，取值如下： 0：代表取时间范围内的所有数据； 15：代表取分钟数为 0、15、30、45 的数据； 60：代表取分钟数为 0 的数据。
返回数据				
参数名称	**类型**	**长度**	**必填**	**说明**
pts			必填	设备下的点位数组
ptid	int		必填	点 ID
tm	String		必填	时间，格式：yyyy-MM-dd HH:mm:ss
val	double		必填	数值

3.6.2 调用举例

请求

```
{
    "begintime": "2019-01-02 00:00:00",
    "endtime": "2019-01-02 23:59:59",
    "ptids":[1001,1002,1003],
    "minuteinterval":15,
"accessToken":"234234xsgsdgx"

}
```

响应

```
{
    "status": 0,
    "msg": "success",
    "data": [{
        "pts": [{
                "ptid": 1001,
                "tm": "2019-01-02 13:22:32",
                "value": 5.32
            },
            {
                "ptid": 1002,
```

```
            "tm": "2019-01-03 13:22:32",
            "value": 1
        }
    ]
```


3.7 告警数据对外接口

3.7.1 接口定义

接口编号	0701	接口名称	无
接口描述	获取项目的告警数据和恢复告警数据		
调用方式	MQTT		
调用说明	**平台给接入方提供 mqtt 的账号和订阅主题。平台产生告警（或者恢复告警）后会发送去指定主题。接入方通过订阅后，接收到告警数据（或者恢复告警）。** **注意：告警数据是使用 UTF-8 编码**		
接入信息	mqtt.host = tcp://192.168.xx.xx:1883 mqtt.username = xx mqtt.password = xx mqtt.subscribe.topic = xx		

返回数据				
参数名称	**类型**	**长度**	**必填**	**说明**
sn	String		必填	网关 sn 号
alarmId	String		必填	告警 ID
alarmLevelName	String		必填	告警等级
alarmTypeName	String		必填	告警类型（遥信变位，遥测越限，通讯异常，需量超容）
factoryId	long			项目 ID
factoryName	String			项目名称

csDeviceName	String			监测设备
csDeviceId	String			cs 设备 id
deviceId	String			设备 id
csPointName	String			监测点
csPointId	String			cs 点 id
pointId	String			点 id
ruleDesc	String			告警规则说明
unit	String			单位
status	int			0 发生告警 1 恢复告警
value	double			告警值/告警恢复值
occurTime	String			告警发生/告警恢复发生时间
createTime	String		必填	告警记录创建时间/告警恢复创建时间

3.7.2 调用举例

响应	告警： {"sn":"PE181101001920204","alarmId":"26605335875682333","alarmLevelName":"紧急告警","alarmTypeName":"遥测越限","createTime":"2020-04-07 11:36:42","csDeviceId":24413,"csDeviceName":"PMAC770_电能质量PMAC770","csPointId":601260,"csPointName":"Uab","deviceId":27005,"factoryId":1454,"factoryName":"测试0802","occurTime":"2020-03-31 14:19:33","partnerId":null,"pointId":2146977,"ruleDesc":"越上限，限值：100V","status":0,"unit":"V","value":234.49} 恢复告警： {"sn":"PE181101001920204","alarmId":"26605335875682333","alarmLevelName":"紧急告警","alarmTypeName":"遥测越限","createTime":"2020-04-07 11:36:42","csDeviceId":24413,"csDeviceName":"PMAC770_电能质量PMAC770","csPointId":601260,"csPointName":"Uab","deviceId":27005,"factoryId":1454,"factoryName":"测试0802","occurTime":"2020-03-31 14:19:56","partnerId":null,"pointId":2146977,"ruleDesc":"越上限，限值：100V","status":1,"unit":"V","value":0}

附录

1.返回结果 Status 状态定义

值	描述	备注
0	SUCCESS	请求成功
1	Json Format Error	请求参数不符合定义 Json 格式
2	Parameter Error	请求必填参数未填
3	Token Invalid	accessToken 已经失效
4	No Authority	请求范围超过允许范围
5	Server Busy	服务发生异常
6	Over-frequent Visits	10 分钟内请求次数超过规定上限
7	Other Error	不在以上情况内的异常情况
8	Not Exit User	当前用户不存在

2. AES 加密解密方法：

Java 代码：

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import sun.misc.*;

public class AES {

    // 加密
    public static String Encrypt(String sSrc, String sKey) throws Exception {
        if (sKey == null) {
            System.out.print("Key 为空 null");
            return null;
        }
        // 判断 Key 是否为 16 位
        if (sKey.length() != 16) {
            System.out.print("Key 长度不是 16 位");
            return null;
        }
        byte[] raw = sKey.getBytes("utf-8");
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");//"算法/模式/补码方
式"
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(sSrc.getBytes("utf-8"));
```

```
    BASE64Encoder encoder = new BASE64Encoder();

        return    encoder.encode(encrypted);//此处使用 BASE64 做转码功能，同时能起到 2 次加密的作用。
    }

    // 解密
    public static String Decrypt(String sSrc, String sKey) throws Exception {
        try {
            // 判断 Key 是否正确
            if (sKey == null) {
                System.out.print("Key 为空 null");
                return null;
            }
            // 判断 Key 是否为 16 位
            if (sKey.length() != 16) {
                System.out.print("Key 长度不是 16 位");
                return null;
            }
            byte[] raw = sKey.getBytes("utf-8");
            SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.DECRYPT_MODE, skeySpec);
             BASE64Decoder decoder = new BASE64Decoder();

            byte[] encrypted1 = decoder.decodeBuffer(sSrc); //new Base64().decode(sSrc);//先用 base64 解密
            try {
                byte[] original = cipher.doFinal(encrypted1);
                String originalString = new String(original,"utf-8");
                return originalString;
            } catch (Exception e) {
                System.out.println(e.toString());
                return null;
            }
        } catch (Exception ex) {
            System.out.println(ex.toString());
            return null;
        }
    }

    public static void main(String[] args) throws Exception {
        /*
```

```
         * 此处使用 AES-128-ECB 加密模式，key 需要为 16 位。
         */
        String cKey = "ae125efkk4454eef";
        // 需要加密的字串
        String cSrc = "数据 2343esdofsdfsdfw@wr[1";
        System.out.println(cSrc);
        // 加密
        String enString = AES.Encrypt(cSrc, cKey);
        System.out.println("加密后的字串是：" + enString);

        // 解密
        String DeString = AES.Decrypt(enString, cKey);
        System.out.println("解密后的字串是：" + DeString);
    }
}
```

C#代码：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace testAESEncoder
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            String result = AesEncrypt("数据 2343esdofsdfsdfw@wr[1", "ae125efkk4454eef");
            Console.WriteLine("加密前：" + result);


            Console.WriteLine("解密后：" + AesDecrypt(result, "ae125efkk4454eef"));
```

```
        }

        /// <summary>
        ///  AES 加密
        /// </summary>
        /// <param name="str"></param>
        /// <param name="key"></param>
        /// <returns></returns>
        public static string AesEncrypt(string str, string key)
        {
            if (string.IsNullOrEmpty(str)) return null;
            Byte[] toEncryptArray = Encoding.UTF8.GetBytes(str);
            RijndaelManaged rm = new RijndaelManaged
            {
                Key = Encoding.UTF8.GetBytes(key),
                Mode = CipherMode.ECB,
                Padding = PaddingMode.PKCS7
            };
            ICryptoTransform cTransform = rm.CreateEncryptor();
            Byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
toEncryptArray.Length);
            return Convert.ToBase64String(resultArray, 0, resultArray.Length);
        }

        /// <summary>
        ///  AES 解密
        /// </summary>
        /// <param name="str"></param>
        /// <param name="key"></param>
        /// <returns></returns>
        public static string AesDecrypt(string str, string key)
        {
            if (string.IsNullOrEmpty(str)) return null;
            Byte[] toEncryptArray = Convert.FromBase64String(str);
            RijndaelManaged rm = new RijndaelManaged
            {
                Key = Encoding.UTF8.GetBytes(key),
                Mode = CipherMode.ECB,
                Padding = PaddingMode.PKCS7
            };
            ICryptoTransform cTransform = rm.CreateDecryptor();
            Byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0,
toEncryptArray.Length);
            return Encoding.UTF8.GetString(resultArray);
```

}
}}

3. 点类型中文说明

点类型中文说明.x
ls