智能 PDU 接口说明文档

记录更改历史

序号	更改原因	版本	作者	更改日期	备　注
1	新增	1.0	Yixinyi	2015-02-5	新增
2	日程接口修改	2.0	Yixinyi	2015-02-20	修改
3	验证接口修改	2.1	Yixinyi	2015-04-1	修改
4	验证的补充说明	2.2	Yixinyi	2015-8-20	修改
5	增加重启、延时开关指令	2.3	Yixinyi	2016-08-16	修改
6	增加日程删除指令	2.4	Yixinyi	2016-11-17	修改
7	增加校验码参数说明	2.5	yixinyi	2016-12-09	修改
8	取实时数据增加 8 路电流	2.6	yixinyi	2017-09-25	修改
9	取实时数据增加电能	2.7	yixinyi	2017-09-25	修改
10	增加 tag、开关指令支持多口	2.8	xrm	2017-12-08	修改 2.4 新增 2.8

目录

1：引言

本文档仅提供本公司智能 PDU 接口规范，便于开发者开发自己的系统与智能 PDU 的数据通信。**未经本公司同意，任何人不得分发、修改、散布、再使用或将其用于任何商业用途。**

2：接口说明

2.1 socket 连接

首先建立 socket 连接(java 版本)
ServerSocket serverSocket = new ServerSocket(4600);//socket 服务监听端口 4600
Socket incoming = serverSocket.accept(); //等待客户端连接

2.2 验证

建立连接后插座会发送一条验证信息,如：

START login type='ES0001TVAPW' id='5d7ff303832473143227616' device='espeed138' user='admin' password='admin' check='237' END

接收到该信息后(以 START login 开头)，可进行判断插座是否有效等，再回发送：
Login Successful 给插座就可以了。
其中：以 START login 开头，以 END 结尾. Type:不能修改,默认 ES0001TVAPW；id：插座芯片的唯一 id，device：插座名称；user/password：插座内部网页的登录名及密码；check:根据附录中的 getCode 方法生成的检验码。

2.2 补充：

发送 Login Successful 信息给 PDU 后，PDU 会返回一系列信息(防止未连服务器时修改 PDU 信息,服务器在验证 socket 连接后，必须同步以下信息到数据库)，包括以下信息:

a:插口的开关状态: START iostate total='8' io8='254' check='86' END

说明：以 START iostate 开头；total 表示 PDU 的插口数(4,8,16…)；io8 表示各插口的开关二进制状态,

如 254=11111110 （二进制）1 代表开,0 代码关。

b:功率电流电压温度实时数据: START PVC_Info p='806' v='27558' c='36' ……END

说明:详见《2.3 取实时数据》

c:PDU 插口名称：START remark name1='苹果电脑' s_delay1='0' c_delay1='0'…… END

说明:详见《2.5 修改插座口名称》

d:日程信息:START programme1 p_name1='' frequency1='48' ……END

说明:详见《2.6 新增、修改日程》

e:PDU 限制信息:START PVC_setup ph='3000' pl='0' ……END

说明:详见《2.7 设置上限、警告、下限》

2.3 取实时数据

发送的指令：START PVC_get check='136' END

接收到的数据（没插温度传感器）:START PVC_Info p='9' v='23132' c='4' tflag='0' check='111' END

8 路电流 START PVC_Info p='9' v='23132' c0='4' c1='4' c2='4' c3='4' c4='4' c5='4' c6='4' c7='4' tflag='0' check='111' END

8 路电流电能 START PVC_Info p='9' v='23132' c0='4' c1='4' c2='4' c3='4' c4='4' c5='4' c6='4' c7='4' e='123' tflag='0' check='111' END

接收到的数据（有温度传感器）:START PVC_Info p='9' v='23342' c='4' tflag='1' t='202' f='683' h='356' check='137' END

其中:接收到的数据 以 START PVC_Info 开头，以 END 结尾.
p:功率,v:扩大 100 倍的电压,c:扩大 100 倍的电流,tflag:是否有温度传感器(1:有,0:无),t:扩大 10 倍的温度值,f:华氏,h：湿度。C0-c7：扩大 100 倍的 8 路电流,e:电能。
check:根据附录中的 getCode 方法生成的检验码 如上面的 111=getCode("PVC_Info p='9' v='23132' c='4' tflag='0'");
以上接收到的数据值为:功率 9w 电压:233.42v 电流:0.04A 温度:20.2C

2.4 发送开、关指令

发送：START close io='1' check='202' END
返回：START close io='1' check='202' END

其中:以 START 开始，以 END 结束，close 为关，open 为开 io 为要开/关的插座口(1-8),check 为校验码
注:不能一次全开/全关
1.3.0 版本接口修改：
参数 io:二进制模式,如要控制第三个口时，state=4 即 0000 0100 也即 2 的(3-1)次方.
即一次可控制多个插座口
云服务器 API 响应的数据与云服务下发的数据相同
内置网页的响应数据同 1.3.0 之前的版本一样，详细请看 2.8 云服务器接口更新说明

补：
重启：START handcontrol state='" + STATE + "' type='Reboot' check='" + check + "' END
延时开：START handcontrol state='" + STATE + "' type='DelayOpen' delaytime='" + delayTime + "' check='" + check + "' END
延时关：START handcontrol state='" + STATE + "' type='DelayClose' delaytime='" + delayTime + "' check='" + check + "' END
延时重启：START handcontrol state='" + STATE + "' type='DelayReboot' delaytime='" + delayTime + "' check='" + check + "' END

说明：
STATE:二进制模式,如要控制第三个口时，state=4 即 0000 0100 也即 2 的(3-1)次方.
check: 校验码
delayTime：延时时间，1-999，单位秒

2.5 修改插座口名称

指令：START remark name1='苹果电脑' s_delay1='0' c_delay1='0' r_delay1='0' action1='1' icoid1='1' name2='台式电脑' s_delay2='0' c_delay2='0' r_delay2='0' action2='1' icoid2='2' name3='笔记本电脑' s_delay3='0' c_delay3='0' r_delay3='0' action3='1' icoid3='3' name4='服务器' s_delay4='0' c_delay4='0' r_delay4='0' action4='1' icoid4='4' name5='路由器' s_delay5='0' c_delay5='0' r_delay5='0' action5='1' icoid5='5' name6='饮水机' s_delay6='0' c_delay6='0' r_delay6='0' action6='1' icoid6='6' name7='空调' s_delay7='0' c_delay7='0' r_delay7='0' action7='1' icoid7='7' name8='交换机' s_delay8='0' c_delay8='0' r_delay8='0' action8='1' icoid8='8' check='248' END

其中:以 START remark 开始 name1 为第一个插座口的名称(1-12 个字符),s_delay1 为第一个插座口的开机延时(0-999 秒),c_delay1 为第一个插座口的关机延时(0-999 秒),
r_delay1 为第一个插座口的重启延时(0-999 秒),action1 为开机动作(0:保持上次状态,1:启动,2:不启动),icoid1 为插座图标(1-34,具体图片可看网页上的)。
nameX 为第 X 个插座口名称，其他的类似.

2.6 新增、修改日程

发送指令：START programme_save flag='0' p_name='1' frequency='49' p_action='49' p_delay='' socket[0]='7' socket[1]='0' socket[2]='0' p_state='49' p_id='48' p_calendar='2015010100001' check='73' END
接收数据(因数据太长，分两段接收,前 5 个和后 5 个)：
START programme1 p_name1='12' frequency1='49' p_action1='49' p_delay1='0' socket[0]1='15' socket[1]1='0' socket[2]1='0' p_state1='49' p_id1='48' p_calendar1='2014010100001' p_name2='' frequency2='48' p_action2='48' p_delay2='0' socket[0]2='0' socket[1]2='0' socket[2]2='0' p_state2='48' p_id2='48' p_calendar2='' p_name3='' frequency3='48' p_action3='48' p_delay3='0' socket[0]3='0' socket[1]3='0' socket[2]3='0' p_state3='48' p_id3='48' p_calendar3='' p_name4='' frequency4='48' p_action4='48' p_delay4='0' socket[0]4='0' socket[1]4='0' socket[2]4='0' p_state4='48' p_id4='48' p_calendar4='' p_name5='' frequency5='48' p_action5='48' p_delay5='0' socket[0]5='0' socket[1]5='0' socket[2]5='0' p_state5='48' p_id5='48' p_calendar5='' check='85' END
START programme2 p_name6='' frequency6='48' p_action6='48' p_delay6='0' socket[0]6='0' socket[1]6='0' socket[2]6='0' p_state6='48' p_id6='48' p_calendar6='' p_name7='' frequency7='48' p_action7='48' p_delay7='0' socket[0]7='0' socket[1]7='0' socket[2]7='0' p_state7='48' p_id7='48' p_calendar7='' p_name8='' frequency8='48' p_action8='48' p_delay8='0'

socket[0]8='0' socket[1]8='0' socket[2]8='0' p_state8='48' p_id8='48' p_calendar8='' p_name9='' frequency9='48' p_action9='48' p_delay9='0' socket[0]9='0' socket[1]9='0' socket[2]9='0' p_state9='48' p_id9='48' p_calendar9='' p_name10='' frequency10='48' p_action10='48' p_delay10='0' socket[0]10='0' socket[1]10='0' socket[2]10='0' p_state10='48' p_id10='48' p_calendar10='' check='192' END

其中:
发送　以 START programme_save 开始,END 结束,flag=0:新增,flag=1:修改，p_name 为名称(1-12 字符),
frequency 为执行方式(49:一次,50:每天,51;每周,52:每月),p_action 为动作(49:开,50:关,51:重启，52：延时开,53:延时关,54:延时重启),
p_delay 为延时时间,socket[0]为要执行的插座口二进制反过来(0:没选择,1:选择了 如上面的 socket[0]='7' 7=0000 0111 所以选择的插座口为 123-----),socket[1],socket[2]为备用(16 口,24 口插座),
p_state 为是否有效(49:有效,48:无效,50:已过期),p_id 为日程 id(第几个日程,48-57),p_calendar 为时间(年月日时分星期)

接收：以 START programme1/2 开始,END 结束.
其他同上(只不过是一次 10 条日程信息)

补：日程删除
START programme_delete pidnum='3' id1='0', id2='1', id3='3' check='XXX' END
pidnum 为要删除的日程数量(支持批量删除)，idx 为顺序数,等号后面为日程 id(从 0 开始)。
Check 为校验码。

2.7 设置上限、警告、下限

发送:
电流、压、功率:START PVC_setup type='C' ch='1200' cl='0' cw='1000' cport='135' cstate='6' csec='0' check='156' END
温度:START PVC_setup type='T' trd='1' tset='120' th='50' tsocket='145' check='140' END

接收：START PVC_setup ph='3000' pl='0' pw='2200' pport='0' pstate='6' psec='0' vh='24000' vl='20000' vw='23000' vport='0' vstate='6' vsec='0' ch='1200' cl='0' cw='1000' cport='0' cstate='6' csec='0' tflag='1' t='202' trd='0' tset='0' th='0' tsocket='0' f='683' h='313' check='203' END

其中：
发送：以 START PVC_setup 开始,以 END 结束,type 为类型(C:电流,V:电压,P:功率,T:温度),
ch 为上限,cl 为下限,cw 为警告(注：ch,cl,cw 为电流,电压为 vh,vl,vw,功率为 ph,pl,pw,电流扩大了 100 倍,电压扩大了 10 倍,功率没变),
cport 为超出上限时要执行的插座口(如:cport='135' 135=1000 0111,所以选择的插座口为 123----8),
cstate 为执行的动作(0:关闭,1:开启,2:重启,3:延时关闭,4:延时开启,5:延时重启,6:不动作),

csec 为延时时间.
温度：trd 为温度模式(0：关闭,1:升温,2:降温),tset 为设置的温度(扩大了 10 倍),th 为回差值(扩大了 10 倍),tsocket 为选择的插座口(如 tsocket='145' 145=10010001,选择的插座口为 1---5--8).

接收：同上(只不过一次把电流、电压、功率、温度全发送)

2.8 云服务器接口更新说明（1.3.0 及以后版本）

API 数据通讯中增加 tag 参数，该参数为 12 位数字，由服务器指定任意数字，API 响应指令中也返回相应的 tag 参数。
例如发送 START close tag=’012345678910’ io='1' check=’….’ END
PDU 响应的数据同样为 START close tag=’012345678910’ io='1' check=’….’ END
通过 tag 可判断响应的是先前的哪一条指令
除了响应“Login Successful”成功登陆指令，其它指令同上支持 tag

内置网页操作触发的数据上传仍保持之前的格式，不带 tag 参数

3：附录

3.1 注意事项

A:以上所有指令的发送与接收都是按”行”处理的，即指令最后都有\n，即 java 中的 readline(),println();
B:PDU 会每隔 10 秒发送一个单字节”S”到服务器，来保持和服务器的 socket 长连接。

3.2 生成校验码方法

```
/**
 * 生成校验码(不计算前后空格)
 * @param str：
 * @return
 */
public static int getCode(String str) {
	byte chars[] = str.getBytes();
	int sum = 0;
	for (int i = 0; i < chars.length; i++) {
		sum = sum + chars[i];
	}
	sum = sum & 0xff;
	return sum;
}
```

```
/**
测试校验
**/
public static void main(String args[]){
    String str = "START login type='ES0001TVAPW'
id='5d7ff303832473143227616' device='espeed138' user='admin'
password='admin' check='237' END";
    String str2= str.substring(6,str.indexOf(" check="));//从login开始
一直到password='admin'结束
    System.out.println("校验码:"+getCode(str2));
    }

//如以下图片中红框内字符串
```

START login type='ES0001TVAPW' id='5d7ff303832473143227616' device='espeed138 user='admin' password='admin' check='237' END