API系列PDU

MQTT功能及开发手册

# 概述

API智能PDU产品支持通过MQTT接入物联网。IOT 用户端APP或云服务器可以查询，配置及控制设备。

测试或二次开发MQTT功能可以使用开源项目Eclipse Mosquitto来构建完整IOT系统,其架构如下图：

MQTT云服务器

浏览器

JS

**PDU**

**PDU**

测试PC机

Linux

手机

APP

其它设备或软件

# 云服务搭建

PDU MQTT功能需要一个MQTT云服务器。在ubuntu下，可以直接通过apt install mosquitto来安装MQTT服务器；

如果无法安装，可以偿试从源码编译安装。

mosquitto开源官网是http://mosquitto.org； <https://github.com/eclipse/mosquitto>给出来git下载mosquitto源码的源； 即https://github.com/eclipse/mosquitto.git;可使用git获取mosquitto源码：git clone <https://github.com/eclipse/mosquitto.git>

在云服务器安装完成后，可以在linux主机上测试MQTT服务是否已经安装成功，并正确配置。

# 测试云服务器

在安装有ubuntu系统的PC机上，安装mosquitto客户端，即mosquitto\_sub和mosquitto\_pub两个命令行程序；

mosquitto\_sub执行MQTT主题订阅，mosquitto\_pub执行MQTT主题发布； 简单说，mosquitto\_sub接收来自MQTT服务器的带有主题的消息；mosquitto\_pub向MQTT服务器推送消息。

### 测试第一步

打开一个终端，输入：

$ mosquitto\_sub -h [www.cloudwsn.com](http://www.cloudwsn.com/) -p 5132 -t “#”

回车后，mosquitto\_sub等待MQTT消息。

1. 参数： -h [www.cloudwsn.com](http://www.cloudwsn.com/) 指定MQTT服务器的地址；www.cloudwsn.com是我们的云服务器，测试自行搭建的MQTT服务器时，用该服务器的域名或IP替换这个域名。
2. 参数：-p 5132是MQTT服务器使用的端口号，我们的MQTT服务使用了5132端口。在mosquitto安装后，默认的服务端口号是1883;如果你的mosquitto服务使用了其它端口号，请在mosquitto\_sub和mosquitto\_pub命令行上用-p指定。
3. 参数：-t “#”,指出mosquitto\_sub要订阅的主题； #是一个通配符，可以接收任何主题；

### 测试第二步

再打开一个终端，输入：

$mosquitto\_pub -h [www.cloudwsn.com](http://www.cloudwsn.com/) -p 5132 -t “test” -m “hello”

回车后，mosquitto\_pub向MQTT服务器推送带有 test主题的消息hello. 这时，在上一步打开的终端中，查看mosquitto\_sub是否有收到hello消息。如果有收到，则MQTT云服务器搭建成功。如果失败，请查检云服务器是否没有开放mosquitto的服务端口号，默认为1883

# PDU的MQTT主题

MQTT主题是一个字符串，类似于Unix/Linux系统的文件路径，形如：

/topic1/topic11/topic111

PDU定义的MQTT主题如下：

/product\_id/device\_class/custom\_id/device\_id/{in/out}/datapoint

比如

/yespeed/pdu/cloudwsn/16023102389045643038072/in/1000000

/yespeed/pdu/cloudwsn/16023102389045643038072/out/100010

其中：

* product\_id – 指示产品标识，PDU固件中取固定值yespeed
* device\_class – 指示产品类型，PDU固件中取固定值pdu
* custom\_id – 指示产品应用的客户标识，固件中取固定值yespeed
* device\_id – 指示该设备的标识，每个设备标识唯一，PDU中取值由mac地址计算得出；可以web配置的information/system页面中查看，即device label
* in/out – 仅用于区分消息方向
* datapoint – 指示查询，配置的功能点,表达形式如同一个整数，PDU当前定义的datapoint有：

|  |  |
| --- | --- |
| **Datapoint** | **说明** |
| 1000000 | 查询设备组 |
| 1000001 | 设备组控制 |
| 1000002 | 设备组配置 |
| 1000100 | 插座设备查询 |
| 1000101 | 插座设备控制 |
| 1000102 | 插座设备配置 |
| 1000200 | 查看空调设备 |
| 1000201 | 控制空调设备 |
| 1000202 | 配置空调设备 |
| 1001000 | 查看触发器 |
| 1001001 | 配置触发器 |
| 1010000 | 查看操作日志 |
| 1010010 | 查看报警日志 |

# PDU MQTT功能测试

|  |  |
| --- | --- |
| **服务器** | [www.cloudwsn.com](http://www.cloudwsn.com/)， 服务端口5132 |
| **PDU设备** | 16003227681162731644419 |
| **订阅** | mosquitto\_sub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/out/#" |
| **发布** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/ |

# 设备组

### 设备组信息查询

|  |  |
| --- | --- |
| **datapoint** | 1000000 |
| **说明** | 查询设备组及设备组内设备信息 |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000000" -m "{}"  参数：  json数据体；可以是’{}’,或者’{“devid”:1}’  其中，devid参数指明要查询哪个设备组，以整数型id值表达，取值1－4； |
| **返回** | "devices":[{"id":1,"type":1,"slave":1,"name":"dev-1","voltage":"220.0","tcurrent":"36.0","power":7920,"freq":"50.0","energy":"655.4","subdevs":[{"id":1,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S1","who":"","act":"","tim":"","det":"","current":"1.0"},{"id":2,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S2","who":"","act":"","tim":"","det":"","current":"2.0"},{"id":3,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S3","who":"","act":"","tim":"","det":"","current":"3.0"},{"id":4,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S4","who":"","act":"","tim":"","det":"","current":"4.0"},{"id":5,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S5","who":"","act":"","tim":"","det":"","current":"5.0"},{"id":6,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S6","who":"","act":"","tim":"","det":"","current":"6.0"},{"id":7,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S7","who":"","act":"","tim":"","det":"","current":"7.0"},{"id":8,"type":1,"on":0,"rintv":1,"dintv":5,"name":"S8","who":"","act":"","tim":"","det":"","current":"8.0"}]},  {"id":4,"type":2,"slave":100,"name":"misc-1","online":1,"temperature":"27.8","humidity":"55.6","subdevs":[{"id":1,"type":2,"on":0,"rintv":1,"dintv":5,"name":"Air Conditioner","who":"","act":"","tim":"","det":"","factory":"media","typeidx":5,"code":45,"mode":1,"wind":0,"fan":0,"temp":28}]}]  (上述返回数据，为了文档可读性，去掉了一些设备组数据)  返回数据为Json数据体，由devices数组组成。每个数组元素代表一个设备组；  每个设备组数据包括：   1. id － 设备组标识号，整数型 2. type－设备组类型，整数型；1为插座类设备组，2为空调类设备组 3. name － 设备组名称，字符型 4. voltage －设备组的当前电压，字符型 5. tcurrent－设备组的当前总电流，字符型 6. power － 设备组当前功率，整数型 7. freq: － 设备组当前频率，字符型 8. energy － 设备组当前电量，字符型 9. temperature － 设备组当前环境温度，字符型，只用于传感器类设备组 10. humidity － 设备组当前相对湿度，字符型，只用于传感器类设备组 11. subdevs － 设备组下的子设备，数组类型。 每个数组元素代表一个子设备，每个子设备数据包括： 12. id － 设备标识，整数型 13. type－设备类型，整数型；1代表插座设备 14. on － 设备的开关状态；整数型0 ， 关闭状态；1,打开状态 15. rintv － 重启间隔，整数型；重启动作是先关，再开，该值为从关到开间隔时间 16. dintv －延时动作时间，整数型；用于延时控制 17. name － 设备名称，字符型 18. who － 最后一次操作，来源于哪个接口，字符型 19. act － 最后一次操作的动作，字符型 20. tim － 最后一次操作的时间，字符型； 21. det － 最后一次操作的描述 22. current － 设备的当前电流，字符型。仅用于插座 23. factory － 设备厂商，字符型。仅用于空调 24. typeidx －设备类型内部索引，整数型；仅用于空调 25. mode － 设备当前模式，整数型，仅用于空调(0自动；1制冷；2抽湿；3摆风；4制热） 26. wind － 设备当前风速，整数型，仅用于空调（0自动；1低风；2中风；3高风） 27. fan － 设备当前风向，整数型，仅用于空调（0自动，1手动） 28. temp－设备当前温度，整型，仅用于空调 |

### 设备组控制

|  |  |
| --- | --- |
| **datapoint** | 1000001 |
| **说明** | 向设备组发送动作指令 （示例打开第一设备组内的所有插座） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000001" -m '{"devid":1,"actid":3}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值1－4； 2. actid，动作标识，整数型，取值（1无动作，2关闭，3打开，4重启，5延时关闭，6延时打开，7延时重启，8报警） |
| **返回** | 参数devid指定的设备组查询信息，详见“设备组查询"返回 |

### 设备组配置

|  |  |
| --- | --- |
| **datapoint** | 1000002 |
| **说明** | 向设备组发送配置指令 （示例设置第一设备组的名称为group1） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000002" -m '{"devid":1,"name":”group1”}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值1－4； 2. name，设备组名称，字符型 |
| **返回** | 参数devid指定的设备组查询信息，详见“设备组查询"返回 |

# 插座设备

### 插座设备查询

|  |  |
| --- | --- |
| **datapoint** | 1000100 |
| **说明** | 查询指定插座的信息 （示例查询第一设备组内的第二个插座的信息） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000100" -m '{"devid":1,"linid":2}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值1－4； 2. linid，插座在设备组中的索引标识，整数型1－8 |
| **返回** | 参见“设备组查询”返回值 |

### 插座设备控制

|  |  |
| --- | --- |
| **datapoint** | 1000101 |
| **说明** | 向插座发送控制指令 （示例打开第一设备组内第二个插座） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000101" -m '{"devid":1,"linid":2,"actid":3}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值1－4； 2. linid, 指明哪个插座设备，整数型，取值1－8； 3. actid, 指明动作标识，整数型，取值“见前面关于动作标识的描述” |
| **返回** | 参见“插座查询”返回 |

### 插座设备配置

|  |  |
| --- | --- |
| **datapoint** | 1000102 |
| **说明** | 向插座发送配置指令 （示例设备第一设备组内第二个插座的名称为so2） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000102" -m '{"devid":1,"linid":2,"name":"so2"}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值1－4； 2. linid, 指明哪个插座设备，整数型，取值1－8； 3. name, 指定设备名称，字符型 4. reset\_interval，指定设备重启间隔，整数型，单位秒 5. delay\_interval，指定设备延时动作时长，整数型，单位秒；   name,reset\_interval,delay\_interval可以单独给定，也可以同时给出； |
| **返回** | 参见“插座查询”返回 |

# 空调设备

### 空调设备查询

|  |  |
| --- | --- |
| **datapoint** | 1000200 |
| **说明** | 查询空调设备信息 （示例查询空调设备信息） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000200" -m '{"devid":4,"linid":1}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值4； 2. linid，空调在设备组中的索引标识，整数型1 |
| **返回** | 参见“设备组查询”返回值 |

### 空调设备控制

|  |  |
| --- | --- |
| **datapoint** | 1000201 |
| **说明** | 空调设备控制 （示例关空调） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000201" -m '{"devid":4,"linid":1,"command":"switch","action":"off"}  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值4； 2. linid，空调在设备组中的索引标识，整数型1； 3. command，控制空调命令，详见下表 4. action，控制空调动作  |  |  |  | | --- | --- | --- | |  | **Command** | **action** | | 开关 | switch | on,off,reset,”delay off”,” delay on”,”delay reset” | | 模式 | mode | auto，cool，dehum,fan,hot | | 温度 | temperature | １６-３１ | | 风速 | wind | auto,low,medium,high | | 风向 | fan | auto,manual | |
| **返回** | 参见“空调设备查询”返回值 |

### 空调设备配置

|  |  |
| --- | --- |
| **datapoint** | 1000202 |
| **说明** | 配置空调设备 （示例配置空调设备开机自动开启） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1000202" -m '{"devid":4,"linid":1, "boot":1}'  参数：  json数据体，其中：   1. devid，指明哪个设备组，整数型，取值4；必须指定； 2. linid，空调在设备组中的索引标识，整数型1；必须指定 3. name，设置空调名称，字符型 4. reset\_interval,设置空调重启间隔时间（即关到开的时间），整数型，单位秒 5. delay\_interval，设置空调延时动作的延时时长，整数型，单位秒 6. boot，设置在设备开机时，是否开启空调；整数型，0不开启，1开启，2依据上次动作； 7. ac\_type，设置空调类型，字符型；详细见后面”空调类型描述” 8. ac\_mode，设置空调预置模式，字符型。取值参见“空调控制”相关部分 9. ac\_wind，设置空调预置风速，字符型。取值参见“空调控制”相关部分 10. ac\_fan，设置空调预置风向，字符型。取值参见“空调控制”相关部分 11. ac\_temperature，设置空调预置温度，字符型。取值参见“空调控制”相关部分 |
| **返回** | 参见“空调设备查询”返回值 |
| **空调类型描述** | 空调类型由“厂家－型号索引”组成，如“media-5”  型号索引是PDU支持的空调型号的一个内部编号数字，与具体空调型号没有直接关系。该数字的确定，可以通过PDU的web配置界面中的空调自动检测获得。也可以从1开始按个偿试；  当前,ＰＤＵ支持的厂家有：  gree，haier，media，changhong，chigo，huabao，kelon，TCL，galanz，hualing，chunlan，AUX |

# 触发器

### 触发器查询

|  |  |
| --- | --- |
| **datapoint** | 1001000 |
| **说明** | 查询触发器信息 （示例查询电压触发器信息） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1001000" -m '{"where":"voltage"}'  参数：  json数据体，其中：   1. where，指明触发器类型，字符型，取值见下表 2. devid, 指明在哪个设备上的触发器，整数型。仅用于电压，电流和电量触发器 3. id,指明触发器的标识。仅用于温度，定时器和网络ping触发器  |  |  | | --- | --- | | **Where** | **说明** | | voltage | 当某个设备组电压达到预定条件时触发 | | current | 当某个设备组总电流达到预定条件时触发 | | energy | 当某个设备组电量达到预定条件时触发 | | temperature | 当环境温度达到预定条件时触发 | | timer | 当定时器条件到达时触发 | | ping | 当ping指定网络主机通或不通时触发 | |
| **返回** | { "voltage": [ { "section": "lower1v", "name": "lower", "device": 1, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "upper1v", "name": "upper", "device": 1, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "near1v", "name": "alert", "device": 1, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "lower2v", "name": "lower", "device": 2, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "upper2v", "name": "upper", "device": 2, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "near2v", "name": "alert", "device": 2, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "lower3v", "name": "lower", "device": 3, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "upper3v", "name": "upper", "device": 3, "conditon": 0, "threshold": "", "action": 0, "params": "" }, { "section": "near3v", "name": "alert", "device": 3, "conditon": 0, "threshold": "", "action": 0, "params": "" } ] }  这是电压触发器数据体，其它触发器有相同格式的数据体；其中：   1. section － 作为触发器记录的唯一标识，用于识别触发器，字符型 2. name －触发器名称，字符型 3. device － 触发器属于哪个设备组，整数型，取值1－4 4. conditon － 触发条件，整数型。取值详见下表 5. threshold － 触发阀值，字符型，取值详见下表 6. action － 触发动作，整数型，取值详见以上相关“actid”描述部分 7. params － 触发器参数，用于指定触发动作作用于哪个设备上。  |  |  |  |  | | --- | --- | --- | --- | | **conditon** | **threshold** | **说明** | **适用于** | | 0 - disable |  | 禁用该触发器 | 所有触发器 | | 257 – lower | 整数，或浮点数 | 小于阀值时触发 | 电压，电流，电量及温度 | | 258 – upper | 大于阀值时触发 | | 259 – near | 接近于阀值时触发 | | 513 – oneshot | 格式：Hh:mm:ss, d,M  如：“08：22：30，24，11”表示11月24日8点22分30秒 | 只在该时间点触发 | 定时 | | 514 – everyday | 格式：hh:mm:ss | 每天 | 定时 | | 515 – everyweek | 格式：hh:mm:ss, w  如：”09:30:00，3”星期三9点30分00秒  星期日用7表示 | 每周 | 定时 | | 516 – everymonth | 格式：hh:mm:ss,d | 每月 | 定时 | | 1025 – netloss | IPv4地址  如192.168.0.123 | 网络ping不通时触发 | ping | | 1026 – netavailable | IPv4地址 | 网络能ping通时触发 | ping | |

### 触发器配置

|  |  |
| --- | --- |
| **datapoint** | 1001001 |
| **说明** | 配置触发器 （示例电流大于10A时关闭第一设备组第3个插座） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1001001" -m '{"section":"upper1c","conid":258,"threshold":"10","actid":2,"params":"3"}'  参数：  json数据体，其中:   1. section – 触发器记录的唯一标识，字符型 2. conid － 触发条件，整数型，参见前面关于conid的描述 3. threshold － 阀值，字符型 4. actid － 动作标识，整数型；参见前面关于actid的描述； 5. params － 触发参数，字符型；通常指明触发动作，作用于哪些设备；对于电压，电流，电量触发器，因为它们会隶属于某个设备组，所以params参数只要指明子设备id；而对于温度，定时器，ping触发器，它们不专属某个设备组所有，需要在params中指定哪个设备组。所以，此类触发器的params格式为”devid+subdevid”，比如“11”代表第一设备组，第一个插座，“23”代表第二设备组第三个插座；params可以一次给出多个设备，用“，”分隔；如“11，23，15” |
| **返回** | 参见“触发器查询”返回值 |

# 日志

### 查看操作日志

|  |  |
| --- | --- |
| **datapoint** | 1010000 |
| **说明** | 查看操作日志 (示例查看全部操作日志） |
| **示例** | mosquitto\_pub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/in/1010000" -m '{"where":0}  参数：  json数据体，其中:   1. where – 查看日志类型，整数值，0操作日志，1报警记录 2. sdate – 查询起始日期，字符型，如2019-10-28 3. edate – 查询终止日期，字符型，如2019-10-01 4. pidx – 结果分页索引，即第几页，整数值 5. psize – 每页记录数，整数值 |
| **返回** | { "log": [ { "time": "2019-10-30 12:42:46", "msg": "1:dev-1 3:S3 OFF Internal Trigger current (> 10.0A)\n" }, { "time": "2019-10-30 12:42:36", "msg": "device connect to local network\n" }, { "time": "2019-10-30 12:42:33", "msg": "network cable has connected\n" }, { "time": "2019-10-30 12:42:30", "msg": "network cable is missing\n" }, { "time": "2019-10-30 12:42:29", "msg": "network cable has connected\n" }, { "time": "2019-10-30 12:42:29", "msg": "network cable is missing\n" }, { "time": "2019-10-30 12:42:29", "msg": "system bootup\n" } ] }  每条日志记录有两部分，即：  time － 日志产生时间，字符型  msg － 日志信息，字符型 |

# 消息推送

PDU的MQTT功能支持消息推送。即PDU主动向服务器和客户端推送信息，无需客户端发送请求指令。这有助具备实时接收MQTT消息的客户端，实时接收PDU的消息或事件。

当前，PDU支持推送的消息为控制类消息，在设备开关操作上，PDU会主动向MQTT客户端发送设备状态信息。

测试方法：

运行下面订阅命令，等待消息。然后，在web页面，或telnet等终端，执行开关动作，观察订阅输出。mosquitto\_sub -h www.cloudwsn.com -p 5132 -t "/yespeed/pdu/yespeed/16003227681162731644419/out/#"

# 关于客户端

PDU的MQTT功能需要相应的客户端应用来支撑；这里说的客户端应用，可能包括如下几种：

1. 运行于云服务器端的数据收集，或是自动控制服务；
2. 基于云服务器WEB服务的，PDU集群管理系统。
3. 基于移动用户的APP软件或控制面板
4. 可能需要接入PDU管理的其它嵌入式应用设备

A，B两种应用的开发思路是，使用不同编程语言的mosquitto库，实现与PDU通讯；运行于服务器较的C/Ｃ++,ＰＨＰ,Ｊａｖａ或go语言开发的应用，调用库接口来获取，控制或配置远程PDU；

在PDU的固件源码中，我们提供了一个webside的代码，它使用websocket直接在mosquitto服务与web页面之间建立通讯。web页面使用JavaScript执行websocket接口；服务器端，mosquitto直接配置websocket支持。

C应用的开发思路是，在android里，调用MQTT库，直接与云端的mosquitto服务通讯。

D应用的开发思路是，在嵌入式设备里，调用MQTT库（多为C语言实现），直接与云端的mosquitto服务通讯。

# 关于安全

mosquitto支持安全通讯。但是，我们在PDU端还没有启动密钥，认证等管理。这部分，还需要研究；现在不影响MQTT的功能应用。