机智云平台标准接入协议

之通用数据点协议

(v4.0.2)

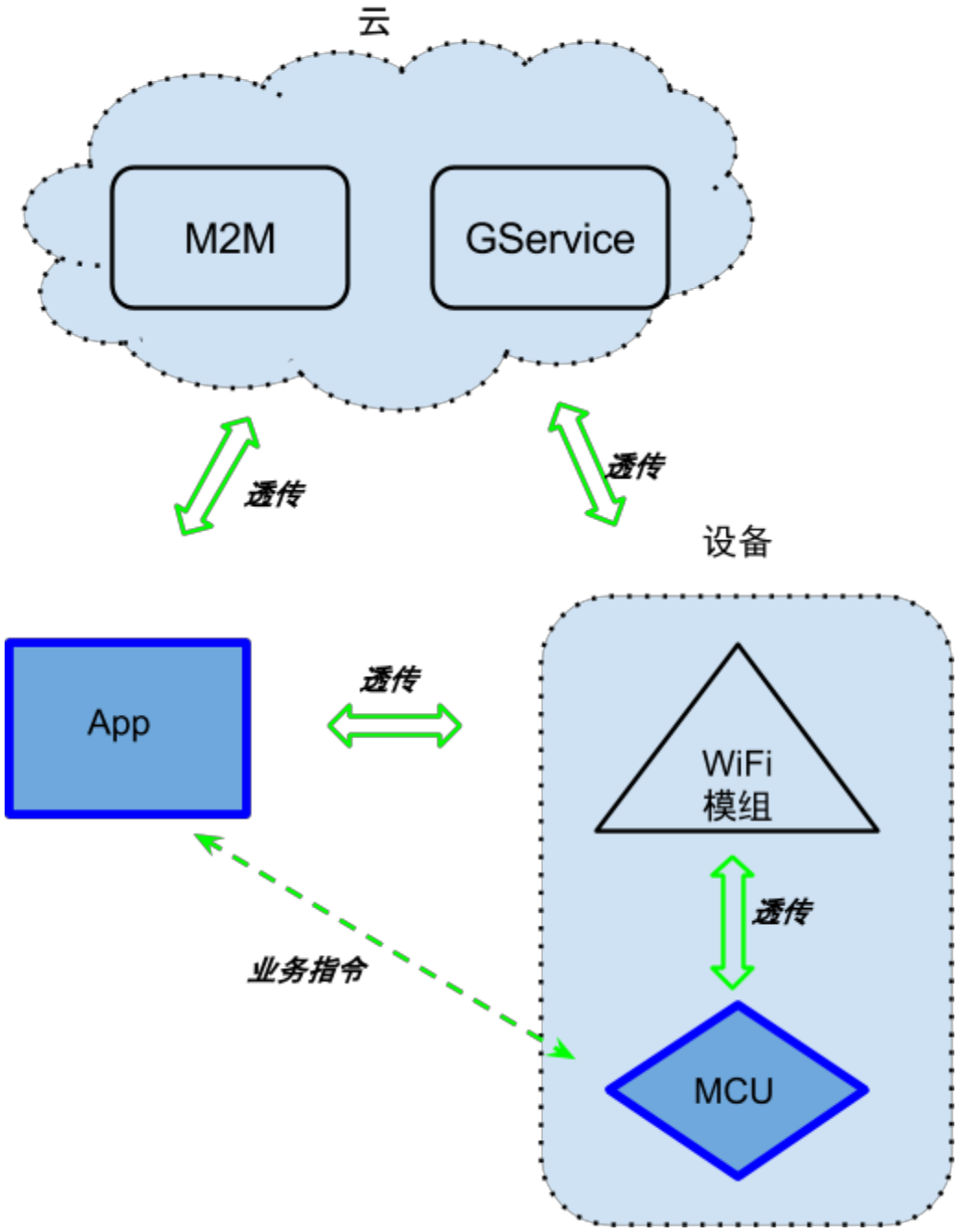
修订历史

版本	修订内容	修订人	修订日期
4.0.0	V4版协议	刘冠华	2014-11-17
4.0.1	修改小类的排序方式，由6小类改为3小类排序	刘冠华	2014-12-03
4.0.2	1. 规定y, k, m的整数和小数部分合起来最多8位数字，其中小数部分最多4位数字；k的值必须大于0 2. 添加透传指令(0x05, 0x06) 3. 添加中控协议	刘冠华	2015-02-15

目录

1. 通讯模型

App与设备间的业务逻辑指令是经WiFi模组或云端透传至目的地的。



为了规范并简化设备业务逻辑协议的制定，制定本通用数据点协议。

2. 数据点协议概述

2.1. 数据点分类。设备的状态和可控制点可分为四大类（下文简称**大类**）：

- **只读**：不能被控制或修改，只能由设备上报。如环境温度，湿度等。
- **可读写**：可以被远程控制或修改，设备同时也会上报当前值。如开关，马达转速等。
- **报警**：不可被控制或修改，只能由设备上报，有警示作用。如温度过高，门没有关好等。
- **故障**：不可被控制或修改，只能由设备上报，提示出现了硬件或软件故障。如抽风马达坏，温度传感器坏等。

2.2. 数据点数据类型。每一种数据点可以指定为以下的一种数据类型：

- 布尔型(bool)：占一个bit的长度
- 枚举型(enum)：占一个或多个bit的长度，最多8 bits
- 整型(uint)：长度可分为1B(uint8)，2B(uint16)，4B(uint32)几种
- 二进制数据(binary)：占一个或多个字节，最大长度为2048B

3. 约定

3.1. 协议阅读说明

- **长度**：一般由一个(1B)或两个字节(2B)组成。若多于一个字节组成，采用大端编码方式，即高字节在前，低字节在后。
- **指令格式**：每条指令由命令(1B)和数据内容两部分组成，即cmd(1B), data(xB)。

4. 通讯协议

4.1. 写数据点

写数据点，即控制设备或给设备下发命令。

写数据点，App ⇒ 设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x01
2	Flags		由可读写数据点的个数据决定，一个可读写数据点占用一个bit，表示后接对应数据点的值是否有效，1为有效，0为无效。Flags必须为8 bits的整数倍，多出的bits的值无意义。由右至左编号为bit0 ~ bitx(x为8的整数倍减1)， bit0表示第一个可读写数据点的值是否有效，bit1表示第二个可读写数据点的值是否有效，依次类推。
3	可读写数据点的值		依次由每一个可读写数据点的值组成。按以下次序出现（下文简称**小类**）： 1. 布尔型(bool) 2. 枚举型(enum) 3. uint8、uint16、uint32、二进制数据(binary) uint8、uint16、uint32和binary有同等的排序优先级，也就是说bool数据点一定出现在enum前，eunm一定出现在uint8、uint16、uint32和binary前，但uint8可出现在uint16后，binary可出现在uint8前。同种数据类型的次序以数据点UI编辑界面出现的次序排序。

			bool和enum在JSON的单位必须为bit，而其它的数据类型的单位必须为byte。bool和enum以占用最小空间为原则。所以bool和enum会合用同一个byte或enum会出现跨byte的情况。生成时，需要先计算bool和enum该占用的总byte数，然后从最右边的byte的最右边的bit开始从右至左以bit0开始编号和放值。

例：现有下面的可读写数据点（按数据点UI编辑界面的次序出现）。注意只读、警告或故障数据点不需考虑。

- 1个binary类型的数据点称为bin_0，宽度为8 bytes
- 1个2 bits宽的enum称为enum_0
- 1个uint8类型的数据点称为uint8_0
- 1个uint16类型的数据点称为uint16_0
- 1个uint8类型的数据点称为uint8_1
- 5个bool类型的数据点分别称为bool_0、bool_1、bool_2、bool_3和bool_4
- 1个3 bits宽的enum称为enum_1

则组成数据包如下

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x01
2	Flags	2	见下面“Flags的值”
3	bool和enum数据类型的值	2	见下面“bool和enum的值”
4	bin_0的值	8	
5	uint8_0的值	1	
6	uint16_0的值	2	
7	uint8_1的值	1	

Flags的值。由于共有11个数据点，需占用2个bytes，从左至右编号为byte0和byte1，从byte1的最右边的bit0开始向从右至左以bit0开始编号，得bit0 ~ bit15。由于只有11个数据点，所以多出的bit11 ~ bit15的不使用。

byte0

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
					uint8_1	uint16_0	uint8_0

byte1

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
bin_0	enum_1	enum_0	bool_4	bool_3	bool_2	bool_1	bool_0

如全部值有效，则值为0b00000111 11111111。
如只有bool_1的值有效，则值为0b00000000 00000010。
如只有bool_2和uint8_1的值有效，则值为0b00000100 00000100。

bool和enum的值。共需占用2个bytes，从左至右编号为byte0和byte1，从byte1的最右边的bit0开始向从右至左以bit0开始编号，得bit0 ~ bit15，从bit0开始先放入各个bool类型的值，再放入enum类型的值，最后的bit10 ~ bit15不使用。

byte0								byte1							
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
						enum_1	enum_1	enum_1	enum_0	enum_0	bool_4	bool_3	bool_2	bool_1	bool_0

各数据点的byte_offset和bit_offset如下：

数据点	byte_offset	bit_offset
bool_0	0	0
bool_1	0	1
bool_2	0	2
bool_3	0	3
bool_4	0	4
enum_0	0	5
enum_1	0	7
bin_0	2	0
uint8_0	10	0
uint16_0	11	0
uint8_1	13	0

4.2. 读数据点

读取设备的当前状态。

App读数据点请求，App ⇒ 设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x02

设备回复数据点内容，设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x03
2	全部数据点的值		依次由每一个数据点的值组成。按以下次序出现（即大类的次序）： 1. 可读写(status_writable) 2. 只读(status_readonly) 3. 报警(alert) 4. 故障(fault) 也就是说可读写数据点一定出现在只读数据点前，只读数据点一定出现在报警数据点前，报警数据点一定出现在故障数据点前。同种数据点类型的次序（小类的次序）请参考上面“写数据点”那一节。

例：现有下面的数据点（按数据点UI编辑界面的次序出现）。

- 2个uint8类型的数据点分别称为w_uint8_0和w_uint8_1，为可读写类型
- 2个bool类型的数据点分别称为f_bool_0和f_bool_1，为故障类型
- 1个bool类型的数据点称为a_bool_0，为报警类型
- 2个bool类型的数据点分别称为w_bool_0和w_bool_1，为可读写类型
- 1个2 bits宽的enum称为w_enum_0，为可读写类型

则组成数据包如下：

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x01
2	可读写bool和enum数据类型的值	1	见下面“可读写bool和enum数据类型的值”
3	w_uint8_0的值	1	
4	w_uint8_1的值	1	
5	报警bool数据类型的值	1	见下面“报警bool数据类型的值”
6	故障bool数据类型的值	1	见下面“故障bool数据类型的值”

可读写bool和enum数据类型的值。共需占用1个byte，从右至左以bit0开始编号，得bit0 ~ bit7，如下表放入值，最后的bit3 ~ bit7不使用。

byte0							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
					w_enum_0	w_bool_1	w_bool_0

报警bool数据类型的值。共需占用1个byte，从右至左以bit0开始编号，得bit0 ~ bit7，如下表放入值，最后的bit1 ~ bit7不使用。

byte0							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
							a_bool_0

故障bool数据类型的值。共需占用1个byte，从右至左以bit0开始编号，得bit0 ~ bit7，如下表放入值，最后的bit2 ~ bit7不使用。

byte0							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
						f_bool_1	f_bool_0

各数据点的byte_offset和bit_offset如下：

数据点	byte_offset	bit_offset
w_bool_0	0	0
w_bool_1	0	1
w_enum_0	0	2
w_uint8_0	1	0

w_uint8_1	2	0
a_bool_0	3	0
f_bool_0	4	0
f_bool_1	4	1

4.3. 状态上报

设备定期(每10分钟)或当状态改变后主动上报当前状态。如状态改变的是由于App控制所引起的，设备应在状态改变后立刻上报状态。如状态的改变是由于设备所在的环境状况或用户通过遥控或物理按键所引发的状态改变，上报状态的频率最快为2秒每次。

设备状态上报，设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x04
2	全部数据点的值		依次由每一个可读写、只读、报警及故障数据点的值组成。请参考上面“读数据点”那一节。

4.4. 透传业务指令

App和设备MCU可以直接透传业务指令。

App向设备MCU透传业务指令，App ⇒ 设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x05
2	业务指令内容		

设备MCU向App透传业务指令，设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x06
2	业务指令内容		

4.5. 写子设备数据点（中控）

写子设备数据点，即控制子设备或给子设备下发命令。

写数据点，App ⇒ 子设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x07
2	SDID/GroupID	4	代表写数据点的目标设备。最高bit为1表示值为GroupID，否则为SDID。
3	Flags		与“写数据点”定义一致。
4	可读写数据点的值		依次由每一个可读写数据点的值组成。请参考上面“写数据点”那一节。

4.6. 读子设备数据点（中控）

读取子设备的当前状态。

App读数据点请求，App ⇒ 子设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x08
2	SDID	4	Sub-Device-ID，代表读数据点的目标设备

子设备回复数据点内容，子设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x09
2	SDID	4	Sub-Device-ID，子设备ID
3	全部数据点的值		依次由每一个可读写、只读、报警及故障数据点的值组成。请参考上面“读数据点”那一节。

4.7. 子设备状态上报（中控）

子设备定期(每10分钟)或当状态改变后主动上报当前状态。如状态改变的是由于App控制所引起的，子设备应在状态改变后立刻上报状态。如状态的改变是由于子设备所在的环境状况或用户通过遥控或物理按键所引发的状态改变，上报状态的频率最快为2秒每次。

子设备状态上报，子设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0a
2	全部数据点的值		依次由每一个可读写、只读、报警及故障数据点的值组成。请参考上面“读数据点”那一节。

4.8. 添加子设备（中控）

由App向中控设备发起添加子设备的请求，中控设备完成添加子设备的动作。

添加子设备，App ⇒ 中控设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0b

4.9. 删除子设备（中控）

由App向中控设备发起删除子设备的请求，中控设备完成删除子设备的动作。

删除子设备，App ⇒ 中控设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0c
2	SDID	4	被删除子设备ID

4.10. 子设备列表（中控）

由App向中控设备查询当前的子设备表列信息。

请求查询子设备，App ⇒ 中控设备。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0d

中控设备响应子设备列表信息，中控设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0e
2	产品个数	2	大端字节序

3	产品信息列表		依次由一个或多个产品信息项组成，每一项的结构请见下表“产品信息”

产品信息：

序号	字段名称	字节长度(B)	内容说明
1	产品标识码	32	即product_key
2	子设备个数	2	大端字节序，表示当前产品的子设备个数
3	子设备信息		依次由一个或多个子设备信息项组成，每一项的结构请见下表“子设备信息”

子设备信息：

序号	字段名称	字节长度(B)	内容说明
1	SDID	4	子设备ID
2	是否在线	1	0表示离线，1表示在线

4.11. 子设备列表变更通知（中控）

当中控设备的子设备列表信息发生变化时，中控设备应把最新的子设备列表信息通知到App。

中控设备推送最新子设备列表消息，中控设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x0f
2	产品个数	2	大端字节序
3	产品信息列表		依次由一个或多个产品信息项组成，每一项的结构请参见“子设备列表中的“产品信息”

4.12. 子设备上下线状态变更通知（中控）

当中控设备的子设备上下线状态发生变化时，中控设备应把最新的状态通知到App。

中控设备推送子设备上下线状态变化消息，中控设备 ⇒ App。

序号	字段名称	字节长度(B)	内容说明
1	命令	1	0x10
2	SDID	4	子设备ID

3	是否在线	1	0表示离线，1表示在线

4.13. JSON Template

这个JSON是用于存放用户编辑完成后的数据点定义。

```
{
   version: “1.0”,
   protocolType: "standard",
   packetVersion: "0x00000004",
   product_key: <str>,
   entities: [
   {
      id: <int>,
      name: <str>,
      display_name: <str>,
      attrs: [
      {
         id: <int>,
         name: <str>,
         display_name: <str>,
         desc: <str>,
         type: "status_writable"/"status_readonly"/"alert"/"fault",
         data_type: "bool"/"enum"/"uint8"/"uint16"/"uint32"/"binary",
         position:
         {
            byte_offset: <int>,
            bit_offset: <int>, (default=0, from right to left)
            unit: "bit"/"byte", (for bool and enum is "bit"; for others, is "byte")
            len: <int> (for "binary" data_type, max len is 2048 bytes)
         },
         uint_spec:
         {
            min: <int>, (x最小值)
            max: <int>, (x最大值)
            ratio: <int/float>, (修正系数k, default=1),
            addition: <int/float>, (增量m, default=0)
                  ( y=k*x+m,
                    x=(y-m)/k,
                    x必须为整数(uint),
                    y, k, m 可以是浮点数，整数和小数部分合起来最多8位数字，其中的小数部分最多4位数字;
                    k的值必须大于0;
                    x是通讯传输的值, y是App UI界面的显示值)
           },
           enum: ["item1", "item2", ...]
      },
      ...
   ]
   },
```

```
    ...
  ]
}
```