MQTT 对接规范

目录

该文档主要是对客户端和服务器之间的通信协议进行的一个说明，客户端已经基于该 proto 文件进行了开发，服务器需要按照该 proto 文件进行相应的开发，来实现双方的通信。

订阅和发布主题

1、心跳

发送心跳的 topic：　　AP/echo

客户端会每 10s 往该主题发布设备信息;服务器端则需要订阅该主题,从信息中去解析，获取设备的相关状态信息。

2、配置

下发配置的 topic：　　AC/%s/M_WRconfig　　// 中间填充序列号

客户端会订阅该主题消息，当服务器在该主题上发布了消息后，客户端会根据收到的信息，去解析配置内容。

3、命令

下发命令的 topic：　　AC/%s/M_cmd　　// 中间填充序列号

客户端会订阅该主题消息，当服务器在该主题上发布了消息后，客户端会根据收到的信息，去执行相应的命令。

4、回复

消息确认回复的 topic：　　AP/response

客户端在收到配置或者命令下发后， 会往该主题上发布相应的确认信息，服务器则需要订阅该主题，从信息中去解析，获取设备的执行结果。

Proto 说明

Protocol Buffers（简称 ProtoBuf）是由谷歌开发的一种轻量级、高效的数据交换格式,它基于二进制流,用于结构化数据序列化,通常用于通信协议、数据存储等领域。ProtoBuf 的文件通常以.proto 为扩展名，称为 Proto 文件，用于定义数据结构的消息类型以及消息之间的交互规则。

Proto 文件说明

类型：

1、AP 工作模式：

```
enum APMode {
        FIT_AP = 1;
        FAT_AP = 2; /* soho */
        CPE_BASE = 3; /* CPE 基站*/
        CPE_STA = 4; /* CPE 接收端*/
}
```

2、射频类型；

```
enum RadioBand {
        RB_2G = 1;
        RB_5G = 2;
}
```

3、黑白名单类型；

```
enum MACListType {
        MT_WHITE_LIST = 1;
        MT_BLACK_LIST = 2;
}
```

4、HTMode;

```
enum RadioHTMode {
        RHT_20 = 1;
        RHT_40 = 2;
        RHT_40Minus = 3;
        RHT_40Plus = 4;
        RHT_80 = 5;
```

```
        RHT_160 = 6;
        RHT_160Plus = 7;
    }
```

5、回复到云平台的消息类型

```
enum WRType {
        W_REBOOTDEV = 1;     /* 重启设备 */
        W_REBOOTWIFI = 2;     /* 重启无线 */
        W_SCHREBOOT = 3;      /* 定时重启无线 */
        W_UPGRADE = 4;        /* 升级 */
        W_SETACADDR = 5;      /* 重置云 ac 地址   */
        W_SETHOSTNAME = 6;    /* 配置设备名称 */
        W_PINGWDT = 7;        /* 看门狗   */
        W_WHITELIST = 8;      /* 黑白名单 */
        W_SSID = 9;           /* SSID */
        W_RADIO = 10;         /* RADIO */
        W_NETWORK = 11;       /* NETWORK */
        W_WIFIDOG = 12;       /* WiFiDog*/
        W_SCHREBOOTDEV = 13;      /* 定时重启设备 */
        W_REBOOTAP = 14;      /* 重启 AP */
        W_SHELL = 100;        /* 执行下发的 shell 命令 */
}
```

6、命令类型

```
enum CMDType {
        REBOOT = 1;
        UPGRADE = 2;
        SETACADDR = 4;
        /* 解绑 AP 和 AC， AP 和 AC 的绑定关系由配置下发时确定 */
        UNBIND = 5;
        /* 下线用户支持 */
        KICK_USER = 6;
        /* 配置设备名称 */
        SETHOSTNAME = 7;
        /* 没有绑定的时候要求 AP 断开和 AC 的连接，重启查询 AC 的流程*/
        KICK_AP = 8;
        /* 启动扫描 */
        START_SCAN = 9;
        /* 下发认证 */
        AUTH = 10;
```

```
        /*  下线用户  */
        LOGOUT = 11;
        /*  下线用户  */
        REBOOTAP = 12;

        SHELL = 100;
}
```

消息类型

1、配置下发

WRConfNetwork：核心配置入口

和云平台进行交互的消息体；该消息体中包括了网络、射频、ssid、黑白名单、看门狗、计划任务等信息；云平台下发配置，需封装该结构，然后发布到配置下发的主题中，客户端订阅该主题，收到信息后，解析该结构体，并根据里面的配置，去修改设备上面的参数。

每一次下发一个配置，不能所有配置一起下发。

```
message WRConfNetwork {
        /*  网络配置  lan/wan */
        optional Network network = 21;

        /*  基础无线配置  */
        repeated WlanConfig wlanconfig = 22;

        /* RADIO  射频配置  */
        repeated RadioConfig radioconfig = 23;

        /*  无线黑白名单配置  */
        optional MACListType maclist_type = 24 [default = MT_WHITE_LIST];
        repeated string mac = 25;

        optional PingWatchdog pingwatchdog = 26;

        /* wifi  定时开关  */
        optional int32 wifiSch    = 201;
        optional FuncSchedule wifiSchedule = 27;

        optional Wifidog wifidog = 28;
        optional string auto_reboot    = 29;
```

}

Network：网络配置

封装 lan/wan 的消息体；
WanConfig： 定义 lan/wan 消息体；

```
message WanConfig {
        required string ipproto = 1;
        optional string ip = 2;
        optional string netmask = 3;
        optional int32 metric = 4;
        optional string gateway = 5;
        optional string dns1 = 6;
        optional string dns2 = 7;
}

message Network {
        optional WanConfig wanconfig = 1;          /* 网络配置 wan */

        optional WanConfig lanconfig = 2;          /* 网络配置 lan */
}
```

Network 中可同时封装内外网接口参数，也可以单独配置其中一项。

数据内容：

字段名称	属性名称	权限	备注
ipproto	网络模式：dhcp/satic	读写	
ip	IP 地址	读写	网络模式为 static 下有效
netmask	子网掩码	读写	网络模式为 static 下有效
gateway	网关	读写	网络模式为 static 下有效
dns1/dns2	DNS	读写	
ip	内网 IP 地址	读写	工作模式为路由模式下有效
netmask	内网掩码	读写	工作模式为路由模式下有效
dns1/dns2	内网 DNS	读写	工作模式为路由模式下有效

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

WlanConfig：无线配置

定义 SSID 消息体；

```
message WlanConfig
{
        required RadioBand band = 1;
        required string ssid = 2;
        required int32 gbk_enable = 3 [default = 0];
        required string encryption = 4;
        optional string key = 5;
        required int32 disabled = 6 [default = 0];

        required int32 vlan = 10 [default = 0];
        required int32 maxsta = 11 [default = 0];
        required int32 rejrssi = 12 [default = -85];
        required int32 wmm = 13 [default = 1];
        required int32 isolate = 14 [default = 0];
        required int32 hide = 15 [default = 0];
        required int32 ieee80211r = 22 [default = 0];
        optional int32 auth_type= 25; /*  认证类型,将认证方式和 ssid 关联*/
}
```

数据内容：

对不使用的字段，直接使用默认值代替。

字段名称	属性名称	权限	备注
band	无线模块	读写	**2.4G/5G**
ssid	SSID	读写	
encryption	加密方式	读写	
key	密码	读写	

wmm	wmm 开关	读写	
isolate	终端隔离	读写	
hide	隐藏无线	读写	
vlan	vlan id	读写	
ieee80211r	ieee80211r	读写	
maxsta	最大连接数	读写	
auth_type	认证方式	读写	
rejrssi	rejrssi	读写	

使用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

RadioConfig：射频配置

定义 Radio 消息体；

```
message RadioConfig {
        required RadioBand band = 1;
        optional string hwmode = 3;
        optional string htmode = 4;
        optional int32 channel = 5;
        optional int32 txpower = 6;
        optional int32 rejrssi = 8 [default = -85];
        optional string country = 9;
        optional int32 enable_fils = 10 [default = 1];
}
```

数据内容：

对不使用的字段，直接使用默认值代替。

字段名称	属性名称	权限	备注
band	无线模块	读写	**2.4G/5G**
htmode	频宽	读写	
channel	信道	读写	
txpower	发射功率	读写	

rejrssi	弱信号拒绝阀值	读写	
country	国家码	读写	
enable_fils	快速连接开关	读写	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

MAC： 黑白名单

maclist_type：定义下发的名单类型，取值范围 3、黑白名单类型；
mac： 定义名单列表。

数据内容：

字段名称	属性名称	权限	备注
maclist_type	名单类型	读写	
mac	名单列表	读写	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

PingWatchdog：看门狗配置

定义看门狗消息体
message PingWatchdog {
required int32 enable = 1;
optional string target = 2;

```
        optional int32 ping_interval = 3;
        optional int32 ping_failures = 4;
        optional int32 ping_timeout = 5;
        optional int32 ping_watchdog_action = 6 [default = 3];
}
```

数据内容：

对不使用的字段，直接使用默认值代替。

字段名称	属性名称	权限	备注
enable	开启、关闭	读写	
target	检测目的地址	读写	
ping_interval	检测间隔	读写	
ping_failures	失败次数	读写	
ping_timeout	Ping 超时时间	读写	
ping_watchdog_action	动作	读写	重启设备 /关闭无线 /重启网络 /开启救援 SSID /无动作，对应取值范围【0,1,2,3,4】

调用方式：

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

FuncSchedule:计划任务

定义计划任务 消息体，和 wifiSch 配套使用，当开启 wifi 定时重启，需设置 wifiSch 为 1， 并下发对应对应的计划任务；

```
message FuncSchedule {
        /* Monday Tuesday Wednesday Thursday Friday*/
        required string daysofweek = 1;

        required string starttime = 2;          /* 12:01 */

        required string stoptime = 3;           /* 23:01 */
```

}

数据内容：

对不使用的字段，直接使用默认值代替。

字段名称	属性名称	权限	备注
daysofweek	开始的日期	读写	值范围为【0,1,2,3,4,5,6】,对应【星期天,星期一，星期二，星期三，星期四，星期五，星期六】
starttime	检测目的地址	读写	
stoptime _interval	检测间隔	读写	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

WiFiDog:

定义 wifidog 的消息体；

```
message Wifidog {
        optional string auth_server = 1; /* 认证服务器地址 */
        optional string auth_port = 2;      /* 认证服务器端口 */
        optional string portal = 3;       /* portal 路径 */
        optional int32   auth_time = 4; /*认证时效*/
        repeated string ip = 5; /* 放行的 ip 列表 */
        repeated string mac = 6; /* 放行的 mac 列表 */
}
```

数据内容：

字段名称	属性名称	权限	备注
auth_server	服务器地址	读写	

auth_port	服务器端口	读写	
portal	Portal 路径	读写	
auth_time	认证时效	读写	
ip	放行的 ip 列表	读写	
mac	放行的 mac 列表	读写	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

auto_reboot：自动重启

封装该字符串，让设备按照配置自动重启，该字符串下发规范，请按照以下说明执行：

字段说明：

apreboot：定义开启或者关闭，取值：true / false

apreboottime: 定义执行的具体时间，比如 03:35

aprebootloop: 定义计划任务是每天、每周、每月，依次对应的值是 1、2、3

apday： 定义具体哪一天，该参数仅对每周和每月有效，当 aprebootloop 是为 2，则 apday 的取值范围为【0,1,2,3,4,5,6】,对应【星期天，星期一，星期二，星期三，星期四，星期五，星期六】，当 aprebootloop 是 3， 则 apday 是从 1 开始取值，1~31.

1、以每天 03:35 执行重启为例说明：

服务器需直接下发完整字符串如下：

```
"/sbin/uci set system.@bingbee[0].apreboot='true';"
"/sbin/uci set system.@bingbee[0].apreboottime='03:35';"
"/sbin/uci set system.@bingbee[0].aprebootloop=1;"
"/sbin/uci set system.@bingbee[0].apday=1"
"/sbin/uci commit system;"
"/etc/init.d/prework start;"
```

2、以每周星期天 03:35 执行重启为例说明：

服务器需直接下发完整字符串如下：

```
"/sbin/uci set system.@bingbee[0].apreboot='true';"
"/sbin/uci set system.@bingbee[0].apreboottime='03:35';"
"/sbin/uci set system.@bingbee[0].aprebootloop=2;"
"/sbin/uci set system.@bingbee[0].apday=0"
```

"/sbin/uci commit system;"
"/etc/init.d/prework start;"

3、以每月 1 号 03:35 执行重启为例说明：

服务器需直接下发完整字符串如下：

"/sbin/uci set system.@bingbee[0].apreboot='true';"
"/sbin/uci set system.@bingbee[0].apreboottime='03:35';"
"/sbin/uci set system.@bingbee[0].aprebootloop=3;"
"/sbin/uci set system.@bingbee[0].apday=1"
"/sbin/uci commit system;"
"/etc/init.d/prework start;",

4、关闭自动重启

服务器需直接下发完整字符串如下：

"/sbin/uci set system.@bingbee[0].apreboot='false';"
"/sbin/uci commit system;"
"/etc/init.d/prework start;"

2、消息回复

Resonse：应答消息

定义应答消息 ，针对每一个来自云上下发指令的回复。

```
message Response {
        required WRType type = 1;
        required string sn = 2;
        required int32 flag = 3 [default = 1];
        required string reasons = 4;
}
```

字段名称	属性名称	取值	备注
Type	命令类型	WRType	5、回复到云平台的消息类型
Sn	设备序列号		
Flag	成功、失败	**1、0**	
Reasons	失败理由		

3、命令下发

CMD: 核心命令下发入口

```
message CMD {
        required CMDType type = 1;
        optional string args = 2;
}
```

CMDType: 定义于 6、命令类型
args: 定义下发的参数，参数以”;”进行分隔。

比如下发在线升级命令。
CMDType 对应的 type 字段设置为：UPGRADE
args:对应的字符串的组成为 url;md5

对没有参数的，直接填充 type。

设备名称配置:

数据内容：

字段名称	属性名称	取值	备注
type	命令类型	SETHOSTNAME	
args	参数	**SAILSKY**	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

软件升级:

数据内容：

字段名称	属性名称	取值	备注
type	命令类型	UPGRADE	
args	参数	http://www.xxx.com/firmware;5951a9b6fda715748aa718a3fbaf807a	对应的字符串的组成为 url;md5

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

设备重启:

数据内容:

字段名称	属性名称	取值	备注
type	命令类型	REBOOT	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

客户端进程重启:

数据内容：

字段名称	属性名称	取值	备注
type	命令类型	REBOOTAP	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

AC 地址修改:

数据内容：

字段名称	属性名称	取值	备注
type	命令类型	SETACADDR	
args	参数	AC 地址	

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

下线无线终端:

数据内容：

字段名称	属性名称	取值	备注
type	命令类型	KICK_USER	
args	参数	00:11:22:33:44:55;22:33:44:55:66:66	mac 列表，以;分隔

调用方式:

管理模式	协议	调用方式	说明
云端管理	MQTT	接口调用	云平台主动发送查询、修改命令。接收命令处理之后 AP 回应结果

4、心跳上报

Echo： 心跳消息入口

Echo 定义了设备的基本信息。没有的字段，上报为空。

```
message Echo {
        required string sn = 1;
        required string product_name = 2;
        /* 标识设备的唯一 MAC */
        optional string mac = 3;
        optional string board = 4;
        optional string hostname = 5;
        /* 运行时间 */
        optional string uptime = 6;
        optional uint64 uptime_sec = 61;

        optional string version = 7;
        required APMode apmode = 8 [default = FIT_AP];


        /* 是否是第一次连接(要求下发配置) */
        required int32 newconnect = 9 [default = 0];

        /* 关键配置的 MD5 值 */
        optional string apnetwork_md5 = 10;

        optional string country = 11;

        /* CPU 占用率*/
```

```
        optional string cpu = 12;


        /* 是否云端管理的 */
        optional int32 is_on_cloud = 13 [default = 0];
        optional string username = 14;
        /* 通过设备的上下行总流量统计 */
        optional uint64 uploadspeed = 21;
        optional uint64 downloadspeed = 22;
        optional uint64 uploadbytes = 23;
        optional uint64 downloadbytes = 24;

        required string acaddr = 25;

        required ManageInterface mif = 50;   /* 接口信息 */
        repeated RadioInfo radioinfo = 51;    /* 射频信息 */
        optional WanConfig lanconfig = 52;      /* lan 口配置 */
        optional PingWatchdog pingwatchdog = 53;    /* Ping 看门狗 */
        repeated WlanConfig ssids = 54;             /*   ssid 配置   */


        optional int32 iptvSupport = 70;
        optional int32 iptvEnable = 71;

        /* 假设所有的 CPE 都是单频的，单频 2.4 或单频 5G */
        /* 增加一个 CPE 专用的信息上报,信道列表*/
        optional string cpeChannelsJson = 100;

    }
```

射频信息中包括多 wlaninfo，每一个 wlaninfo 包括了多个 sta。

StaInfo:

```
    定义终端信息
    message StaInfo {
            required string mac = 2;
            optional string ip = 3;
            required int32 signal = 4;
            required int32 noise = 5;
            required int32 snr = 6;
            required string txrate = 7;
            required string rxrate = 8;
    }
```

WlanInfo：

定义无线信息

```
message WlanInfo {
        optional string ssid = 1;
        repeated StaInfo stas = 2;
}
```

RadioInfo：

定义射频信息

```
message RadioInfo {
        required RadioBand band = 1;
        repeated    WlanInfo wlaninfo = 2;

        /* 基础三属性 */
        required RadioHTMode htmode = 3 [default = RHT_20];
        required uint32 txpower = 4 [default = 0];
        required uint32 channel = 5 [default = 0];

        /* 其他属性 */
        optional int32 signal = 20;
        optional int32 noise = 21;
        optional string bitrate = 22;

        /* 增加字段 */
        optional int32 maxsta = 23 [default = 0];
        optional int32 rejrssi = 24 [default = -85];
        optional string country = 25;
        optional int32 enable_fils = 26 [default = 1];
        optional string mac = 27;

}
```

ManageInterface：

定义接口信息

```
message ManageInterface {
        required string ifname = 1;
        optional string ip = 2;
        optional string mac = 3;
        optional string netmask = 4;
        optional string gateway = 5;
        optional string dns1= 6;
        optional string dns2= 7;
        optional string ipproto = 8; // dhcp static pppoe
}
```

MQTT 参数重置

登录 AP web 页面，以 AP 的地址为 192.168.1.2 为例，登录后访问 192.168.1.2/mqtt

按照实际的情况填写用户名、密码和端口。