# 设备通讯 MQTT 协议

V1.5.0

| 版本 | 更新时间 | 更新内容 | 负责人 |
| --- | --- | --- | --- |
| V1.0 | 2015.08.09 | 修订 HTTP 通讯协议 | Jason.Huang |
| V1.3 | 2018.11.05 | 修订 TCP 通讯协议 | Jason.Huang |
| V1.4.0 | 2020.04.17 | 修订 MQTT 通讯协议 | Jason.Huang |
| V1.4.1 | 2020.08.28 | 1. 更新在线呼叫方式 3.10IP<br>2. 更新在线验证方式 3.11 | Benson.Chen |
| V1.4.2 | 2021.07.14 | 3.1 设备鉴权->增加 commType 通讯协议字段上传和服务端返回，以及增加服务端签名校验<br>3.4 设备控制->修改连接服务器地址增加可切换通讯协议 | Lynn.Chen |
| V1.4.3 | 2021.11.25 | 1.更新 3.5 设备控制->增加开启关闭紧急常闭，TRTC 进入房间指令；<br>2.新增 4.11 通行时间策略表和 4.12 通行时间策略用户关联表； | Lynn.Chen |
| V1.4.4 | 2022.05.06 | 1.实时事件类型中 废除 1126 健康码开门成功和失败的编码，改为 26 健康码开门成功 和 27 健康码开门失败 的编码 | Lynn.Chen |
| V1.4.5 | 2022.08.06 | 1.增加协议规范设计（设备对每条下发的命令不论成功失败都需要上传对应执行结果）；<br>2.新增【房间 ID 或卡号或密码/二维码 number】，原字段为 cardNo，改为使用 number 字段；<br>3.新增【上传心跳 "resource":"device/control?cmd=heartbeat"】 | Lynn.Chen |
| V1.4.6 | 2022.09.19 | 1.设备鉴权连接【TCP】新增操作系统或镜像版本，SD 卡大小，相机总数，人脸 SDK 版本，人脸 SDK 状态 5 个字段上传；<br>2.定时上传心跳及同步时间 新增 tableData 和 systemData 等数据字段上传；<br>3.设备参数说明->人脸|人证参数 新增【抓拍图片上传 uploadRecordImage】字段，默认 1 智能模式; ( 0 关闭;1 智能模式 2 | Lynn.Chen |

| | | | |
|---|---|---|---|
| | | 启用【智能模式 4G 网络默认不上传，其他网络默认上传】）<br>4.新增语音模板表 | |
| V1.4.7 | 2023.09.06 | 1.设备控制->增加开启和关闭紧急常开的指令 | Lynn.Chen |
| V1.4.8 | 2024.3.15 | 1.更新-对讲-结束通话增加 callMode 字段【1：内网接听；2:外网接听；3：电话转接；4：TRTC；5：IPPBX】;<br>2.新增 获取设备门磁状态[室内机] 指令；<br>3.更新-远程开门-增加字段 scene 场景【0：普通，1：对讲 】，callId：呼叫标识 ID<br>4.更新-事件记录-type 字段增加 【35：对讲开门；36：转接电话开门】; | Lynn.Chen |
| V1.4.9 | 2024.5.22 | 1.新增-设备参数-语音参数；<br>2.新增-文字转语音接口； | Lynn.Chen |
| V1.5.0 | 2024.6.21 | 1.设备参数-对讲参数-新增-对讲呼叫方式 sipCallWay 字段 0：群呼；1：选呼；2：循呼【默认 0】,门口机使用；<br>2.新增-对讲-查询用户 指令，用于门口机或室内机选人呼叫场景； | Lynn.Chen |

# 1.通讯流程概述

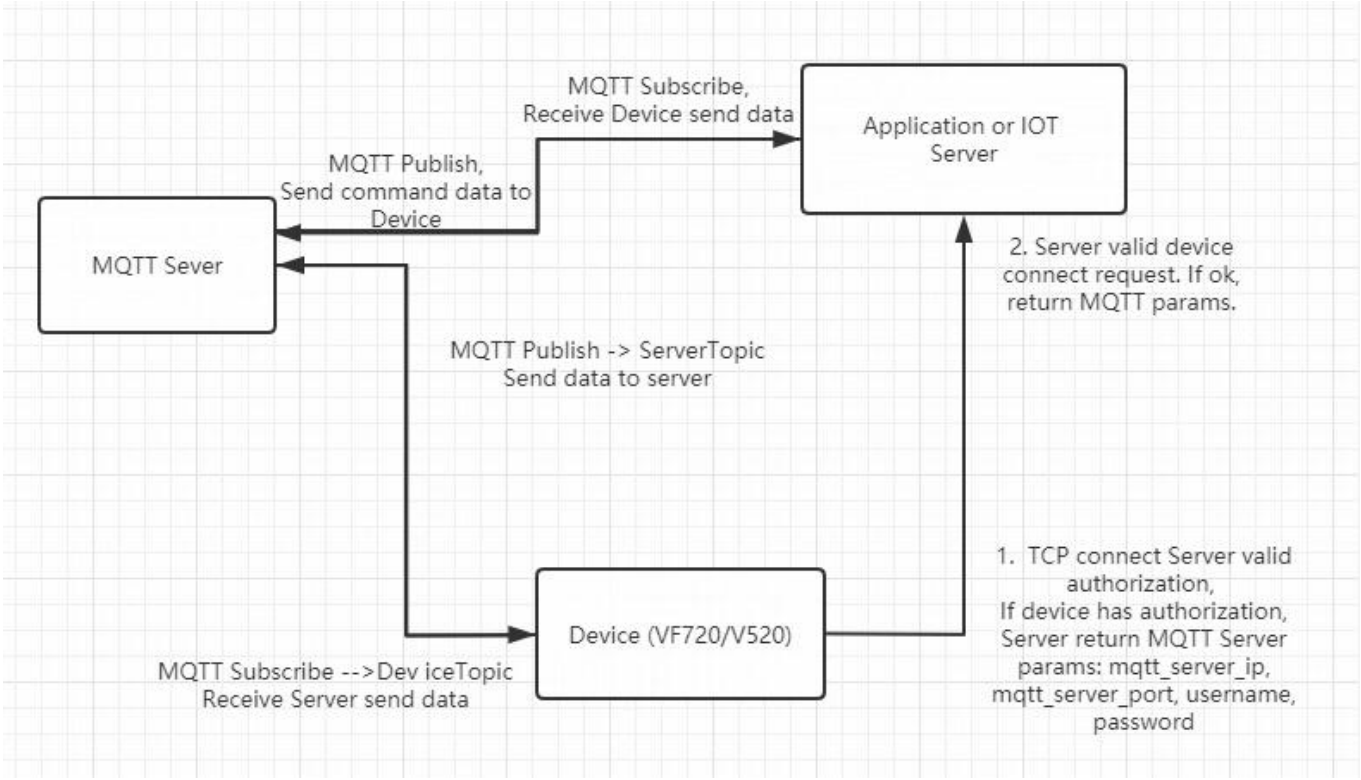
本协议采用 TCP Socket + MQTT 通讯方案， TCP 通讯用于设备首次连接鉴权、交互参数， MQTT 通讯用于常规数据上传、下发。

**设备端->MQTT 服务器->IOT 服务器，通讯流程设计：**

1. 设备启动后，按设备配置参数：YunServer，进行 TCP 连接服务器鉴权；
2. TCP 服务器收到设备请求后，校验设备鉴权；
3. 鉴权验证通过后，返回 MQTT 服务器参数给设备端；
4. 设备端获得 MQTT 参数，按分配的参数连接 MQTT 服务器，订阅相关 Topic，注意：TCP 服务器的端口不能使用 8089 端口，推荐使用 6010 端口 ；
5.

6. 设备数据上传：设备端产生事件、状态等数据，发布到指定的 Topic，服务器端订阅 Topic 获取数据；
7. 服务端数据下发：服务器端需要下发卡权限、人脸权限数据，发布到指定的 Topic，设备端订阅获取数据，执行完成后，将对应命令的执行结果，发布到指定 Topic；服务器端订阅获知设备执行命令结果，判断是否发送下一条命令；
8. 服务器端订阅 MQTT 系统主题： "$SYS/brokers/+/clients/+/connected" 和 "$SYS/brokers/+/clients/+/disconnected" ，获取设备上线、离线通知；

**通讯流程图：**




# 2. MQTT 服务器部署

- 选择使用 EMQX 服务器，开源社区使用量庞大，拥有大量实践解决方案。
- EMQX 官网地址：*https://docs.emqx.io*
- 选择开源版本，EMQ X 开源版本，用户可以免费下载和使用。

## 2.1 MQTT 的使用

- 使用 MQTT，仅借助 MQTT 用于数据传输，数据的持久化，由应用层处理，减少对于 MQTT 的依赖。
- MQTT 协议设计的目的是为了解决高并发、低流量问题，而不是持久化。

## 2.2 修改系统主题订阅权限

**用于监听设备离线在线情况，默认是不开放的**

修改配置文件： /etc/emqx/acl.conf

增加以下配置：

```
#所有客户端有权限：
{allow, all, subscribe, ["$SYS/brokers/+/clients/#"]}

#只有指定的服务器 IP 有权限：
{allow, {ipaddr, "127.0.0.1"}, subscribe, ["$SYS/brokers/+/clients/#"]}
```

修改配置后，重启： service emqx restart

### 2.3 创建 HTTP API 应用

登录 EMQ 控制台，默认端口 18083，在控制台中创建应用提供秘钥给 HTTP API 调用

可在服务器使用定时任务每 5 分钟或 10 分钟通过 http api 获取所有客户端信息和服务器数据库数据进行比对校准；

接口文档： https://docs.emqx.com/zh/enterprise/v4.4/advanced/http-api.html

调用：/api/v4/clients 接口，返回集群下所有客户端的信息，支持分页。

# 3. 通讯协议

## 3.1 设备鉴权连接【TCP】

#### 3.1.1 设备 TCP 请求服务器

```
{
    "SN":"1234567890",
    "resource":"connection",
    "commType":"MQTT",
```

```
    "timestamp": "1408636833",
    "data":
    {
        "version":"2.0.0",
        "versionCode":"22",
        "ipAddress":"192.168.10.198", //内网 IP 地址
        "publicIpAddress" : "218.18.166.142"     //公网 IP 地址
        "systemData": {
            "systemVersion" : "11.3",   //操作系统或镜像版本
            "sdCardSize" : 2048,     //SD 卡大小【单位：KB】
            "memorySize" : 0,      //内存大小【单位：KB】
            "cameraCount" : 2,      //相机总数
            "faceSdkVersion" :"1.3",     //人脸 SDK 版本
            "faceSdkStatus" : 0,     //人脸 SDK 状态【0：未激活；1：已激活】
         }
        "errorData": [{
            "code": 1, //错误编码:【1.表数据异常】
            "data": ["users", "cards"]
        }]
    }
}
```

**上传参数**

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| SN | String | 是 | 设备序列号 | 1234567890 |
| resource | String | 是 | 资源类型 | connection |
| commType | String | 否 | 通讯协议类型（TCP/MQTT/HTTP），设备当前通讯协议类型，如需更换通讯 | TCP |

| | | | | |
|---|---|---|---|---|
| | | | 类型，服务器返回数据中可对 commType 进行设置 | |
| timestamp | int | 是 | 时间戳（秒） | 1408636833 |
| data | String | 是 | 数据内容 | 116.397428 |
| version | String | 是 | 版本号 | 2.0.0 |
| versionCode | String | 是 | 版本 CODE | 22 |
| ipAddress | String | 是 | ip 地址 | 192.168.31.96 |
| errorData | String | 否 | 错误数据【用于上传设备启动时发现系统异常上传错误信息】 | |

### 3.1.2 服务器验证设备请求

**提醒：需要在返回内容增加换行符 \r\n 返回，设备以\r\n 为单个数据包的结束标志进行解析。**

JAVA 代码示例：ctx.channel().writeAndFlush(JSONUtil.toJsonStr(tcpJson) + "\r\n");

**1）未授权返回，设备端等待 10 秒重新发起连接**

```
{
    "ret":-1037,
    "message":"Device not authroized"
}
```

**2）验证码不正确返回数据，设备端等待 10 秒重新发起连接**

```
{
    "ret":-1037,
    "message":"Device not authroized"
}
```

**3） 验证通过，返回**

```
{
```

```
    "ret":0,
    "message":"OK",
    "data": {
        "commType":"MQTT",
        "mqttAddr":"192.168.1.10",
        "mqttPort":1883,
        "mqttAccount": "client",
        "mqttPwd": "pwd123",
        "clientID": "Factory/SN",
        "subscribeTopic": "Factory/SN",
        "dataPublishTopic": "DevicePublishTopic/1",
        "rtDataPublishTopic": "DevicePublishTopicRT/2"
    }
}
```


**返回参数**

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| commType | String | 通讯协议类型（TCP/MQTT/HTTP），设备将根据返回的协议类型进行切换 |
| mqttAddr | String | MQTT 服务器地址，IP 或者域名 |
| mqttPort | int | MQTT 服务器连接端口 |
| mqttAccount | String | 登录账号 |
| mqttPwd | String | 登录密码 |
| clientID | String | 设备连接 MQTT 服务器的 clientid，格式：厂家/设备 ID |
| subscribeTopic | String | 设备需要订阅的 Topic |
| dataPublishTopic | String | 设备发送非实时响应数据发布 Topic |
| rtDataPublishTopic | String | 设备发送实时响应数据发布 Topic |

**DataPublishTopic 主题发布内容**

| 序号 | 数据内容 | 示例 |
| --- | --- | --- |
| 1 | 开门记录 | |
| 2 | 设备状态 | |

**RTDataPushTopic 主题发布内容**

| 序号 | 功能 | 示例 |
| --- | --- | --- |
| 1 | 在线密码验证 | |
| 2 | 在线二维码验证 | |
| 3 | 呼叫房间 | |
| 4 | 呼叫管理处 | |
| 5 | 订阅命令执行结果 | |

### 3.1.3 服务端连接示例

以下为 java 语言 netty TCP 代码示例

```
/**
 * TCP 服务端
 */
@Component
public class TcpServerDemo implements ApplicationRunner, ApplicationListener<ContextClosedEvent> {

    @Value("${netty.tcp.port}")
    private int port;

    @Value("${netty.tcp.ip}")
    private String ip;

    @Autowired
    TcpDataHandlerAcc tcpDataHandlerAcc;

    private Channel serverChannel;

    private final EventLoopGroup bossGroup = new NioEventLoopGroup(1);
    //处理 hadnler 的工作线程，其实也就是处理 IO 读写 。线程数据默认为 CPU 核心数乘以 2
    private final EventLoopGroup workerGroup = new NioEventLoopGroup(16);

    private static final Logger LOGGER = LoggerFactory.getLogger(TcpServerDemo.class);

    public void run(ApplicationArguments args) throws Exception {
        ServerBootstrap serverBootstrap = new ServerBootstrap();
        serverBootstrap.group(bossGroup, workerGroup);
        serverBootstrap.channel(NioServerSocketChannel.class);

        //标识当服务器请求处理线程全满时，用于临时存放已完成三次握手的请求的队列的最大长度
```

```
            serverBootstrap.option(ChannelOption.SO_BACKLOG, 4096);//原为 2048

            serverBootstrap.childHandler(new ChannelInitializer<SocketChannel>() {
                @Override
                protected void initChannel(SocketChannel socketChannel) throws Exception {
                    ChannelPipeline pipeline = socketChannel.pipeline();
                    //netty 基于分割符的自带解码器，根据提供的分隔符解析报文，这里是 0x7e;1024 表示单条消息的最大长度，解码器在查找分隔符的时候，达到该长度还没找到的话会抛异常
                    pipeline.addLast(new DelimiterBasedFrameDecoder(2048, Unpooled.wrappedBuffer("\n".getBytes())));

                    //自定义 Hadler
                    pipeline.addLast(new StringDecoder(CharsetUtil.CHARSET_UTF_8), new StringEncoder(CharsetUtil.CHARSET_UTF_8));
                    pipeline.addLast(tcpDataHandlerAcc);
                }
            });

            //绑定端口后，开启监听
            Channel channel = serverBootstrap.bind(port).sync().channel();
            this.serverChannel = channel;
            if (channel.isActive()) {
                LOGGER.info("TCP 服务启动成功：" + port);
            }
        }
        public void onApplicationEvent(ContextClosedEvent event) {
            if (this.serverChannel != null) {
                this.serverChannel.close();
            }
            LOGGER.info("TCP 服务停止");
        }
}


/**
 * TCP 数据处理
 */
@ChannelHandler.Sharable
@Component
@Slf4j
public class TcpDataHandlerDemo extends SimpleChannelInboundHandler<String> {

    @Override
    protected void channelRead0(ChannelHandlerContext ctx, String msg) throws Exception {
        try {
            InetSocketAddress ipSocket = (InetSocketAddress) ctx.channel().remoteAddress();
            String clientIp = ipSocket.getAddress().getHostAddress();
            log.info("客户端 ip 地址：{}", clientIp);
        } catch (Exception e) {
            log.error("获取客户端 ip 异常", e);
        }
```

```
        boolean isJson = JSONUtil.isJson(msg);

        try {
            String devSn = "";
            JSONObject dataJson = null;
            if (isJson) {
                dataJson = JSONUtil.parseObj(msg);
                 devSn = dataJson.getStr("SN");

                 if(StrUtil.isEmpty(devSn)){
                     devSn = formatDevSn(devSn);//去除前缀
                     log.info("拿不到序列号:" + devSn + "断开链接,msg 内容:{}", msg);
                     ctx.disconnect();
                     return;
                 }
            } else {
                log.info("接收数据格式错误断开链接,msg 内容:{}", msg);
                 ctx.disconnect();
                 return;
            }

            ThreadStore.set(IotThreadDataDto.builder().devSn(devSn).build());
            String id = ctx.channel().id().toString();
            log.info("TCP 收到信息{}:{}", msg, id);

            receiveData(devSn, ctx, dataJson);

        } catch (Exception e) {
            log.error("TCP 接收数据出现异常：" + msg, e);
        }
    }

    //解析数据
    private static void receiveData(String devSn, ChannelHandlerContext ctx, JSONObject dataJson){

     // TODO
     Device device = null;// db.getBySn(devSn);   //从缓存或数据库查询设备是否存在
     if (device == null) {//不存在，返回设备未授权
            TcpJsonResultDto tcpJsonResultDto = new TcpJsonResultDto();
            tcpJsonResultDto.setRet(-1037);
            tcpJsonResultDto.setMessage("Device not authroized");
            ctx.channel().writeAndFlush(JSONUtil.toJsonStr(tcpJsonResultDto) + "\r\n");
            return;
        } else {

           //解析 json，获取设备信息，更新到数据库或缓存，如 IP 地址，版本号等
           // TODO
           //updateDeviceInfo(device);
```

```
        }

        //发送 MQTT 服务信息给设备
        sendConnectionData(devSn, ctx);
    }

    //发送 MQTT 服务信息给设备
    private static void sendConnectionData(String devSn, ChannelHandlerContext ctx){

     MqttConnevtResultDto mqttConnevtResultDto = new MqttConnevtResultDto();

     mqttConnevtResultDto.setCommType("MQTT");
     mqttConnevtResultDto.setMqttAddr("mqttHost");
        mqttConnevtResultDto.setMqttPort(1883);
        mqttConnevtResultDto.setMqttAccount("username");
        mqttConnevtResultDto.setMqttPwd("password");
        mqttConnevtResultDto.setClientID(devSn);
        mqttConnevtResultDto.setSubscribeTopic("cloud_subscribe_topic_" + devSn);
        mqttConnevtResultDto.setDataPublishTopic("rt_data_publish_topic_1");//需改为随机获取
        mqttConnevtResultDto.setRtDataPublishTopic("data_publish_topic_1");//需改为随机获取

        TcpJsonResultDto tcpJsonResultDto = new TcpJsonResultDto();
        tcpJsonResultDto.setRet(0);
        tcpJsonResultDto.setMessage("OK");
        tcpJsonResultDto.setData(mqttConnevtResultDto);
        //需要在返回时增加换行符，
     ctx.channel().writeAndFlush(JSONUtil.toJsonStr(tcpJsonResultDto) + "\r\n");
    }

     // 格式化序列号
     private static String formatDevSn(String devSn) {
            if (StrUtil.isEmpty(devSn)) {
                    return devSn;
            }

            // 所有带 V 开头的序列化 只截取后 10 位数
            if (devSn.startsWith("V")) {
                    return StrUtil.subSufByLength(devSn, 10);
            }

            return devSn;
     }

    @Override
    public void channelInactive(ChannelHandlerContext ctx) throws Exception {
        super.channelInactive(ctx);
        log.info("{}链接断开：{},{}", ctx.channel().remoteAddress(),ctx.channel().id());
    }
```

```
    @Override
    public void channelActive(ChannelHandlerContext ctx) throws Exception {
        super.channelActive(ctx);
        ThreadStore.set(IotThreadDataDto.builder().devSn(null).build());
        log.info("链接创建：{}", ctx.channel().remoteAddress());
    }

}

@Data
@NoArgsConstructor
@AllArgsConstructor
public class TcpJsonResultDto {
    private Integer ret;
    private String message;
    private MqttConnevtResultDto data;
}

@Data
public class MqttConnevtResultDto {

    private String commType;
    private String mqttAddr;
    private Integer mqttPort;
    private String mqttAccount;
    private String mqttPwd;
    private String clientID;
    private String subscribeTopic;
    private String dataPublishTopic;
    private String rtDataPublishTopic;
}
```

## 3.2 MQTT 连接示例

### 3.3.1 服务端连接示例

```
/**
     * MQTT 连接器选项
     */
    @Bean
```

```
    public MqttConnectOptions getOptions() {
        MqttConnectOptions options = new MqttConnectOptions();
        options.setServerURIs(host.split(","));//tcp://*.*.*.*:port
        options.setCleanSession(false);
        options.setUserName(username);
        options.setPassword(password.toCharArray());
        options.setConnectionTimeout(timeout);//1000
        options.setKeepAliveInterval(keepalive);//10
        options.setMaxInflight(10000);
        options.setAutomaticReconnect(true);
        return options;
}
/**
     * MQTT 消息订阅绑定（消费者）
     */
    @Bean
    public MessageProducer inbound() {
        // 可以同时消费（订阅）多个 Topic
        MqttPahoMessageDrivenChannelAdapter adapter = new
MqttPahoMessageDrivenChannelAdapter(
                configuration.getClientId() + "_in_"+ BusinessUtil.getIpAndPortAndServerId(),
                configuration.mqttClientFactory());
        adapter.setCompletionTimeout(5000);
        adapter.setConverter(new DefaultPahoMessageConverter());
        adapter.setQos(2);
        //订阅设备非实时响应数据，多个主题防止拥堵
        for (int i = 0; i <= 10; i++) {
            adapter.addTopic("$share/iot/"+configuration.getDataPublishTopic() + (i+1), 2);
        }
```

```
        //阅设备实时响应数据，多个主题防止拥堵
        for (int i = 0; i <= 20; i++) {
            adapter.addTopic("$share/iot/"+configuration.getRtDataPublishTopic() + (i+1), 2);
        }
        //订阅系统主题 上线
adapter.addTopic("$share/iot/"+"$SYS/brokers/+/clients/+/connected", 2);
        //订阅系统主题 下线
adapter.addTopic("$share/iot/"+"$SYS/brokers/+/clients/+/disconnected", 2);
        // 设置订阅通道
        adapter.setOutputChannel(mqttInboundChannel());
        return adapter;
    }
```

### 3.3.2 设备端连接示例

设备端使用 MQTT 示例（注意：下文绿色字体为变量参数，非固定字符串，使用 TCP 鉴权返回参数值）

```
// clean_session 设置为 False，需要保留会话
client = mqtt.Client(clientID, clean_session=False)
// 设置账号、密码验证
client.username_pw_set(mqttAccount, mqttPwd)
// 设置遗嘱，Topic 固定为："Device/death"，内容上传 clientID，qos 设置为 2
client.will_set("Device/death", payload=clientID, qos=2, retain=True)
client.on_connect = on_connect
// 连接，设置连接心跳时间：60s
client.connect(mqttAddr, mqttPort, 60)
// MQTT 连接结果回调
def on_connect(client, userdata, flags, rc):
client.subscribe(subscribeTopic, qos=2) // 订阅
```

## 3.3 协议规范设计【重要】

说明：使用 json 数据格式、UTF8 编码，设备对每条下发的命令不论成功失败都需要上传对应执行结果

通用命令下发报文格式：

```
{
      "commandid":2,
      "resource":"***",
      "operation":"***",
     "data": {}或[] //不同命令不一样
}
```

通用命令响应报文格式：

```
{
     "SN":"1234567890",
     "resource":"commands/result",
     "status":{
           "commandid":1,
           "result":0,   // 0 成功，其它为失败，具体值参考底部 -->错误码定义
           "message":"OK"
     },
     "data": {}或[] //不同命令不一样
}
```

通用响应报文字段说明：

| 名称 | 类型 | 是否必传 | 描述 |
| --- | --- | --- | --- |
| commandid | Integer | 是 | 命令 ID |
| result | Integer | 是 | 命令结果【0 成功，其它为失败，具体值参考底部 -->错误码定义】 |
| message | String | 是 | 对命令结果的文本描述内容 |
| data | Object | 否 | 返回数据（部分接口无返回数据） |

## 3.4 定时上传心跳及同步时间【上行】

说明：设备开机时上传或联网方式变动时立即上传，另定时 2 个小时或 24 小时上传一次（嵌入式设备 2 小时，安卓设备 24 小时），也可手动请求上传心跳数据，命令参考【设备控制->上传心跳】指令

提醒：服务端收到 heartbeat 心跳消息后必须要进行回复，否则客户端会认为服务器的 mqtt client 异常，重新进行 tcp 连接

**上传**

```
{
    "SN":"1234567890",
    "resource":"heartbeat",
    "data":{
        "bleVersion":"1.0.0"
        "simId":"89860422131970157956",
        "simNumber":"861193044335470",
        "netWorkType":1,
        "rssi":100,
        "simOperators":1,
        "mac":"1a:dd:04:87:bc:54", //网卡 MAC 地址
        "ipAddress":"192.168.10.198", //内网 IP 地址
        "publicIpAddress" : "218.18.166.142"    //公网 IP 地址
        "tableData": {//提醒：联网方式变动时不会传
            "usersCount" : 0,    //用户数
            "cardsCount" : 0,    //用户卡数
            "facesCount" : 0,    //用户人脸数
            "successFacesCount" : 0,    //用户注册成功人脸数
            "roomsCount" : 0,    //房间数
            "advertiseCount" : 0,    //广告数
            "announcementCount" : 0,    //公告数
            "timeschedulesCount" : 0,    //广告时间段数
```

```
            "holidaysCount" : 0,      //广告节假日数
            "timePassageGroupCount" : 0,     //通行时间策略数
            "userTimePassageGroupCount" : 0,      //通行时间策略用户关联数
            "eventsCount" : 0     //事件记录数
        },
        //提醒：联网方式变动时不会传
        "systemData": {//系统数据
            "faceSdkStatus" : 0,     //人脸 SDK 状态【0：未激活；1：已激活】
            "sdCardFreeSize" : 1024,     //SD 卡可用大小【单位：KB】
            "memoryFreeSize" : 1024,     //内存可用大小【单位：KB】
        },
        //提醒：联网方式变动时不会传
                "flowData": {//流量数据
            "date" : "2023-03-24",     //昨日日期，服务器更新日期进行流量覆盖存储
                        "flow" : 8955466,     //昨日流量【单位：字节】
        },
        //提醒：联网方式变动时不会传
        "wmpfData": {//微信小程序硬件框架数据
            "isSupportWmpf" : 0,     //是否支持微信小程序硬件框架【0：不支持，1：支持】
            "isInstallWmpfApk" : 0,     //是否已安装 WMPF 的 APK【0：未安装，1：已安装】
            "wmpfApkVersion" : "2.1.0.1565",     //WMPF 的 APK 版本【用于是否需要进行升级判断】
                        "deviceId" : "556664325",     //设备 ID
            "pushToken" : "656SDFg3456786786789889",     //设备消息令牌，用于微信小程序给设备发送消息【门口视频使用】
        }
    }
}
```

**上传参数**

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |

| | | | | |
|---|---|---|---|---|
| SN | String | 是 | 设备序列号 | 1234567890 |
| resource | String | 是 | 资源类型 | connection |
| data | String | 是 | 数据内容 | 116.397428 |
| bleVersion | String | 是 | 蓝牙版本 | 1.0.0 |
| simId | String | 是 | 物联网卡号 | 89860422131970157956 |
| simNumber | String | 是 | 物联网模块 ID | 861193044335470 |
| netWorkType | int | 是 | 网络类型（0： 无，1：NB、2：2G、3：4G、4：ETHERNET、5：WIFI、6：LORA） | 1 |
| rssi | int | 是 | 信号强度 | 100 |
| simOperators | int | 是 | 卡运营商（0: 无， 1：China Mobile、2：China Unicom、3：<br>China Telecom、200：Other） | 1 |
| mac | String | 是 | MAC 地址 | 1a:dd:04:87:bc:54 |
| ipAddress | String | 是 | 内网 ip 地址 | 192.168.10.198 |
| publicIpAddress | String | 否 | 公网 IP 地址 | 218.18.166.142 |

**返回**

```
{
    "resource":"heartbeat/result",
    "ret":0,
    "message":"OK",
    "data": {
        "time": 1636119926,
        "utcTime":1636091126
    }
}
```

**返回参数**

| **参数** | **类型** | **描述** |
|---|---|---|
| time | long(长整形) | 时区秒数时间戳，在 0 时区时间戳的基础增加或减少对应时区差额时间，如北京时间：1636091126+8*60*60 |

| | | |
|---|---|---|
| utcTime | long(长整形) | 0 时区秒数时间戳，安卓设备支持时区，如北京时间 1636091126 |

## 3.5 设备参数设置【下行】

### 3.5.1 更新参数

**服务器发送指令到 MQTT 服务器，指定 topic 发布，以下为部分示例，更多参数说明请参考下方—>【设备参数说明】**

```
{
    "commandid":1,
    "operation":"PUT",
    "resource":"params",
    "data":
     {
         "door1SupperPassword":"123456",
         "door1OpenDuration":"5",
         "door1DetectorDuration":"15"
    }
}
```

**设备端订阅到数据，发布执行结果**

```
datas = {
    "SN":"1234567890",
    "resource":"commands/result",
    "status":
    {
         "commandid":1,
         "result":0,
         "message":"OK"
    }
```

```
}
client.publish(dataPublishTopic, payload=datas, qos=2)
```

### 3.5.2 获取参数

**服务器发送获取参数指令 MQTT 服务器**

```
{
    "commandid":2,
    "operation":"GET",
    "resource":"params",
    "data":
    [
        "door1SupperPassword",
        "door1ForcePassword",
        "door1OpenDuration",
        "door1DetectorDuration"
    ]
}
```

**设备端更新命令执行状态**

```
datas = {
    "SN":"1234567890",
    "resource":"params",
    "status": {
        "commandid":2,
        "result":0,
        "message":"OK"
    },
    "data": {
        "door1SupperPassword":"123456",
```

```
            "door1ForcePassword":"",
            "door1OpenDuration":"5",
            "door1DetectorDuration":"15"
        }
}
client.publish("DevicePublishTopicRT/2",, payload=datas, qos=2)
```

## 3.6 设备控制【下行】

| 序号 | 功能 | 命令 | 备注 |
| --- | --- | --- | --- |
| 1 | 远程开门 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=on&time=5","data":{<br>"floor":"0000000000000001",<br>"eleKeyMode":1,<br>"userId":"1", // 用户 ID【用于开门记录回传】<br>"scene":0 //场景:0：普通，1：对讲<br>"callId":"呼叫方标识_2111191417256"<br>}} | 1.resource 中数字代表所对应的对象的编号,数字 1 为对应门 id,当 id 为 0 时,则对应所有的门<br>2.time 开门时长对应的单位为秒<br>3.最大值是该设备门总个数,id<br>4. 开门时，如果有下发楼层权限，则需要添加 floor 字段，内容是楼层开放的十六进制数据（在事件上传时使用 number 字段传递记录），eleKeyMode 为梯控按键模式：1：直达，2：手选，默认不传时为直达<br>5. scene 为场景字段，0：普通，1：对讲，未传时可默认 0，用于上传事件类型时判断【远程开门，对讲开门】<br>6. callId 呼叫 ID【在上传对讲开门事件时赋值到 number 字段中，如果 scene 为对讲 但是没有 callId 字段或 callId 为空，传递当前通话中的呼叫 ID】 |
| 2 | 开启门禁常开 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=openKeepOn"} | 该指令下发后设备将保持常开，以及设备将上传 type 值为 125 的远程开门常开事件 |
| 3 | 关闭门禁常开 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=closeKeepOn"} | 当开启门禁常开后，需要关闭时，可通过该指令进行关闭恢复，以及设备将上传 type 值为 9 的远程关门事件 |

| | | | |
|---|---|---|---|
| 4 | 开启门禁常闭 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=openKeepOff"} | |
| 5 | 关闭门禁常闭 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=closeKeepOff"} | |
| 6 | 开启梯控消防模式 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=openEleFireMode"} | 开启梯控消防模式，梯控进入消防模式，所有楼层常亮（消防是应急用的，将会全部楼层开放，直至发送关闭指令为止，有断电记忆功能） |
| 7 | 关闭梯控消防模式 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/lock/status?value=*closeEleFireMode*"} | 关闭梯控消防模式，梯控进入正常模式（消防是应急用的，将会全部楼层开放，直至发送关闭指令为止，有断电记忆功能） |
| 8 | 取消门报警 | {"commandid":1,"operation":"PUT","resource":"device/doors/1/alarm?value=off"} | |
| 9 | 重启设备 | {"commandid":1,"operation":"PUT","resource":"device/control?cmd=reboot"} | 设备断电重启 |
| 10 | 重新连接 | {"commandid":1,"operation":"PUT","resource":"device/control?cmd=reconnect"} | 设备重新连接到中心服务器（之前 TCP 协议使用的是 reconnet） |
| 11 | ~~同步时间（废弃）~~ | {"commandid":1,"operation":"PUT","resource":"params","dateTime":"2020-04-17 11:20:00"} | 校正设备时间 |
| 12 | 上传心跳 | {"commandid":1,"operation":"PUT","resource":"device/control?cmd=heartbeat"} | 目前主要用于同步时间，设备时区变更时及时给设备同步时间 |
| 13 | ping 地址 | {"commandid":1,"operation":"PUT","resource":"device/control?cmd=ping","data":{"ip":"127.0.0.1","count": 5}} | ping ip 或域名，主要用于测试网络延迟情况，需在 data 中返回 ping 的数据（多套一个 data），如："data": {"data": "182.61.200.7: icmp_seq=1 ttl=50 time=11.1 ms 64 bytes from \n 182.61.200.7: icmp_seq=1 ttl=50 time=11.1 ms 64 bytes from"} |
| 14 | 固件升级 | {"commandid":1,"operation":"PUT","resource":"files/firmware","data":{ "url":"http://serveraddr/files/firmware/upgrade.zip"} } | 升级设备固件程序 |
| 15 | 扩展板蓝牙升 | {"commandid":1,"operation":"PUT","resour | 用于 RD02 等系列蓝牙固件升级 |

| | | | |
|---|---|---|---|
| | 级 | ce":"files/bleFirmware","data":{<br>"version": "30"<br>"url":"http://serveraddr/files/firmware/upgrade.zip"}<br>} | version 字段为版本号，设备根据该版本号判断是否和自身版本号相同，相同则不进行升级 |
| 15 | WMPF 更新 APK | {"commandid":1,"operation":"PUT","resource":"files/wmpfApkUpdate","data":{<br>"url":"http://serveraddr/files/wmpf.apk",<br>"version": "2.1.0.2556"<br>}<br>} | 用于 WMPF 小程序 APK 下载更新<br><br>设备需判断当前升级版本是否和设备本身版本一致，不一致时才进行下载和升级，避免重复下载 |
| 16 | 修改服务器地址 | {<br>"commandid":1,<br>"operation":"PUT",<br>"resource":"params",<br>"data":{<br>"YunServer":"192.168.1.2:8019",<br>"commType":"MQTT"<br>}<br>} | 修改设备连接的 TCP 服务器地址 |
| 17 | 进入房间（门口视频） | {"commandid":1,"operation":"PUT","resource":"device/doors/call/enter_room?room_id=1000911789&sip_mode=1&name=大门&sn=2502056048", "data": {<br>"callId": "1234546_65566"<br>"captureImage":"http://***/123.jpg",<br>"trtcUserId":400001832,<br>"trtcFromUserId":400001833,<br>"trtcSign": "eJw1jsEKgpE....."<br>}} | 进行视频对讲<br>room_id 和 sip_mode 需替换 sip_mode<br>对讲模式：0：DMVPHONE；1：TRTC<br>name：门口机名称（H810）<br>sn：门口机序列号（H810）<br>captureImage：抓拍图片<br>callId: 呼叫 ID，需在结束通话时上传<br>trtcUserId: TRTC 用户 ID<br>trtcSign: TRTC 签名<br><br>注：DMVPHONE 模式下用于门口机呼叫室内机，管理机时在设备接听页面显示抓拍图片 |
| 18 | 模拟呼叫房间 | {"commandid":1,"operation":"PUT","resource":"device/doors/call/debug_call_room?building=12&room=101"} | building 为 楼栋编号<br>room 为房间编号 |
| 19 | 告警消息通知 | {"commandid":1,"operation":"PUT","resour | 说明：当前消息仅支持发送给管理机，用 |

| | | | |
|---|---|---|---|
| | | ce":"device/alarmNotify","data":{<br>"id":"123",<br>"title":"燃气报警触发",<br>"content":"1 栋 1 单元 101 室主卧",<br>"status": 0,<br>"type": 1,<br>"devId": "1",<br>"time": "2023-03-27 16:27:12",<br>"icon": "http://123.jpg",<br>"captureImage":"http://456.jpg"<br>}<br>} | 于当室内机发送告警消息时，将通知管理机进行显示通知<br><br>id：记录 ID（必传，唯一 ID，每条记录 ID 不一样）<br>title：标题 （必传）<br>content: 内容（必传）<br>type：类型（必传，1：防拆报警；2：一键报警；3：燃气报警；4：烟雾报警；5：火警报警；6：红外入侵；7：门窗入侵）<br>status：状态（必传，0：触发；1：解除）<br>devId：设备 ID（必传，目前为室内机 ID）<br>time: 时间（必传）<br>icon：图标（非必传）<br>captureImage：抓拍图片（非必传） |
| 20 | 退出房间 | {"commandid":1,"operation":"PUT","resource":"device/doors/call/exit_room", "data": {<br>"callId": "1234546_65566"<br>}} | callId: 呼叫 ID，需要结束通话的通话 ID |
| 21 | 开启 KVS Webrtc | {"commandid":1,"operation":"PUT","resource":"device/doors/call/start_master?type=0<br>} | type: 0 是业主接听开启视频对讲，1 是业主视频监控设备 |

**服务器发送 MQTT 示例**

```
{
    "commandid":1,
    "operation":"put",
    "resource":"device/doors/1/lock/status?value=on&time=5"
}
```

**设备端更新命令执行状态**

```
{
```

```
    "SN":"1234567890",
    "resource":"commands/result",
    "status":{
        "commandid":1,
        "result":0,
        "message":"OK"
    }
}
```

## 3.7 表数据操作【下行】

**说明：以下以用户表进行示例命令操作，其它数据表操作请查看下方【表结构说明】具体表的数据结构。其它表操作和用户表操作的格式保持一致,不同部分为各表的数据结构。**

**备注：数据删除时,必须明确指定表中主键的值,每一个表的主键请参考下方【表结构说明】。**

### 3.7.1 新增/更新【用户示例】

**服务器发送：**

```
{
    "commandid":2,"operation":"PUT","resource":"tables/users",
    "data":  [
            {"userID":111,"cardNo":623531,"Group":3,"startTime":"2012-03-24 14:11:12","stopTime":"2015-03-24 14:11:12","superUser":0 },
            {"userID":911,"cardNo":857851,"Group":3,"startTime":"2012-03-24 14:11:12","stopTime":"2015-03-24 14:11:12","superUser":0}
    ]
}
```

**设备端返回：**

```
{
```

```
    "SN":"1234567890",
    "resource":"commands/result",
    "status":{"commandid":2,"result":0,"message":"OK"}
}
```

**备注：**

- **数据更新时,必须含有主键,每一个表的主键请参考下方【表结构说明】。**
- **新增数据时，需先下发关联表数据，再下发本表数据，如用户和卡，需先用户，再下发卡。**
- **数据较多时，需**进行分批下发，一次最多 50 条。
- **更新其它表时,将蓝色字符串** users **更新为其它表名称即可。**

## 3.7.2 删除数据【用户示例】

特别说明：

1. 下发删除指令的时候，设备将把关联数据一起删掉，比如删除用户，把用户关联人脸，卡，时间策略用户关联表都要一起删掉，删除时间策略把关联的时间策略用户关联表一起删掉，其它的都一样，
2. 可以使用关联字段进行删除，如删除某个用户的所有人脸时，可以使用：
   `{ "commandid": 1482, "operation": "delete", "resource": "tables/faces", "data": [ { "userID": "18074234" } ] }`

### 3.7.2.1 删除指定用户

服务器发送：

```
{
    "commandid":2,
    "operation":"DELETE",
    "resource":"tables/users",
     "data":[{"userID":111},{"userID":121}]
}
```

设备端返回：

```
{
    "SN":"1234567890",
```

```
    "resource":"commands/result",
    "status":{"commandid":2,"result":0,"message":"OK"}
}
```

#### 3.7.2.2 删除全部用户

服务器发送：

```
{"commandid":2,"operation":"DELETE","resource":"tables/users"}
```

设备端返回：

```
{
    "SN":"1234567890",
    "resource":"commands/result",
    "status":{"commandid":2,"result":0,"message":"OK"}
}
```

## 3.8 实时事件【上行】

### 3.8.1 上传事件记录

设备上传

```
{
    "SN":"1234567890",
    "resource":"events",
    "data":
    [
        {
            "eventID":49,
             "userID": "111",
```

```
            "name":"姓名",  //仅刷脸和身份证事件上传
             "number":"123456",//卡号、密码、二维码、房间 ID，身份证号，梯控楼层，faceID，callId
        "doorID":1,
        "type":8,//事件选项 Type 详见文档底部的《门禁事件代码定义》
        "time":"2015-04-28 11:08:30",
             "fileName": "20210825200717-0-1234567890.jpg",  //抓拍图片文件名
             "faceAge": 23,//人脸识别-预估年龄
             "faceGender": 1,//人脸识别-预估性别【1:男；2：女】
             "faceMatchScore": 1,//人脸识别分数【0-100 分，分数越高相似度越高】
              "reason":1,//人脸注册失败记录原因，具体值参考下方事件表中定义
    }
   ]
}
```

字段说明：

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
|---|---|---|---|---|
| eventID | int | 否 | 设备产生的事件 ID,从 1 开始递增,用于服务端判断设备是否重复上传事件 | 111 |
| userID | int | 否 | 用户表 ID，无则传 0 | 123 |
| name | String | 否 | 用户姓名【仅刷脸和身份证事件上传】 | 张三 |
| number | String | 否 | 卡号、密码、二维码、房间 ID，身份证号，梯控楼层，faceID, callId 等数据 | 123456 |
| doorID | int | 否 | 设备门 ID【继电器下标，多门控制器需要，其他一体机可写死传 1】 | 1 |
| type | int | 是 | 事件选项 Type 详见文档底部的《门禁事件代码定义》 | 0 |
| time | String | 是 | 格式：YYYY-MM-DD hh:mm:ss（2011-05-21 14:51:20） | 2015-04-28 11:08:30 |
| fileName | String | 否 | 抓拍图片文件名，图片采用上方 HTTP 上传文件的方式，设备先上传图片再上传记录，文件名称【和文件上传请求中的文件名称保持一致，命名规范：yyyyMMddHHmmss-事件类型-序列号】 | 20210825200717-0-1234567890.jpg |

| | | | | |
|---|---|---|---|---|
| faceAge | int | 否 | 人脸识别-预估年龄 | 23 |
| faceGender | int | 否 | 人脸识别-预估性别【1:男；2：女】 | 1 |
| faceMatchScore | int | 否 | 人脸识别分数【0-100 分，分数越高相似度越高】 | 85 |
| temperature | String | 否 | 人脸温度，仅适用于带测温，并且开启了测温功能的设备 | 36.5 |
| reason | int | 否 | 人脸注册失败原因（type 为 90 时上传）：<br>1、图片格式错误<br>2、图片中未发现人脸<br>3、图片模糊<br>4、人脸角度不符合要求<br>5、人脸面积过小<br>6、图片中存在多张人脸<br>7、图片下载失败<br>8、图片过大【不能超过 720p】 | 2 |

### 3.8.2 上传文件[HTTP]

该接口使用 http(s)协议

服务器会在参数中下发文件服务器地址，以及文件服务器秘钥。设备在上传图片，视频等文件时，通过此接口进行提前上传，在具体事件中传入对应的文件名称即可。主要用于解决 IOT 服务器网络带宽压力。

**请求方式：** POST（form-data）

**请求地址：根据设备参数中下发的** fileServiceUrl 参数

**请求参数：**

| **名称** | **类型** | **必传** | **描述** | **示例值** |
|---|---|---|---|---|
| accessSecret | String | 是 | 访问秘钥【根据网络参数中下发的 fileServiceSecret 参数进行传递】 | e10adc3949ba59abbe56e057f20f8896 |
| devSn | String | 是 | 设备序列号 | 1234567890 |
| type | int | 是 | 事件编号【和上传的事件记录值一致】 | 200 |
| fileName | String | 是 | 文件名称【和事件上传的文件名称保持一致，命名规范：yyyyMMddHHmmss-事件类型-序列号】 | 20210825200717-0-1234567890.jpg |
| sourceFile | File | 否 | 文件对象【抓拍的图片或视频等文件】 | |

| | | | | |
|---|---|---|---|---|
| dateStr | String | 否 | 格式：yyyy-MM-dd，不传取当前服务器日期 | 2022-02-22 |

**请求示例**

```
accessSecret={access_token}
&devSn=1234567890
&name=20210825200717-0-1234567890.jpg
&file=File
```

**响应示例**

```
{
    "code":0,
    "msg":"成功"
}
```

**响应参数**

| 名称 | 类型 | 描述 | 示例值 |
|---|---|---|---|
| code | Integer | 响应结果编码【0，表示成功，非 0 时都表示上传失败】 | 0 |
| msg | String | 响应文本说明（一般为失败原因文本说明） | 成功 |

## 3.8.3 对讲-查询用户

备注：用于当【对讲参数】中的【对讲呼叫方式】为【选呼】时，输入或选定房号/一键呼叫 请求当前指令查询当前可呼用户列表，也可用于室内机呼叫当前房间成员使用。优化体验，设备可以在查询用户只有一人时直接发起呼叫

设备发送：

```
{
  "SN":"1234567890",
  "operation":"findUser",
  "resource":"calling",
  "serial":205074,
  "data":{
```

```
            "building":1,        //楼栋单元号，如果无需输入楼栋编号时可不传
            "room":101,          //房间号，若呼叫管理处楼则栋单元号和房间号均为 0，如果为室内机查询用户呼叫，传-1
    }
}
```

服务器返回结果：

```
{
    "resource":"calling/findUser",
    "serial":205074,        //返回的 serial 需要跟设备请求时发送的 serial 一致
    "data":
    {
        "ret":0,            //0 为验证成功，其他为失败【失败默认 1】
        "msg":"ok",     //结果描述
        "data": {
            "userList": [ //优化体验，当返回用户只有一人时直接发起呼叫，不用选择
                {
                    "userID": 111,      //用户 ID
                    "name": "李*",      //姓名（脱敏）
                    "phoneNumber": "132****5632", //手机号（脱敏）
                }
            ]
        }
    }
}
```

### 3.8.4 对讲-开始呼叫

备注：

设备发送：

```
{
   "SN":"1234567890",
   "operation":"SEARCH",
   "resource":"calling",
   "serial":205074,
   "data":{
                "building":1,          //楼栋单元号
                "room":101,             //房间号，若呼叫管理处楼则栋单元号和房间号均为 0
               "fullName":"1a,2,a", //定制，格式：楼栋，楼层，房间，仅安卓设备支持
                "callingType": 1,  //呼叫类型【1：房间；2：用户手机号；3：设备；4：用户 ID；5：一键呼叫】
               "callingTarget": "userID/phone/序列号", //呼叫目标【呼叫类型为 2 传手机号，为 3 传设备序列号（目前用于室内机和管理机 TRTC 呼叫门口机），为 4 是传用户 ID，当为 2，3，4 时该字段必传】
               "fileName":"20210825200717-0-1234567890.jpg", //图片名称，实际图片使用上方的【上传文件】接口进行传递，用于 APP 和室内机待接听界面显示抓拍图片
               "callingStep": 1 //呼叫步骤【1：设备；2：APP】，备注：当呼叫设备无设备时直呼 APP，不会出现第 2 步骤

   }
}
```

服务器返回结果：

```
{
      "resource":"calling/result",
      "serial":205074,        //返回的 serial 需要跟设备请求时发送的 serial 一致
      "data":
      {
            "ret":0,          //0:成功，房间/用户不存在返回 10400
            "roomId":86418325,        //TRTC 呼叫的房间 ID
            "trtcUserId":86418325,        //TRTC 设备的用户 ID
            "trtcSign": "eJw1jsEKgkAURf9l1mFvnlPME1pIuspFlEGJmpE.....", //TRTC 签名
```

```
        "devSn":"1234567891" //如果是呼叫房间，则返回室内机的序列号，如果是管理处，则返回管理机的序列号
        "ringWaitTime": 20, //响铃等待时间：秒
        "voipAccounts":"000001837432923392332",    //SIP 对讲振铃组号
        "callForwardNum":"138****3425",      //转接号码
        "ipAddress":"192.168.1.20",              //房间里室内机的 ip，或者管理机的 ip
         "videoCode":0,                          //默认为 0，房间里室内机为 H800 则为 2，H801 则为 3，
         "callId":"1234567890_211119141725256",          //呼叫 ID，通话结束时传递该 ID，用于记录本次通话状态，如: 1234567890_211119141725256
         "callUserCount": 2,                             //呼叫用户数，用于判断是否需要等待再进行转接号码或者呼叫室内机
         "voipCount": 10,                                //振铃组对讲账号总数
         "wmpfOpenIds": "openid1,openid2", //呼叫小程序用户的 openId，多个时逗号隔开轮询呼叫
    }
}
```

### 3.8.5 对讲-接通电话（室内机）

备注：用于一人接了，其他终端来电方停止响铃，目前仅 *TRTC* 模式下生效

设备发送:

```
{
  "SN":"1234567890",
  "operation":"answerCall",
  "resource":"calling",
  "serial":205074,
  "data":{
          "callId":"呼叫方标识_2111191417256",     //呼叫 ID，通话结束时传递该 ID，用于记录本次通话状态，如: 1234567890_211119141725256
  }
```

```
}
```

服务器返回结果：

**暂无需返回**

## 3.8.6 对讲-结束通话

设备发送：

```
{
   "SN":"1234567890",
   "operation":"callEnd",
   "resource":"calling",
   "serial":205074,
   "data":{
            "callId":"呼叫方标识_2111191417256",     //呼叫 ID，通话结束时传递该 ID，用于记录本次通话状
                                                      态，如: 1234567890_211119141725256
              "waitTime":15,            //等待时长，单位秒
               "callMode":1,            //通话模式（0：未接听；1：内网接听；2:外网接听；3：电话转接；4：
TRTC；5：IPPBX；6：模拟电话）
               "direction":1,           //通话方向（1：主叫；2：被叫）
            "callTime":60,             //通话时长，单位秒
            "callTarget": "12345567",    //通话对象（DMVphone 时为对讲账号或振铃组号或 IP，其他则传空字
符串）
            "call_result": 1   //呼叫结果（0:拒接  1:未接听  2:接通  3:无响应），说明：拒接存在一呼多的情况，
主叫  可根据呼叫时返回的  callUserCount 和 voipCount  全部拒接时才为拒接
   }
}
```

服务器返回结果：

**暂无需返回**

### 3.8.7 请求远程开门[对讲]

说明：该方法主要用于管理机或室内机对讲时进行远程开门，服务器将判断目标设备是否门禁设备而且启用了梯控联动，然后根据小区【梯控联动模式】配置方式进行按键模式的下发，直连模式则下发直达，组合模式将下发手选

设备发送：

```
{
   "SN":"1234567890",
   "operation":"control",
   "resource":"openDoor",
   "serial":205074,
   "data":{
          "targetDevSn": "1234567891",     //目标设备序列号，二选一，同传时服务器优先取序列号
         "targetSipAccount": 123456,       //目标设备对讲账号，二选一，同传时服务器优先取序列号
   }
}
```

服务器返回结果：

```
{
      "resource":"openDoor/result",
      "serial":205074,     //返回的 serial 需要跟设备请求时发送的 serial 一致
      "data":
      {
            "ret":0,         //0:成功
             "msg":"ok",     //结果描述
      }
}
```

### 3.8.8 请求远程召梯[室内机]

说明：该方法主要用于室内机远程召梯，给启用了梯控联动的门禁设备下发按键模式为直达，当有多个梯控设备时服

务端判断当有一个远程开门成功时及为成功，否则为失败

设备发送：

```
{
   "SN":"1234567890",
   "operation":"control",
   "resource":"callElevator",
   "serial":205074,
   "data":{
        "time":"2015-04-28 11:08:30"       //时间
   }
}
```

服务器返回结果：

```
{
      "resource":"callElevator/result",
      "serial":205074,     //返回的 serial 需要跟设备请求时发送的 serial 一致
      "data":
      {
            "ret":0,        //0:成功; 1:未绑梯控设备；2:设备离线；3:请求超时；4:室内机不存在；-1:其它
             "msg":"ok",     //结果描述
      }
}
```

## 3.9 在线验证【上行】

提醒：设备进行在线验证时应做验证后再次验证时需进行延时处理，建议延时 3 秒，服务器验证结果需要 3 秒内返回

### 3.9.1 在线验证密码

设备发送：

```
{
   "SN":"1234567890",
   "operation":"SEARCH",
   "resource":"password",
   "serial":205074,
   "data":{
      "value":"123456",                          //密码内容
      "time":"2015-04-28 11:08:30"       //时间
   }
}
```

服务器返回结果：

```
{
      "resource":"password",
      "serial":205074,                          //返回的 serial 需要跟设备请求时发送的 serial 一致
      "data":
      {
            "ret":0,                          //0 为验证成功，其他为失败【失败默认 1】
             "msg":"ok",                  //结果描述
             "data": {
                      "userId":"123" //  用户 ID【用于开门记录回传】
                      "type":20//  事件编号【用于开门记录回传】
                     "floor":"FFFFFFFFFFFFFFFF" //  楼层【梯控联动使用】
            }
      }
}
```

## 3.9.2 在线验证二维码

设备发送：

```
{
   "SN":"1234567890",
   "operation":"SEARCH",
   "resource":"qrcode",
   "serial":205074,
   "data":{
       "value":"123456",                        //二维码内容
       "time":"2015-04-28 11:08:30",       //时间
       "fileName":"20240715153931-type-1234567890.jpg" //同(3.9.1)
   }
}
```

服务器返回结果：

```
{
     "resource":"qrcode",
     "serial":205074,                        //返回的 serial 需要跟设备请求时发送的 serial 一致
     "data":
     {
          "ret":0,                          //0 为验证成功，其他为失败【失败默认 1】
           "msg":"ok",                  //结果描述
           "data": {
                    "userId":"1" //  用户 ID【用于开门记录回传】
                    "type":23//  事件编号【用于开门记录回传】
                                                                  "floor":"FFFFFFFFFFFFFFFF" //  楼层【梯控联动使用】
          }
     }
}
```

### 3.9.3 在线验证卡号(含身份证)

说明：该模式仅在设备参数->控制参数->卡验证方式(cardVerifyType) 的值为 8（在线验证卡号）时才触发，默认刷卡或身份证是设备本地验证的

设备发送：

```
{
   "SN":"1234567890",
   "operation":"SEARCH",
   "resource":"cardno",
   "serial":205074,
   "data":{
         "value":"123456",                    //卡号或身份证号
         "type": 1,                    //卡类型  1：IC 卡；2：身份证
         "name": 1,                     //姓名（身份证时需传递）
         "time":"2015-04-28 11:08:30"       //时间
   }
}
```

服务器返回结果：

```
{
      "resource":"cardno",
      "serial":205074,                    //返回的 serial 需要跟设备请求时发送的 serial 一致
      "data":
      {
            "ret":0,                    //0 为验证成功，其他为失败
             "msg":"ok",                //结果描述
             "data": {
                              "userId":"1",  // 用户 ID【用于开门记录回传】
                               "type":23,  // 事件编号【用于开门记录回传】
                                                                  "floor":"FFFFFFFFFFFFFFFF",    // 楼层【梯控
```

```
                                                    联动使用】
                        "screenText": "正在通行，请等待"  // 屏显文本【用于显示提示内容到屏显设备上，目前
                                                    用于电单车刷卡提示】
            }
    }
}
```

## 3.10 设备请求获取数据【上行】

### 3.10.1 WMPF 获取数据

#### 3.10.1.1 WMPF 获取票据

说明:

**上传**

```
{
    "SN":"1234567890",
    "resource":"wmpfSnTicket",
    "operation":"GET",
    "serial":205074,
    "data":{
        "deviceId": "SDFGSD655445",    //WMPF 设备 ID
     }
}
```

**上传参数**

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| faceSdkVersion | String | 是 | 人脸 SDK 版本，后续可能用于区分不同厂商的 SDK | 1.3 |
| identityId | String | 是 | 身份 ID（也叫指纹 ID） | HGGGHJKJGJVBNM |

**返回**

```
{
    "resource":"wmpfSnTicket/result",
    "serial":205074,        //返回的 serial 需要跟设备请求时发送的 serial 一致
    "data": {
        "ret":0,
         "message": "OK",
        "snTicket": "A119-3044335JHGSKLDFDSJFSDIF",
    }
}
```

**返回参数**

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| snTicket | 票据 | A119-3044335JHGSKLDFDSJFSDIF |

#### 3.10.1.2 WMPF 注册结果

**上传**

```
{
    "SN":"1234567890",
    "resource":"wmpfRegister",
    "operation":"result", //  待定
    "data":{
        "result": 0,
        "msg": "success",
     }
}
```

**上传参数**

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| result | Integer | 是 | 授权结果【0：成功；其他：失败】 | 0 |

### 3.10.2 获取设备门磁状态[室内机]

说明：用于室内机中打开设备列表时，显示当前设备的打开还是关闭的状态，另外用于需要推送实时状态的心跳保持机制，室内机在打开页面时每 30 秒推送一次

**上传**

```
{
    "SN":"1234567890",
    "resource":"doorStatus",
    "operation":"GET",
    "serial":205074,
    "data":{
        "isReturnData":1
     }
}
```

**上传参数**

| 名称 | 类型 | 是否必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| isReturnData | Int | 是 | 是否返回数据【0：否；1：是】 | 1 |

**服务器推送**

```
{
    "commandid":2,"operation":"PUT","resource":"doorStatus",
    "data":  [
              {
                "devSn": "8886661230",
                "accRelayStatus":1
              }
    ]
}
```

**返回参数**

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| devSn | String | 设备序列号 |
| accRelayStatus | Int | 门磁状态【-1:未知 0:关闭 1:打开】 |

## 3.10.3 文字转语音[HTTP]

该接口使用 http(s)协议

服务器会在参数中下发文字转语音接口地址。设备在需要进行文字转语音时，通过此接口进行转换下载接口。为了降低请求量和设备流量，设备需在转换后缓存至本地。

**请求方式：** POST（form-data）

**请求地址：根据设备参数->通讯参数中下发的 ttsServiceUrl 参数**

**请求参数：**

| 名称 | 类型 | 必传 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| accessSecret | String | 是 | 访问秘钥【根据网络参数中下发的 fileServiceSecret 参数进行传递】 | e10adc3949ba59abbe56e057f20f8896 |
| devSn | String | 是 | 设备序列号 | 1234567890 |
| text | String | 是 | 合成语音的源文本【不可超过 30 个字，含标点符号】 | 您的物业费还有 5 天即将到期 |
| codec | String | 否 | 返回音频格式，可取值：wav（默认），mp3，pcm 示例值：wav | wav |

**请求示例**

```
accessSecret={access_token}
&devSn=1234567890
&codec=wav
```

**响应示例**

```
{
    "code":0,
    "msg":"成功"
    "data": {
        "audio":
```

"UklGRlR/AABXQVZFZm1OIBAAAAABAAEAgD4AAAB9AAACABAAZGFOYSx9AAD+"
}
}

**响应参数**

| 名称 | 类型 | 描述 | 示例值 |
| --- | --- | --- | --- |
| code | Integer | 响应结果编码【0，表示成功，非 0 时都表示异常失败】 | 0 |
| msg | String | 响应文本说明（一般为失败原因文本说明） | 成功 |
| data | Json | 响应数据内容 | UklGRlR/AABXQVZFZm1OIBAAAAABAAEAgD4AAAB9AAACABAAZGFOYSx9AAD+ |
| audio | String | base64 编码的 wav/mp3 音频数据 | |

# 4. 表结构说明

**注：请注意表格字段的大小写。**

## 4.1 用户表

| TableName | 字段 | 数据类型 | 备注 |
| --- | --- | --- | --- |
| users | userID | 整型 | 用户 ID，主键。 |
| | name | 字符串 | 姓名，带屏设备才下发数据<br>长度：20 |
| | group | 整型 | Group 是多卡开门人员组； |
| | passageMode | 整型 | 通行模式 [0: 不允许通行，1：允许通行]，和开始时间停止时间配套使用 |
| | startTime | 字符串 | 开始时间（格式：YYYY-MM-DD hh:mm:ss，例如:2015-01-23 14:42:40；空值为'0'） |

| | | | |
|---|---|---|---|
| | stopTime | 字符串 | 停止时间（格式：YYYY-MM-DD hh:mm:ss，例如:2015-01-23 14:42:40；空值为'0'） |
| | superUser | 整型 | superUser 是用户是否为管理员（0 普通用户， 1 管理员，默认值 0） |
| | userType | 整型 | userType（0 普通用户 ，1 内部人员，仅进行双重验证时使用，默认 0） |
| | disable | 整型 | 禁用用户/黑名单用户， （0 有效， 1 禁用，默认值 0），禁用后刷卡、人脸等不能开门，上传“黑名单用户”事件 |
| | password | 字符串 | 个人密码（长度，4 到 8 位） |
| | identityNo | 字符串 | 身份证号码，用于身份证开门，设备支持才下发字段 |
| | phoneNumber | 字符串 | 手机号码，用户对讲直接拨打手机号，验证号码，设备支持才下发字段 |
| | access | 字符串 | 门权限值（继电器编号）：1,2,3,4 ，逗号隔开。目前仅门禁控制器使用 |
| | floor | 字符串 | 用户梯控楼层权限（门禁梯控联动，485 输出），16 进制字符串表示，长度 16，例如：FFFFFFFFFFFFFFFF，表示用户拥有 1-64 层权限。授权单层的话是点亮直达，多层是不会点亮，只会授权，需要手选所要达到的楼层 |
| | miMAC | 字符串 | 小米手环 mac 地址，设备支持才下发字段 |

## 4.2 用户卡表

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| cards | userID | 整型 | 用户表主键 ID |
| | cardNo | 整型 | cardNo 最大为 4294967295（2^32 次方） |
| | status | 整型 | 卡状态（1 有效，2 挂失卡，3 禁用卡）<br>挂失卡刷卡，上传“卡已挂失”事件<br>禁用卡，上传“卡已禁用”事件 |

## 4.3 用户人脸表

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| faces | userID | 整型 | 用户表主键 ID |

| | | | |
|---|---|---|---|
| | faceID | 整型 | 人脸 ID，主键，数值唯一 |
| | faceImage | 字符串 | 人脸图片，服务器下发为下载路径 |

## 4.3 用户掌纹表

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| palm_print | userID | 整型 | 用户表主键 ID |
| | palmID | 整型 | 人脸 ID，主键，数值唯一 |
| | palmImage | 字符串 | 人脸图片，服务器下发为下载路径 |

## 4.4 房间表（可视对讲）

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| rooms | roomID | 整型 | 房间 ID，主键 |
| | roomNum | 整型 | 房间编号 |
| | roomName | 字符串 | 房间名称 |
| | empName | 字符串 | 业主名称 |
| | voipAccounts | 字符串 | 对讲振铃组号 |
| | callForwardNum | 字符串 | 转接电话号码 |
| | voipList | 字符串 | |
| | ipAddress | 字符串 | |
| | videoCode | 整型 | |
| | ndstartTime | 字符串 | 免打扰开始时间 |
| | ndstopTime | 字符串 | 免打扰结束时间 |

## 4.5 广告表（带屏设备）

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| advertise | advID | 整型 | 广告 ID，主键 |
| | startDate | 字符串 | 广告开始时间，格式：YYYY-MM-DD |
| | endDate | 字符串 | 广告结束时间，格式：YYYY-MM-DD |
| | videoName | 字符串 | 广告视频文件名称 |
| | videoURL | | 广告视频文件下载路径 |
| | imageNameList | | 广告图片文件名称列表 |
| | imageURLList | | 广告图片下载路径列表 |
| | playInterval | | 广告图片切换时间间隔，单位：秒 |
| | playPriority | | 广告播放优先级，1 为优先播放 |
| | suitableGender | | 广告适合性别，设备广告播放模式为：按性别和年龄播放，使用 |
| | suitableAge | | 广告适合年龄，设备广告播放模式为：按性别和年龄播放，使用 |
| | isDefault | 整型 | 是否默认广告，更换设备出厂图片轮播（0 否，1 是） |
| | bpVideoName | 字符串 | 半屏广告视频文件名称 |
| | bpVideoURL | 字符串 | 半屏广告视频文件下载路径 |
| | bpImageUrlList | | 半屏广告图片下载路径列表 |
| | bpImageNameList | | 半屏广告图片文件名称列表 |

## 4.6 公告表（带屏设备）

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| announcement | annID | 整型 | 公告 ID，主键 |
| | title | 字符串 | 公告标题 |
| | playSpeed | 整型 | 文字滚动播放速度 |
| | content | 字符串 | 公告内容 |
| | startTime | 字符串 | 开始播放时间 |
| | endTime | 字符串 | 停止播放时间 |

## 4.9 广告时间段表

| TableName | 字段 | 数据类型 | 备注 |
| --- | --- | --- | --- |
| timeschedules | tsID | 整型 | 时间段 ID，主键 |
| | sunTime1 | | 星期天时段 1，格式: |
| | sunTime2 | | 星期天时段 2 |
| | sunTime3 | | 星期天时段 3 |
| | monTime1 | | |
| | monTime2 | | |
| | monTime3 | | |
| | tueTime1 | | |
| | tueTime2 | | |
| | tueTime3 | | |
| | wedTime1 | | |
| | wedTime2 | | |
| | wedTime3 | | |
| | thuTime1 | | |
| | thuTime2 | | |
| | thuTime3 | | |
| | friTime1 | | |
| | friTime2 | | |
| | friTime3 | | |
| | satTime1 | | |
| | satTime2 | | |
| | satTime3 | | |
| | hol1Time1 | | |
| | hol1Time2 | | |
| | hol1Time3 | | |
| | hol2Time1 | | |

| | | | |
|---|---|---|---|
| | hol2Time2 | | |
| | hol2Time3 | | |
| | hol3Time1 | | |
| | hol3Time2 | | |
| | hol3Time3 | | |

## 4.10 广告节假日表

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| holidays | hID | 整型 | 节假日 ID，主键 |
| | startDate | 字符串 | 节假日开始时间，格式：YYYY-MM-DD |
| | stopDate | 字符串 | 节假日结束时间，格式：YYYY-MM-DD |
| | type | 整型 | 节假日类型（） |
| | recursYearly | 整型 | 按年循环（0 否，1 是） |

## 4.11 通行时间策略表

规则说明：先查询是否在该时间有常开模式，再根据 userID 查询关联表，如果找到该用户的关联策略，只执行个人关联策略，未找到则查询是否有应用于所有人的策略，另当存在多个策略时间重叠时以不允许通行规则优先

设备实现逻辑说明：

1.如果用户有单独设置通行时间组，则不在执行应用所有人的规则

| **表名** | **time_passage_group** | |
|---|---|---|
| **字段** | **数据类型** | **备注** |
| tpgID | 整型（int） | 主键 ID |
| passageMode | 整型（int） | 通行模式 [0: 不允许通行，1：允许通行，2：常开，3：允许通行（免鉴权）] |
| timeMode | 整型（int） | 时间模式【0：日期；1：星期，2：区间（仅判断第一个时间段）】 |
| weekDays | 字符串 | 星期值【1-7；如：1,2,3,4,5】 |
| startDate | 字符串 | 开始日期，格式：YYYY-MM-DD，如：2021-11-13 |
| endDate | 字符串 | 结束日期，格式：YYYY-MM-DD，如：2021-11-13 |
| startTime1 | 字符串 | 开始时间 1,格式：HH:mm，默认:'00:00' |

| | | |
|---|---|---|
| endTime1 | 字符串 | 结束时间 1,格式：HH:mm，默认:'00:00' |
| startTime2 | 字符串 | 开始时间 2,格式：HH:mm，默认:'00:00' |
| endTime2 | 字符串 | 结束时间 2,格式：HH:mm，默认:'00:00' |
| startTime3 | 字符串 | 开始时间 3,格式：HH:mm，默认:'00:00' |
| endTime3 | 字符串 | 结束时间 3,格式：HH:mm，默认:'00:00' |
| applyAllUsers | 整型（int） | 应用于所有人 [0: 否， 1： 是] |

**数据下发示例**


```
{
    "tpgID":182,
    "applyAllUsers":0,
    "startTime3":"00:00",
    "startTime2":"00:00",
    "startTime1":"00:00",
    "endTime1":"23:59",
    "passageMode":0,
    "timeMode":1,
    "startDate":"",
    "endDate":"",
    "weekDays":"1,3,5,7,2,4,6",
    "endTime2":"00:00",
    "endTime3":"00:00"
}
```


## 4.12 通行时间策略用户关联表

| **表名** | **user_time_passage_group** | |
|---|---|---|
| **字段** | **数据类型** | **备注** |
| tpgID | 整型（int） | 通行时间策略策略 ID |
| userID | 整型（int） | 用户 ID |

**数据下发示例**

```
{
    "tpgID" : 1,
            "userID" : 3,
}
```

## 4.13 门禁事件记录表

| TableName | 字段 | 数据类型 | 备注 |
| --- | --- | --- | --- |
| events | eventID | 整型 | 设备产生的事件 ID,从 1 开始递增,用于服务端判断设备是否重复上传事件 |
| | userID | 整型 | 用户表 ID，无则传 0 |
| | name | 字符串 | 用户名字 |
| | number | 字符串 | 卡号、密码、二维码、房间 ID，身份证号，梯控楼层，faceID 等数据 |
| | doorID | 整型 | 设备门 ID【多门控制器需要，其他一体机可写死传 1】 |
| | type | 整型 | 事件选项 Type 详见文档底部的《门禁事件代码定义》 |
| | time | 字符串 | 格式：YYYY-MM-DD hh:mm:ss（2011-05-21 14:51:20） |
| | fileName | 字符串 | 抓拍图片文件名，图片采用上方 HTTP 上传文件的方式，设备先上传图片再上传记录 |
| | faceAge | 整形 | 年龄 |
| | faceGender | 整形 | 性别 |
| | faceMatchScore | 整形 | 人脸识别分数 |
| | temperature | 浮点数 | 人脸温度，仅适用于带测温，并且开启了测温功能的设备 |
| | reason | 整形 | 人脸注册失败原因（type 为 90 时上传）:<br>1、图片格式错误<br>2、图片中未发现人脸<br>3、图片模糊<br>4、人脸角度不符合要求<br>5、人脸面积过小<br>6、图片中存在多张人脸 |

| | | | |
|---|---|---|---|
| | | | 7、图片下载失败<br>8、图片过大【不能超过 720p】 |

## 4.14 对讲设备表

| TableName | 字段 | 数据类型 | 备注 |
|---|---|---|---|
| video_devices | id | 整型 | 设备 ID |
| | devId | 整型 | 设备 ID |
| | devSn | 字符串 | 设备序列号 |
| | devName | 字符串 | 设备名称 |
| | devType | 整形 | 设备类型【1 管理机；2 小区设备；3 单元设备；4 室内机；6 访客机；7 可视门铃】 |
| | videoCode | 整形 | 视频编码格式【0：无；1：H264；2：MJPEG】 |
| | voipAccount | 字符串 | 设备对讲账号 |
| | ipAddress | 字符串 | 设备 ip |
| | enableEle485 | 整形 | 启用梯控 485 输出【0 不启用；1 启用】 |

# 5. 设备参数说明

### *5.1* 硬件参数

| **属性名称** | **参数** | **读写类型** | **备注** |
|---|---|---|---|
| 序列号 | SN | 只读 | 设备唯一 ID |
| Mac 地址 | MAC | 只读 | |
| 门数量 | doorCount | 只读 | 控制板上控制门的数量 |
| 读头数量 | readerCount | 只读 | 仅指韦根读头的数量 |
| 辅助输入数量 | auxInCount | 只读 | |
| 辅助输出数量 | auxOutCount | 只读 | |
| 外部版本号 | version | 只读 | 外部版本号，即对外发布固件版本 |
| 内部版本号 | versionCode | 只读 | 内部版本号，作为内部功能判断使用，只有变更大功能才会更新内部代码版本号 |

## 5.2 通讯参数

| 属性名称 | 参数 | 数据类型 | 备注 |
| --- | --- | --- | --- |
| 网络模式 | dhcpMode | 整型 | 0：动态；1：静态 |
| IP 地址 | ipAddress | 字符串 | |
| 网关 | gateway | 字符串 | |
| 子网掩码 | netmask | 字符串 | 默认 255.255.255.0 |
| DNS | dns | 字符串 | 默认：8.8.8.8 |
| 服务器地址 | yunServer | 字符串 | 格式： ip:port |
| 文件服务器地址 | fileServiceUrl | 字符串 | 如果事件中存在抓拍图片需要先上传文件再上传事件。具体调用方式请参考【文件上传】接口说明 |
| 文件服务器秘钥 | fileServiceSecret | 字符串 | 文件服务器交互需要用到的秘钥 |
| 文字转语音接口地址 | *tts*ServiceUrl | 字符串 | 在线文字转语音的接口请求地址。具体调用方式请参考【文字转语音】接口说明 |
| 文件服务器秘钥 | *tts*ServiceSecret | 字符串 | 文字转语音接口交互需要用到的秘钥 |

## 5.3 屏显参数

| 属性名称 | 参数 | 数据类型 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| 呼叫房间显示方式 | callRoomShowType | 整形 | 0：输入房号；1：展示房间；2：展示房间+姓名；3：一键呼叫 | 0 |
| 设备型号名称 | devModelName | 字符串 | 显示自定义的型号名称 | VF618Pro |
| APP 二维码 | appQrCodeUrl | 字符串 | 二维码图片地址，必须拼接#1/1/140，最好下发 png 格式图片，jpg 部分新版本格式不支持，仅支持 http 格式，https 部分型号设备不支持，清空则下发空字符串 | http://***.png#1/1/140 |
| 屏显时间时区 | timeZone | 字符串 | 安卓设备根据时区进行转换时间显示，心跳下发的是北京时区，可以转换为对应的时区进行显示在屏幕上 | Asia/Shanghai |
| 服务器时区 | serverTimeZone | 字符串 | 安卓设备的系统时间将采用改时区进行设置，事件上传和时间判断都已该时区为标准 | Asia/Shanghai |

## 5.4 语音参数

语音文件生成可请求上方的【设备请求获取数据->文字转语音】接口进行获取，获取后请缓存至本地，如相关参数变更时需清空缓存重新获取；

| 属性名称 | 参数 | 数据类型 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| 是否启用到期播报提醒 | enableDueVoice | 整形 | 0:不启用；1：启用；说明：此到期时间为用户表的到期时间； | 0 |
| 即将到期提醒天数 | preDueVoiceDays | 整形 | 指提前多少天进行提醒（不可大于 30 天） | 7 |
| 即将到期提醒天数内容 | preDueVoiceContent | 字符串 | 指提前多少天进行提醒的内容（不可超过 22 个字），天数以 ${days} 进行占位替换 | 您的物业费还有${days}天即将到期 |
| 到期后提醒天数（可开门） | afterDueVoiceDays | 整形 | 指到期后多少天进行提醒（还是可以开门，不可大于 30 天） | 7 |
| 到期后提醒天数内容 | afterDueVoiceContent | 字符串 | 指到期后多少天进行提醒的内容（不可超过 22 个字），天数以 ${days} 进行占位替换 | 您的物业费已到期${days}天 |
| 到期后超出提醒天数内容 | outDueVoiceContent | 字符串 | 指超出到期提醒天数后的内容（已不可开门，不可超过 22 个字），如：您的物业费已到期 | 您的物业费已到期 |
| | | | | |
| 开门成功提示 | sucVoiceContent | 字符串 | 开门成功提示语（下发空字符串进行清空，使用默认提示，不可超过 22 个字） | 门已开，请注意关门 |

## 5.5 对讲参数

| 属性名称 | 参数 | 数据类型 | 描述 | 示例值 |
| --- | --- | --- | --- | --- |
| 对讲通讯模式 | sipMode | 整形 | 0：SIP；1：TRTC；2：WMPF | 0 |
| 对讲呼叫方式 | sipCallWay | 整形 | 0：群呼；1：选呼；2：循呼【默认0】,门口机使用 | 0 |

| | | | | |
|---|---|---|---|---|
| Sip 服务器地址 | sipAddr | 读写 | SIP 音视频服务器地址 | 114.55.106.95:55060 |
| SIP 通讯类型 | sipTransportType | 整形 | 0：TCP；1：UDP；【默认 0】 | 0 |
| SIP 账号 | voipAccount | 读写 | 音视频账号 | 123 |
| SIP 账号密码 | voipPwd | 读写 | 音视频账号密码 | 564dfsdfsdsd88wsef |
| WMPF 小程序 APPID | wmpfMiniAppId | 字符串 | 微信小程序 APPID | wx325455asdasa65569 |
| WMPF 小程序型号 ID | wmpfMiniModuleId | 字符串 | 微信小程序上硬件设备中的 model_id | ZSRGPvLISRrqVPyVUEeK9w |
| WMPF 移动应用 APPID | wmpfHostAppId | 字符串 | 微信开放平台上注册的移动应用 ID | wx325455asdasa65565 |
| WMPF 设备类型 ID | wmpfProductId | 整形 | 微信终端合作平台上申请的设备类型 ID | 6454 |
| WMPF 秘钥版本 | wmpfKeyVersion | 整形 | 微信终端合作平台设备类型对应设备型号的秘钥版本 | 1 |
| WMPF 签名 | wmpfSignature | 字符串 | 微信终端合作平台（productId+deviceId+私钥文件）生成每个设备的签名 Signature，要去掉换行符 | sdgg56756756gfdfgdfg45568fbhfghfghf |
| 启用电话转接 | enableCallForward | 整型 | 默认值：1；（0 不启用、1 启用），如果启用 IPPBX 电话网关，则不判断该参数，默认 1 是为了兼容 | 1 |

## 5.6 音量参数

| 属性名称 | 参数 | 数据类型 | 备注 |
|---|---|---|---|
| 广告音量 | volume | 整型 | 设置范围为 0 至 8，为 0 时无音量，默认值：3 |
| 通话音量 | callVolume | 整型 | 设置范围为 0 至 8，为 0 时无音量，默认值：4 |
| 提示音量 | tipsVolume | 整型 | 设置范围为 0 至 8，为 0 时无音量，默认值：3 |

## 5.6 控制参数

| 属性名称 | 参数 | 数据类型 | 备注 |
| --- | --- | --- | --- |
| 设备管理密码 | adminPwd | 整型 | 设备的管理密码，带屏一体机设备 |
| 读头配置 | reader1IOState | 整型 | 0 不配置 1 配置为入 2 配置为出 |
| 设备类型 | devType | 整型 | 1 管理机；2 小区设备；3 单元设备；4 室内机；6 访客机；7 可视门铃； |
| 二维码红外灯 | qrCodeReaderLed | 整型 | 默认值：0；（0 关闭、1 开启） |
| 启用时间策略 | enableTimeStrategy | 整型 | 默认值：0 （0 不启用、1 启用） |
| 门磁报警延时检测时间 | sensorDelayAlarm | 整型 | 单位：秒，用于门磁打开后，多长时间未关上产生报警，默认 15 秒 |
| 广告播放模式 | advPlayMode | 整型 | 默认值：1 （1 循环播放、2 靠近播放、3 按性别和年龄播放） |
| 门禁验证方式 | verifyType | 整型 | 3 仅密码；4 仅卡；5 仅人脸；6 卡或人脸；10 健康码；11 卡+人脸；12 内部人员(人脸) / 外部人员人脸+内部人员(人脸或卡)；<br>13 内部人员(人脸) / 外部人员人脸+内部人员(人脸+卡)；<br>14 内部人员(人脸+卡) / 外部人员人脸+内部人员(人脸或卡)；<br>15 内部人员(人脸+卡) / 外部人员人脸+内部人员(人脸+卡)； |
| 启用梯控 485 输出 | enableEle485 | 整型 | 默认值：0；（0 不启用、1 启用） |
| 开启网络自检功能 | networkSelfTestEnable | 整型 | 默认 1 （0 关闭、1 开启）表示是否需要开启网络自检功能，检测到离线，则重启设备，脱机设备可以关闭。 |
| 支持外置二维码功能 | externalQRCodeEnable | 整型 | 默认值：0（0 不支持、1 支持 ） |
| **卡参数** | | | |

| | | | |
|---|---|---|---|
| 读卡方式 | rfidType | 整型 | 读卡方式：1、仅 A 卡　2、仅 B 卡　3、A 卡或 B 卡 |
| 卡验证方式 | cardVerifyType | 整型 | 卡验证方式：1 仅卡号，2 卡号或扇区秘钥，3 卡号加扇区秘钥，4 CPU 卡；5 超级 SIM 卡；6 CPU 卡或超级 SIM 卡；7 卡号或 CPU 卡或超级 SIM 卡；8 在线验证卡号；9：cpu 卡（不验证密钥）；<br>注意事项：<br>1.CPU 卡，当前仅支持 A 卡，卡验证方式为：CPU 卡，卡类型必须选择：A 卡。<br>2.B 卡，当前仅支持卡号验证，不支持秘钥验证。<br>3.在线验证卡号时需调用【3.9.4 在线验证卡号】接口 |
| 卡扇区 | cardMifareSection | 整型 | 默认值：门禁 1，梯控 2，范围 1~16 |
| 卡扇区秘钥 | cardMifareSecretKey | 字符串 | 16 进制数值，示例：DC26D96DBBCA |
| 卡扇区门编号 | cardDoorNo | 整型 | 默认值：0，范围 0~255 |
| 防复制开关 | disableCopyCard | 整型 | 1 启用防复制，　2 关闭防复制 |
| **韦根输出输入参数** | | | |
| 韦根输出虚拟卡号 | virtualCard | 整型 | 10 进制数据，最大 4 字节 |
| 韦根输出卡号类型 | wgOutType | 整型 | 0 真实卡号，1 固定虚拟卡号 |
| 韦根输出卡号次序 | wgOutOrder | 整型 | 1 正序，2 反序 |
| 韦根输入密码格式 | wgInPwdFmt | 整形 | 0：单键值；1：连续值；【默认单键值，连续值时将和卡号混用，先验证本地卡号，不通过时在进行密码在线验证，密码验证失败时上传为卡未注册事件】 |

## 5.7 门控参数

一体机时取第一个参数，控制器才用到 4 个参数

| **属性名称** | **参数** | **数据类型** | **备注** |
|---|---|---|---|
| 固定开门密码 | door1SupperPassword | 字符串 | 最大 8 位 |
| 闭门回锁 | door1LockOnClose | 整型 | 1 启用<br>0 不启用（门闭合就锁） |
| 门磁类型 | door1SensorType | 整型 | 0 代表无<br>1 代表常开<br>2 代表常闭 |

| | | | |
|---|---|---|---|
| 锁驱动时长 | door1OpenDuration | 整型 | 设置范围（0~255）<br>0 代表常闭<br>255 代表常开<br>1~254 代表开门时长 |
| 门磁超时报警时长 | door1DetectorDuration | 整型 | 设置范围（0~255）<br>单位为秒（s） |
| 出门按键配置 | door1REXInput | 整型 | 0 锁定<br>1 不锁定（出门开关是否有效） |
| 报警延时 | door1REXTimeOut | 整型 | 设置范围（0~255），单位为秒（s）<br>门被锁定后，多长时间才去检测报警 |
| 梯控开门时长 | *elevctlholdtimeout* | 整型 | 设置范围（1~250，单位秒，默认值：5）<br>启用梯控联动时设置的梯控开门时长 |

## 5.8 人脸|人证参数

| **属性名称** | **参数** | **数据类型** | **备注** |
|---|---|---|---|
| 人脸识别显示姓名 | faceShowName | 整型 | 0 关闭 1 启用 |
| 开启活体识别 | liveDetect | 整型 | 1 开启 0 关闭 |
| 开启陌生人脸通行 | strangeFacePass | 整型 | 0 关闭 1 开启 默认关闭 0 【设备】 |
| 人证识别阈值 | faceThreshold11 | 整型 | 默认：50 取值范围 30-100 |
| 人脸识别阈值 | faceThreshold1N | 整型 | 默认：80 取值范围 60-100 |
| 人脸识别框位置 | faceBorderIndex | 整型 | 默认值：2（1 左上角；2 中上位置；3 右上角；4 左下角；5 右下角） |
| 人脸识别框大小 | faceBoderSizeScale | 浮点型 | 0-1 之间 表示识别框的宽占屏幕宽的比例，例如 0.5 表示。0.90-0.99 表示识别框的宽占满屏幕宽度，但不是全屏 |
| 人脸识别距离 | recognitionDistance | 整型 | 默认值：2，单位米，当下发 0 时为 0.5 米 |
| 相同人脸连续开门间隔 | sameFaceInterval | 整型 | 表示相同人脸最短的人脸开门间隔，例如 5 表示同一个人最短的开门时间间隔为 5 秒，这是为了避免同一个人一直站在设备面前设置一直频繁提示门已开 |
| 人证比对通行模式 | identityAccessMode | 整型 | 1 仅人证比对；2 人证比对+授权；3 仅身份证比 |

| | | | |
|---|---|---|---|
| | | | 对；4：仅刷身份证 |
| 人证设备模式 | identityModel | 整型 | 默认值：1（1.支持 IC 卡和身份证模式；2.仅支持身份证模式） |
| 双目活体检测允许的偏差 | binocularDistanceLimit | 整型 | 双目活体检测允许的偏差，默认为 50 |
| 活体算法 | liveDetectMode | 整型 | 默认为 0 单目活体算法(0 单目活体算法 ;1 双目活体算法 ） |
| 人脸识别成功自定义提示语 | faceOpenSuccessTips | 字符串 | 例如：验证成功，欢迎光临 |
| 人脸识别失败自定义提示语 | faceOpenFailTips | 字符串 | 例如：无记录，请联系管理员登记 |
| 人脸识别失败记录上传 | uploadFaceRecoFailImage | 整型 | 默认 0 关闭; (0 关闭;1 启用) |
| 抓拍图片上传 | uploadRecordImage | 整型 | 默认 1 智能模式; ( 0 关闭;1 智能模式 2 启用【智能模式 4G 网络默认不上传，其他网络默认上传】） |
| 删除过期数据 | delExpiredUserEnable | 整型 | 默认 0 关闭；（0：关闭；1：启用） |

# 6. 返回值错误码

## 6.1 返回值说明

1. 每一条请求,返回内容都含有 status 状态,result 代表返回值,message 是对 result 的解释。
2. 设备返回功能“不支持”的错误码，服务端需要删除该条命令，并做好记录。

## 6.2 错误码定义

| 错误码 | 说明 |
|---|---|
| 0 | 成功 |
| 1 | 通信密码错误 |
| 2 | 传入参数错误 |
| 3 | 功能暂且不支持 |

| | |
|---|---|
| 4 | 命令执行错误 |
| 5 | 设备内存不足 |
| 6 | 连接未带 token 或者 token 不匹配或者已失效 |
| 7 | 设备磁盘空间不足 |
| 8 | 门或者辅助输出 ID 设置错误 |
| 9 | 门或者辅助输出时间设置错误,超出有效范围 0-255s |
| 10 | 命令不支持 |
| 11 | 时间格式错误,格式规范:YYYY-MM-DD 空格 hh:mm:ss,例如:2015-01-23 14:42:40 |
| 12 | 数据表的元素类型不匹配,例如时间为字符串,下发为整型 |
| 13 | 数据表缺主键 |
| 14 | 传入数据无效,主键为无效值 |
| 15 | 传入参数不允许设置 |
| 16 | 不支持 wifi 功能 |
| 17 | 无效的 table 表名称 |
| 18 | 已达到最大容量【比如用户数】 |
| 19 | 写入数据失败 |
| 20 | 表不允许更新操作 |
| 21 | 表不允许删除操作 |
| | |
| 99 | 命令解析异常【可能接收的数据不是 json 或者不是一个完整的 json 格式，命令 ID（commandid）传-1，message 上传接收命令前 50 个字符】 |

# 7. 事件类型代码

## 7.1 门禁事件代码定义

代码区间划分

范围区分已不准，仅参考，实际以颜色为准

| 代码范围 | 事件类型 |
| --- | --- |
| 0--39 | 正常事件（绿色） |
| 40--79 | 警告事件（橙色） |
| 80--119 | 报警事件（红色） |
| 120--165 | 门状态事件（蓝色） |
| 166-255 | 其他事件（黑色） |

代码定义

| Code | 事件类型 | 说明 |
| --- | --- | --- |
| 0 | 正常刷卡开门 | 指具有开门权限的人员刷卡且验证通过后所触发的正常事件 |
| 4 | 固定密码开门 | 指使用在当前门上设置的固定密码（也称为超级密码）开门且验证通过后所触发的正常事件 |
| 7 | 取消报警 | 当用户对存在报警的门进行了取消报警操作,且取消报警成功时,触发该正常事件 |
| 8 | 远程开门 | 当用户对某个门进行远程开门操作且开门成功时,触发该正常事件 |
| 19 | 手机蓝牙开门成功 | |
| 20 | 临时密码开门成功 | |
| 21 | 人脸识别开门成功 | |
| 22 | 人证识别开门成功 | |
| 23 | 二维码开门成功 | |
| 24 | 指纹开门 | |
| 25 | 离线二维码开门 | |
| 26 | 健康码开门成功 | 需在事件上传中增加 extraData 附加数据健康码信息 |
| 27 | 个人二维码开门 | |
| 28 | 超级 sim 卡开门成功 | |
| 30 | 钥匙开门 | 目前仅门锁设备会上传 |
| 31 | 掌纹开门成功 | |
| 32 | 鲁通码开门 | 该事件由服务器产生 |

| | | |
|---|---|---|
| 33 | 陌生人刷脸开门成功 | 用于通行时间组中通行模式为：可通行（免鉴权）方式时产生 |
| 34 | 陌生人刷卡开门成功 | 用于通行时间组中通行模式为：可通行（免鉴权）方式时产生 |
| 35 | 对讲开门 | 设备在服务器下发远程开门指令中进行判断 scene 场景字段值，0：普通，1：对讲,，将在 number 字段中传递 callId，当远程开门指令没有传递 callId 字段或 callId 为空时，则传递当前通话中的呼叫 ID |
| 36 | 转接电话开门 | 当通话模式为 3：电话转接；5：IPPBX 进行上传，将在 number 字段中传递当前通话中的 callId |
| 40 | 刷卡间隔太短 | 当两次刷卡之间的间隔小于该门设定的刷卡间隔时间时,触发该异常事件 |
| 47 | 卡未注册 | 指当前卡号未注册时,所触发的事件 |
| 48 | 门开超时 | 门被打开后超过延时时间后检测门磁,如果门没有关上,触发该事件 |
| 49 | 卡已过有效期 | 当已设置门禁有效时间的人员,在结束门禁日期后刷卡无法验证通过,触发该事件 |
| 50 | 个人密码开门失败 | 设备找到了个人密码，但是验证合法性时不通过，如用户禁用，不在有效期，不在通行时间段内等（原为密码错误：使用卡加密码验证方式或者胁迫密码、紧急密码开门时,密码验证错误触发该事件） |
| 58 | 人证识别失败 | |
| 59 | 人脸识别失败 | |
| 60 | 身份证认证 | |
| 61 | 未到使用时间 | |
| 62 | 禁用用户权限 | |
| 63 | 温度异常 | |
| 64 | 口罩检测不通过 | 场景：启用口罩检测开门参数，口罩检测不通过上传事件 |
| 65 | 指纹开门失败 | |
| 66 | 人脸已过期 | |
| 67 | 刷身份证开门 | |
| 68 | 刷身份证开门失败 | |
| 69 | 复制卡 | |
| 70 | 蓝牙开门无权限 | 梯控联动可能传递的 userid 找不到时触发 |
| 71 | 对讲开门失败 | |

| | | |
|---|---|---|
| 72 | 二维码未到使用时间 | |
| 73 | 二维码已过期 | |
| 74 | 二维码开门失败 | |
| 75 | 临时密码未到使用时间 | |
| 76 | 临时密码已过期 | |
| 77 | 临时密码开门失败 | |
| 78 | 用户密码过期 | |
| 79 | 手机蓝牙开门失败 | |
| 81 | 门被意外打开报警 | 指在正常事件（如门禁权限的人员刷卡开门、密码开门、门常开时段开门、远程开门、联动设置的开门）以外的情况下,使得门磁检测到门被打开的情况,均为门被意外打开 |
| 82 | 无效卡连续刷卡 5 次 | |
| 83 | 防拆报警触发 | 135 为防拆报警解除 |
| 84 | 一键报警触发 | 室内机告警事件，139 为一键报警解除 |
| 85 | 火警报警触发 | 目前为室内机使用，门口机也有可能用，142 为火警报警解除 |
| 86 | 健康码开门失败 | 需在事件上传中增加 extraData 附加数据健康码信息 |
| 87 | 紧急常闭时开门 | 需在事件上传中增加 extraData 附加数据中上传 openType 开门方式(1：刷脸；2:刷卡；2:刷身份证；4:密码；5：二维码；6：指纹:) |
| 88 | 燃气报警触发 | 室内机防区扩展事件，140 为燃气报警解除 |
| 90 | 人脸注册失败 | 下发到设备的人脸注册失败事件，cardNo 字段为 faceID，reason 字段上传失败原因：1、图片格式错误；2、图片中未发现人脸；3、图片模糊；4、人脸角度不符合要求；5、人脸面积过小；6、图片中存在多张人脸；7、图片下载失败；8、图片过大【不能超过 720p】；9、图片文件不存在(404)；10、人脸亮度过低；11、人脸重复 |
| 91 | 超级 sim 卡开门失败 | |
| 92 | 离线二维码开门失败 | |
| 93 | 卡扇区校验失败 | 卡已注册，但校正卡扇区秘钥门编号失败 |
| 95 | 1 分钟连续 5 次输入 | 目前为仅门锁使用 |

| | | |
|---|---|---|
| | 错误密码 | |
| 96 | 1 分钟连续 5 次指纹识别失败 | 目前为仅门锁使用 |
| 97 | 红外入侵触发 | 室内机防区扩展事件，暂无解除事件，人为干预 |
| 98 | 烟雾感应触发 | 室内机防区扩展事件，141 为烟雾感应解除 |
| 99 | 门窗入侵触发 | 室内机防区扩展事件，暂无解除事件，人为干预 |
| 100 | 门禁报警触发 | 门禁 Alarm 报警触发，对应 143 的解除事件，暂不需定义具体含义，用于拓展事件 |
| 101 | 水浸感应触发 | 小设备当传感器使用，转发接收到的 433M 信号，暂无解除事件，人为干预 |
| 102 | 有毒气体触发 | 小设备当传感器使用，转发接收到的 433M 信号，暂无解除事件，人为干预 |
| 103 | 出门按钮持续按下 | 当出门按钮按下超过 5 秒将触发事件（此时出门开关信号将被拦截不再触发开门动作） |
| 104 | 连续 5 次开门失败 | 目前为仅门锁使用 |
| 105 | 已到达使用次数 | |
| 120 | 门已打开 | 当门磁检测到门已被正常打开时,触发该正常事件 |
| 121 | 门已关闭 | 当门磁检测到门已被正常关闭时,触发该正常事件 |
| 122 | 出门按钮被按下 | 当出门按钮按下时,门打开 |
| 123 | 出门按钮被弹起 | 当出门按钮弹起时,门开始计时,到达计时时,门关闭 |
| 126 | 设备启动 | 设备启动时触发该正常事件 |
| 127 | 个人密码开门 | |
| 129 | 触发出门按钮(被锁定) | 触发出门按键,门不打开 |
| 135 | 防拆报警解除 | |
| 136 | 人脸注册成功 | 下发到设备的人脸注册成功事件，cardNo 字段复用为 faceID 内容 |
| 137 | 人数超额限入 | 超出设备设定的进入人数，不能进入 |
| 138 | 口罩检测超时 | |
| 139 | 一键报警解除 | 室内机告警事件，88 为一键报警解除 |

| | | |
|---|---|---|
| 140 | 燃气报警解除 | 室内机防区扩展事件，88 为燃气报警触发 |
| 141 | 烟雾感应解除 | 室内机防区扩展事件，98 为烟雾感应触发 |
| 142 | 火警报警解除 | 目前为室内机使用，门口机也有可能用，85 为火警报警触发 |
| 143 | 门禁报警解除 | 门禁 Alarm 信号解除 |
| 145 | 掌纹识别失败 | |
| 146 | 掌纹注册失败 | 下发到设备的掌纹注册失败事件，cardNo 字段为 palmID，reason 字段上传失败原因：1、图片大小不对（目前需要 480x640 的）；2、图片下载失败；3、其他错误； |
| 147 | 掌纹注册成功 | |
| 173 | 网络断开 | 上传该类型时需上传 reason 原因字段，reason(1:WIFI 断连；2:WIFI 连接超时；3：MQTT 断连；4：MQTT 连接超时) |
| 174 | 设备故障 | 上传该类型时需上传 reason 原因字段，reason(1:刷卡异常；2:摄像头异常；3:序列号丢失)，注意：不能重复上传 |
| ~~200~~ | 呼叫房号事件 | ~~已废弃-走开始-结束通话~~ |
| 201 | 进入管理设置 | |
| 202 | 人脸检测抓拍 | |
| | | |
| | | |
| | | |

# 8. 常见问题

## 8.1 设备服务器地址如何修改?

支持修改方式如下：

1. 【常用】安卓系统设备-在设备上修改，设备需是支持触屏或鼠标，设备上输入；#115200#6#998#；
2. 【常用】嵌入式 Linux 设备-使用【嵌入式一体机设备 IP 工具.exe】配置工具进行修改（可联系对接技术人员提供）；
3. 设备先添加到我们平台，联网上线，然后提供服务器地址联系我们在线切换服务器地址；
4. 设备先添加到我们平台，在小区中添加员工所属部门权限开启设备管理权限，然后在 APP 设备管理中通过蓝牙配

置设备服务器地址；

5. 对接 APP 蓝牙 SDK，在自己的 APP 中通过蓝牙进行修改；
6. 采购设备时联系商务人员需切换服务器地址（提供服务器地址），设备出厂时切换好服务器地址再发货（适用后期批量订货）；
7. 根据设备协议中【设备控制->修改连接服务器地址及通讯协议】指令进行下发修改，此方法仅适用于对接成功后进行修改；

## 8.2 一直收到设备的 TCP 鉴权请求，是不是返回数据有问题?

检查数据返回格式和内容没问题情况下，一般为没有在返回的 json 数据结尾增加换行符 \r\n，需要在返回内容增加换行符 \r\n 返回，设备以\r\n 为单个数据包的结束标志进行解析。

JAVA 代码示例：ctx.channel().writeAndFlush(JSONUtil.toJsonStr(tcpJson) + "\r\n");

另外回复的数据格式是否符合要求，特别注意大小写的问题，如：message: OK，返回的 OK 是否是大写，如果是小写设备将判断异常导致重复鉴权。

## 8.3 如何监听设备在线状态?

1.订阅 MQTT 系统主题 connected 和 disconnected，设置 setKeepAliveInterval，详情参考->2.2 修改系统主题订阅权限

2.使用定时任务每 5 分钟或 10 分钟通过 MQTT http api 获取所有客户端信息和服务器数据库数据进行比对校准，详情参考->2.3 修改系统主题订阅权限

## 8.4 人脸设备使用 4G 网络时记录中未上传抓拍图片?

设备为了避免 4G 流量默认关闭了抓拍图片上传，可通过下发设备参数 uploadRecordImage 设置为 2 开启（4G 网络非不要不建议开启图片抓拍），uploadRecordImage 默认 1 智能模式;（0 关闭;1 智能模式 2 启用【智能模式 4G 网络默认不上传，其他网络默认上传】）

## 8.5 为什么下发人脸/卡数据后不能刷卡刷脸?

1.需确认先下发了对应的 user（用户）表数据，再下发的人脸/卡表数据；

2.检查用户 ID，人脸 ID，卡 ID 是否有超过整形大小范围，最大值：4294967295，超过该范围设备将会大小溢出保

存数据失败，从而导致无法正常识别；

## 8.6 梯控如何对接授权?

1. 需要在设备参数中将 启用梯控 485 输出 enableEle485 设置为启用才会输出梯控信号给梯控板；
2. 远程开门时，如果有下发楼层权限，则需要添加 floor 字段，内容是楼层开放的十六进制数据，eleKeyMode 为梯控按键模式：1：直达，2：手选，默认不传时为直达；
3. 设备本地验证刷脸和刷卡（身份证）联动梯控，服务器下发用户信息（用户以及楼层权限，用户卡表，用户人脸表），刷脸或刷卡时设备将根据找到的用户信息及授权楼层，然后授权楼层进行 485 信号输出给梯控板；
4. 在线验证密码/二维码时根据协议返回对应 floor 楼层字段将授权楼层进行 485 信号输出给梯控板；
5. 在线验证卡号（身份证）时，需要在设备参数->控制参数->卡验证方式(cardVerifyType) 的值为 8（在线验证卡号）时才触发，默认刷卡或身份证是设备本地验证的，然后后验证密码/二维码一样返回对应 floor 楼层字段将授权楼层进行 485 信号输出给梯控板；
6. 授权单层是点亮直达，多层不会点亮（多层点亮不合理，会停靠多层），只会授权，需要手选所要达到的楼层，授权时间通过设备参数->门控参数->梯控开门时长（elevctlholdtimeout)控制，最大值 250（秒），建议设置 10 秒，如果为远程开门时可设置 time 字段的梯控时长；
7. 楼层梯控继电器下标赋值都是使用 64 层楼层二进制值转为 16 进制进行赋值，长度必须为 16，不足补 0。 比如 0000000000000000000000000000000000000000000000000000000000000001，表示是只有 1 楼（继电 器下标 1）的权限，转 16 进制为：0000000000000001。
8. 楼层数值 16 进制转换 JAVA 代码示例

```
//二进制字符串转十六进制工具类
public class HexConversionUtil {
    //二进制字符串转十六进制
    public static String binaryToHexadecimal(String binary) {
        if (StringUtils.isEmpty(binary)) return "";
        //不足四位补足四位
        while (binary.length() % 4 != 0) {binary = 0 + binary;}
        int byteLen = binary.length() / 4;
        byte[] bytes = new byte[byteLen];
        String hexadecimal = "";
        for (int i = 0; i < byteLen; i++) {
            String substring = binary.substring(binary.length() - 4);
            binary = binary.substring(0, binary.length() - 4);
            bytes[i] = Byte.parseByte(substring, 2);
            if (bytes[i] >= 10) {
                switch (bytes[i]) {
                    case 10:
                        hexadecimal = "A" + hexadecimal;
                        break;
                    case 11:
                        hexadecimal = "B" + hexadecimal;
                        break;
                    case 12:
                        hexadecimal = "C" + hexadecimal;
                        break;
                    case 13:
                        hexadecimal = "D" + hexadecimal;
                        break;
                    case 14:
                        hexadecimal = "E" + hexadecimal;
                        break;
                    case 15:
                        hexadecimal = "F" + hexadecimal;
                        break;
                }
            } else {
                hexadecimal = bytes[i] + hexadecimal;
            }
        }
        return hexadecimal;
    }
}
```

```
public static void main(String[] args) {
       System.out.println(HexConversionUtil.binaryToHexadecimal(
   "00000000000000000000000000000000000000000000000000000000000000000001"));
}

// 根据授权楼层集成转为 16 进制楼层数值的方法
private static String getFloorHexStrByFloorList(List<Integer> floorList) {
   // 过滤掉大于 64 的楼层
   floorList = floorSet.stream().filter(e -> e != null && e <=
64).collect(Collectors.toList());
   if (floorList.size() == 0)
      return "0000000000000000";
   StringBuilder str = new StringBuilder();
   for (int i = 1; i <= 64; i++) {
      if (floorList.contains(i)) {
         str.insert(0, "1");
      } else {
         str.insert(0, "0");
      }
   }
   return HexConversionUtil.binaryToHexadecimal(str.());
}
```