第三方平台对接人脸门禁机

TCP 通讯协议 V3.7

版本记录

日期	版本	说明
2018-11-15	V1.0	确定通讯协议框架
2018-12-20	V1.0	完善人员管理、照片管理、设备控制消息
2019-4-2	V2.0	完善人员管理、照片管理、设备控制消息，修改验签方式，版本号升为 2
2019-5-5	V2.1	新增设备注册人员、删除人员、添加照片、删除照片接口
2019-5-7	V2.2	1. 增加手工抓拍指令：snapshot_picture 2. 设备注册（register)请求协议中增加属性：type，标识设备类型
2019-5-31	V2.5	1.增加时间同步指令：sync_time
2019-6-17	V2.6	1.增加协议 4.6.9：（平台请求设备同步人员）sync_person; 2.增加协议 4.6.10：设备同步人员（device_sync_person)
2019-7-24	V2.7	1. 添加远程开闸 4.5.6 2. 添加开闸模式 4.5.7
2019-8-22	V2.8	1. 添加通过韦根输入获取 IC 卡号，并可上传到平台 2. 添加通过 usb 身份证读卡器获取身份证信息，并上传到平台
2019-10-30	V2.9	1.识别记录返回结果,可选择带回屏幕显示内容
2019-11-07	V2.9	1. 增加 add_person_face（添加人员和人脸信息），平台将人员和人脸的信息一次性传给设备，要求协议头版本号 3 或以上。
2019-03-24	V3.0	设备上报识别结果，增加人体温度字段
2019-03-27	V3.1	上报识别结果回应信息，snaptime，opendoor 为可选择信息
2020-04-10	V3.2	1. add_person_face 增加扩展通行规则 extPassRule 2. 增加 4.5.9 设备通行规则配置。
2020-05-11	V3.3	1. 修改文件名称为<第 3 方平台对接人脸门禁机 TCP 通信协议>
2020-05-12	V3.4	1. 增加 4.6.3 设备上报二维码信息（qrcode_data） 2. 增加 4.5.10 系统信息设置 (config_system) ----设备通行模式设置 (OpenDoor) 3. 完善 4.6.1 上报识别结果（snapshot_face）身份证信息
2020-08-04	V3.5	1.增加 4.5.10.19 身份证号限制配置
2021-04-20	V3.6	1.增加识别结果获取

2021-06-16	V3.7	1.增加可视对讲（只有可视对讲设备才支持）

目录

1 消息定义

1.1 消息结构

消息标识	版本号	消息类型	数据类型	保留字段	时间戳	消息序号	数据长度	消息体
Magic	Version	Msgtype	Datatype	Resv	Timestamp	Seq	datasize	msgBody
4 字节	1 字节	1 字节	1 字节	1 字节	4 字节	4 字节	4 字节	可变字节

1.2 消息头字段说明

1) Magic：消息标志固定为“QYZN”

2) Versio：版本号，当前为 2

3) msgType： 消息类型，见下

4) dataType： 数据类型，目前只支持 json 数据

5) resv： 保留

6) timestamp： 消息创建时间，UNIX 时间戳，单位秒

7) squence： 消息序号，初始值随机，每次递增，消息类型章节会详细说明

8) datasize： 消息体数据长度

9) msgBody：消息体，json 字符串的可变字节

1.3消息字节序约束

通信中字段均为小端字节序；如 0x12345678，内存低地址存储 0x78,内存高地址存储 0x12;

1.4消息示例

例如：设备的注册消息内容如下：

```
51 59 5a 4e 03 03 01 00 47 ec 40 5d 10 00 00 00  QYZN....G.@]....
a1 00 00 00 7b 22 6d 65 73 73 61 67 65 22 3a 22  ....{"message":"
72 65 67 69 73 74 65 72 22 2c 22 64 61 74 61 22  register","data"
3a 7b 22 73 65 72 69 61 6c 22 3a 22 30 61 39 33  :{"serial":"0a93
31 30 65 37 61 63 63 32 39 38 64 33 22 2c 22 6d  10e7acc298d3","m
61 63 22 3a 22 30 30 36 34 38 37 31 45 43 42 30  ac":"0064871ECB0
31 22 2c 22 74 69 6d 65 73 74 61 6d 70 22 3a 22  1","timestamp":"
31 35 36 34 35 33 35 38 37 39 22 2c 22 73 69 67  1564535879","sig
6e 22 3a 22 39 66 35 62 35 37 65 64 62 63 39 35  n":"9f5b57edbc95
33 64 65 33 33 64 61 65 31 38 35 36 63 63 33 35  3de33dae1856cc35
64 37 39 62 22 2c 22 74 79 70 65 22 3a 22 48 47  d79b","type":"HG
30 32 22 7d 7d                                   02"}}
```

红色为框为消息头 16 进制数据，蓝色框内为消息体 json 格式的 16 进制数据

2 消息类型

2.1概述

目前定义了 4 种消息类型，消息体目前采用 JSON 格式，使用 UTF-8 编码，消息体的字符串的长度单位为字节。

```
enum PacketType
{
        HEARTBEAT = 1,   //心跳消息包
        NOTIFICATION = 2,   //通知消息包
        REQUEST = 3,        //请求消息包
        RESPONSE = 4        //相应消息包
}
```

2.2心跳消息包（HEARTBEAT）

心跳消息为设备发给服务器的消息，来保持设备的在线，不带任何数据，sequence 序列号不用递增，保持为零。

请求消息体为空；心跳的响应消息体也为空，默认 25 秒进行心跳保活；

2.3通知消息包（NOTIFICATION）

通知消息是设备或服务器发给对方的，只是通知对方某一事件，不需要对方返回，需要携带数据，sequence 需要每次递增。

消息体的格式为{“message”:””, data:””}

message 是具体的消息类型

data 是消息携带的参数

2.4请求消息包（REQUEST）

请求消息是设备或服务器发给对方的请求，需要对方返回结果，具体的请求内容携带在包的数据域里面，sequence 需要每次递增。
消息体的格式为{“message”:””, data:””}

message 是具体的消息类型

data 是消息携带的参数

2.5返回消息包 （RESPONSE）

返回消息是设备或服务器对 REQUEST 请求的的返回数据，sequence 需要和对应的 REQUEST 消息保持一致

返回消息报文的格式是{“result”:””,”desc”:””,”data”:””}

result 是返回结果，0 表示成功，其他值见错误码

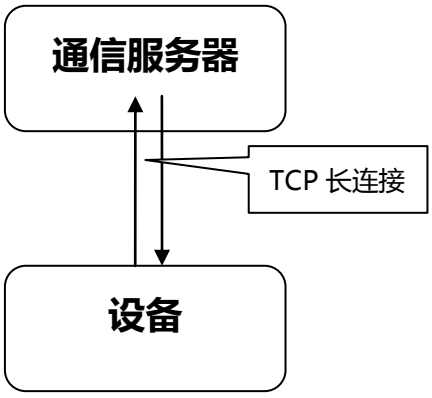
desc 是对返回结果的描述

data 是返回的数据

字段	类型	必须	描叙
result	int	Y	见错误码
desc	String	Y	
data	String	Y	

3 通信交互过程

3.1通信交互图



3.2详细交互过程

1) 设备第一次连接平台时，需要先发送注册（register）消息，获取设备在平台的唯一标志 uuid 和加签秘钥；
2) 后面设备每次登陆都需携带 uuid，需支持发起重新注册；
3) 设备和通信服务器一直保持长连接，等待服务器的指令，如果一直没消息交互，设备需每隔 25s 给通信服务器发送心跳包，

否则通信服务器会断开和设备的连接；是否为 3 次没有检测到心跳消息；

4) 设备发送注销消息（logout），断开和服务器的连接；
5) 设备或服务器给对方发送请求消息时，如果 20s 没返回，将会重发，重发消息的 sequence 要和原消息保持一致；重发 3 次，对方未反馈时，判定为对方通信异常；

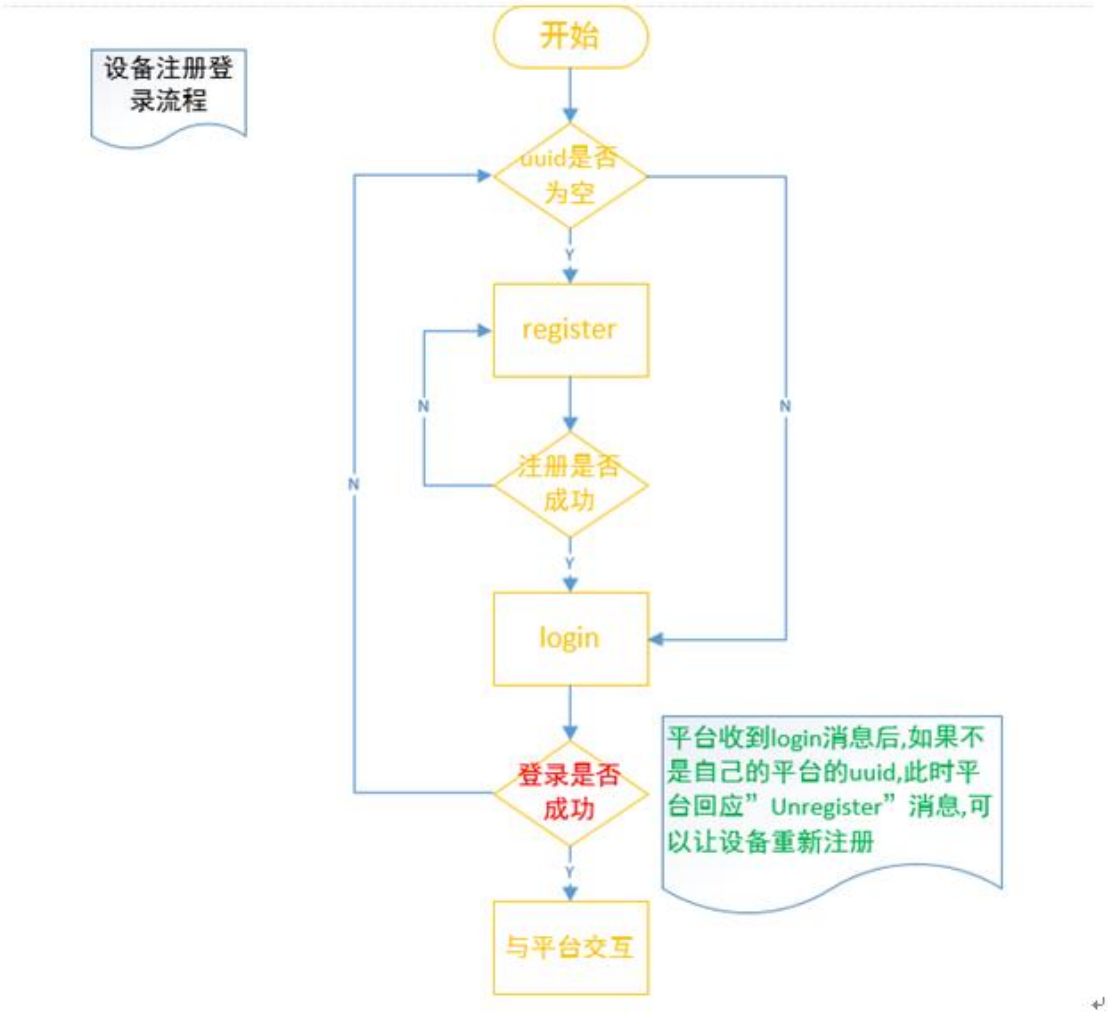
4 消息体定义

4.1概述

消息体目前采用 JSON 格式，使用 UTF-8 编码，消息体的字符串的长度单位为字节。

4.2 基本消息

4.2.1 注册登录流程图



4.2.2 注册（register）

方向： Device >> Server

message: register

参数 data: {“serial”:””, “mac”:””, ”timestamp”:””, ”sign”:””}

字段	类型	长度	必须	描叙
serial	String	32	Y	设备序列号
mac	String	12	Y	设备 MAC 地址，字母用大写表示
timestamp	String	16	Y	请求时的时间戳，相对于 1970 的时间

sign	String	32	Y	参数加签，md5(serial+mac+timestamp+magic)，32 字节
type	String	32	Y	设备类型，如：KG02

Magic 的值为字符串“P3K0eikSRSJHliME”

返回 data: {“uuid”:””,passwd:”” }

字段	类型	长度	描叙
uuid	String	32	设备在平台上的唯一标志
passwd	String	16	加签秘钥

示例：

Request:

```
{
    “message”: “register”,
    “data”:{
        “serial”: “0c16d06a45914b758ca226ef325422a3”,
        “mac”: “8CEC4B91040C”,
        “timestamp”: “1543563114”,
        “sign”: “4a0aa948d87b1308f5b85cc12beff51e”,
        “type”: “KG02”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{
        “uuid”: “fde0a1d337ef4b8ab6e67af87fcfd3c”,
        “passwd”: “12345678”
    }
}
```

4.2.3 登录（login）

方向： Device >> Server

message: login

参数 data: {“uuid”:””, ”timestamp”:””, ”sign”:””}

字段	类型	长度	必须	描叙
uuid	String	32	Y	设备在平台上的唯一标志
version	String	64	Y	设备版本号

timestamp	String	16	Y	请求时的时间戳
sign	String	32	Y	参数加签，md5(uuid+ passwd +timestamp +magic)

返回 data：{}

示例：

Request:

```
{
    “message”: “login”,
    “data”:{
        “uuid”: “fde0a1d337ef4b8ab6e67af87fcfd3f”,
        “version”: “V1.0.0”,
        “timestamp”: “1543563114”,
        “sign”: “4a0aa948d87b1308f5b85cc12beff51e”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

如果平台 uuid 校验不正确，平台回应“Unregistered”，设备就会重新注册

Response:

```
{
    “result”: 100,
    “desc”: “Unregistered”,
    “data”:{}
}
```

4.3人员管理消息

4.3.1 添加人员（add_person）

方向：Server >> Device

message: add_person

data:

字段	类型	长度	必须	描叙

personId	String	32	Y	人员唯一标识 Id，不能为空
personCode	String	16	Y	人员编号，不能为空
name	String	24	Y	Utf8 编码 ,识别通过后会在设备显示屏上显示该名字
type	int		Y	人员类型，1：白名单，2：黑名单，3：访客,4:陌生人
idNum	String	20	N	身份证号码，如：420381200010013857
icNum	String	8	N	Ic 卡号，如：157342643
passNum	int		N	通行次数
passPeriod	String	36	N	允许通行时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开
passMonday	boolean		N	星期一是否允许通行
passTuesday	boolean		N	星期二是否允许通行
passWednesday	boolean		N	星期三是否允许通行
passThursday	boolean		N	星期四是否允许通行
passFriday	boolean		N	星期五是否允许通行
passSaturday	boolean		N	星期六是否允许通行
passSunday	boolean		N	星期日是否允许通行
startTime	string	20	N	授权开始时间,格式：yyyy-MM-dd HH:mm:ss
expireTime	string	20	N	授权结束时间,格式：yyyy-MM-dd HH:mm:ss
passMode	int			255：没有设置个人通行模式，个人通行模式由设备设置的通行模式决定, 0：刷脸模式，1：人卡合一模式，2：刷卡模式，3：人证模式，4：人证白名单模式,5.人脸或卡

返回 data：{}

示例：

Request:

```
{
    "message": "add_person",
    "data":{
        "personId":"100001",
        "personCode":"100001",
        "name":"张三",
        "type":1,
        "idNum":"420381200010013857",
        "icNum":"157342643",
        "passNum": 10,
        "passPeriod": "08:00~09:30,11:30~14:00,17:00~19:00",
        "passMonday": true,
        "passTuesday": true,
        "passWednesday": true,
        "passThursday": true,
        "passFriday": true,
        "passSaturday": false,
        "passSunday": false,
        "startTime": "2019-01-01 00:00:00",
        "expireTime": "2020-01-01 00:00:00",
    }
}
```


Response:


```
{
    “result”: 0,
    “desc”:“Success”,
    “data”:{}
}
```


4.3.2 更新人员（update_person）

方向： Server >> Device

message: update_person

data:

字段	类型	长度	必须	描叙

personId	String	32	Y	人员唯一标识 Id，不能为空
personCode	String	16	Y	人员编号，不能为空
name	String	24	N	Utf8 编码，识别通过后会在设备显示屏上显示该名字
type	int		N	人员类型，1：白名单，2：黑名单，3：访客
idNum	String	20	N	身份证号码,如：420381200010013857
icNum	String	8	N	Ic 卡号，如：157342643
passNum	int		N	通行次数
passPeriod	String	36	N	允许通行时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开
passMonday	boolean		N	星期一是否允许通行
passTuesday	boolean		N	星期二是否允许通行
passWednesday	boolean		N	星期三是否允许通行
passThursday	boolean		N	星期四是否允许通行
passFriday	boolean		N	星期五是否允许通行
passSaturday	boolean		N	星期六是否允许通行
passSunday	boolean		N	星期日是否允许通行
startTime	string	20	N	授权开始时间；yyyy-MM-dd HH:mm:ss
expireTime	string	20	N	授权结束时间；yyyy-MM-dd HH:mm:ss

返回 data：{}

示例：

Request:

```
{
    "message": "update_person",
    "data":{
        "personId":"100001",
        "personCode":"100001"
        "name":"张三",
        "type":1,
        "idNum":"420381200010013857",
        "icNum":"157342643",
        "passNum": 10,
        "passPeriod": "08:00~09:30,11:30~14:00,17:00~19:00",
        "passMonday": true,
        "passTuesday": true,
        "passWednesday": true,
        "passThursday": true,
        "passFriday": true,
        "passSaturday": false,
        "passSunday": false,
        "startTime": "2019-01-01 00:00:00",
        "expireTime": "2020-01-01 00:00:00",
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.3.3 删除人员（delete_person）

方向： Server >> Device

message: delete_person

data: {“idSet”:[]}

字段	类型	长度	必须	描叙
idSet	String Array	1	Y	目前只支持每次删除一个人员，用人员 id（personId）标识

删除人员时，设备将不在存储信息如下：

1）人员信息

2）该人员保存在设备库中的图片

3）目前支持一条命令删除一个人员，如果删除多人，需要发送多条 delete_person 命令

返回 data:{}

示例：

Request:

```
{
    "message": "delete_person",
    "data":{
       "idSet": ["100001"]
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data":{}
}
```

4.3.4 清空人员信息（clear_person）

方向： Server >> Device

message: clear_person

data: {}

清空设备端所有人员信息及人员的图片信息

返回 data:{}

示例：

Request:

```
{
    "message": "clear_person",
    "data":{}
}
```

```
{
    "result": 0,
    "desc": “Success”,
    "data":{}
}
```

4.3.5 手动抓拍图片（snapshot_picture）

平台向设备发送手动抓拍指令，设备在给定的超时时间内抓拍一个合格的人脸图片上传。

方向： Server >> Device

message: snapshot_picture

data: {“timeoutSeconds”:5}

字段	类型	必须	描叙
timeoutSeconds	int	Y	抓拍最长多少秒后超时

返回 data:

字段	类型	必须	描叙
image	String	Y	Base64 编码 JPG 图片，不允许为空，如果未抓拍到合适的图片，则取当前图片。

示例：

Request:

```
{
    "message": "snapshot_picture",
    "data”:{
        "timeoutSeconds": 5
    }
}
```

Response:

```
{
    "result": 0,
    "desc": “Success”,
    "data":{
      "image": "base64"
    ]
}
```

4.3.6 请求设备同步人员（sync_person）

向设备发送同步人员指令，设备收到指令后返回人员总数信息，然后设备通过 device_sync_person 指令主动将设备中的人员及其人脸图片上传到平台。

方向： Server >> Device

message: sync_person

返回 data:

字段	类型	必须	描叙
person_num	int	Y	设备端需要同步的人员总数

示例：

Request:

```
{
        "message": "sync_person"
        "data":{}
}
```

Response:

```
{
    "result": 0,
    "desc": “Success”,
    "data":{
      "person_num": 125
    }
}
```

4.3.7 设备同步人员（device_sync_person）

方向： Device >> Server

message: device_sync_person

data:

字段	类型	长度	必须	描叙
personId	String	32	Y	人员唯一标识 Id，不能为空
personCode	String	16	Y	人员编号，不能为空
name	String	24	Y	识别通过后会在设备的显示屏上显示该名字

type	int		Y	人员类型，1：白名单，2：黑名单，3：访客
idNum	String	20	N	身份证号码，如：420381200010013857
icNum	String	8	N	Ic 卡号，如：157342643
passNum	int		N	通行次数
passPeriod	String	36	N	允许通行时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开
passMonday	boolean		N	星期一是否允许通行
passTuesday	boolean		N	星期二是否允许通行
passWednesday	boolean		N	星期三是否允许通行
passThursday	boolean		N	星期四是否允许通行
passFriday	boolean		N	星期五是否允许通行
passSaturday	boolean		N	星期六是否允许通行
passSunday	boolean		N	星期日是否允许通行
startTime	string	20	N	授权开始时间,格式：yyyy-MM-dd HH:mm:ss
expireTime	string	20	N	授权结束时间,格式：yyyy-MM-dd HH:mm:ss
faces	array		Y	人脸图片，一个人员可以对应多张人脸图片，最多支持 3 张，目前只支持 1 张
faceId	string	32	Y	人脸 ID
image	string		Y	人脸图片 BASE64 编码

返回 data:

字段	类型	长度	必须	描叙
oriPersonId	string	32	Y	原 personId，请求参数中的 personId
newPersonId	String	32	Y	新 personId,不能为空，有可能与 oriPersonId 相等，也有可能不一样，如果不一样，接收端需要将原来的 personId 修改为新 personId.
faces	array		Y	人脸图片，一个人员可以对应多张人脸图片
oriFaceId	string	32	N	原人脸 ID

newFaceId	string	32	N	新人脸 ID,有可能与原人脸 ID 相同，也有可能不一样，如果不一样，接收端需要将原来的 faceId 改为新 faceId

示例：

Request:

```
{
    "message": "device_sync_person",
    "data":{
        "personId":"100001",
        "personCode":"100001",
        "name":"张三",
        "type":1,
        "idNum":"420381200010013857",
        "icNum":"157342643",
        "passNum": 10,
        "passPeriod": "08:00~09:30,11:30~14:00,17:00~19:00",
        "passMonday": true,
        "passTuesday": true,
        "passWednesday": true,
        "passThursday": true,
        "passFriday": true,
        "passSaturday": false,
        "passSunday": false,
        "startTime": "2019-01-01 00:00:00",
        "expireTime": "2020-01-01 00:00:00",
        "faces": [
            {
                "faceId":"3000004",
                "image":"0aB1D98bsdDSBDDS23sdfisfs"
            }
        ]
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”:“Success”,
    “data”:{
      "oriPersonId": "100001",
      "newPersonId": "100002",
       "faces": [
           {
               "oriFaceId":"3000004",
               "newFaceId":"3000004"

           },
           {
               "oriFaceId":"3000005",
               "newFaceId":"3000006"
           }

        ]
    }
}
```

4.3.8 添加人员与人脸（add_person_face）

将人员与人脸信息一起传输给设备。

方向： Server >> Device

message: add_person_face

协议版本号：3

data:

字段	类型	必须	描叙
personId	String	Y	人员唯一标识 Id，不能为空
personCode	String	Y	人员编号，不能为空
name	String	Y	识别通过后会在屏幕上显示该名字
type	int	Y	人员类型，1：白名单，2：黑名单，3：访客
idNum	String	N	保留
icNum	String	N	识别通过后会在屏幕上显示该名字
passNum	int	Y	通行次数
passPeriod	String	N	允许通行时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开

passMonday	boolean	Y	星期一是否允许通行
passTuesday	boolean	Y	星期二是否允许通行
passWednesday	boolean	Y	星期三是否允许通行
passThursday	boolean	Y	星期四是否允许通行
passFriday	boolean	Y	星期五是否允许通行
passSaturday	boolean	Y	星期六是否允许通行
passSunday	boolean	Y	星期日是否允许通行
startTime	string	N	授权开始时间,格式：yyyy-MM-dd HH:mm:ss
expireTime	string	N	授权结束时间,格式：yyyy-MM-dd HH:mm:ss
faces	array	Y	人脸图片，一个人员可以对应多张人脸图片
faceId	string	N	人脸 ID
image	string	N	人脸图片 BASE64 编码
passMode	int		255：没有设置个人通行模式，个人通行模式由设备设置的通行模式决定, 0：刷脸模式，1：人卡合一模式，2：刷卡模式，3：人证模式，4：人证白名单模式,5.人脸或卡
extPassRule	Object	Y	扩展的通行规则，如果内容不为空，则使用该通行规则通行，否则使用旧的（简易）通行规则。

扩展通行规则

周通行规则：**weekly** （定义周一至周日基本通行规则）

字段	类型（长度）	必须	描叙
mode	Byte（1）	Y	0:全天禁止通行 1:全天允许通行 2:分时段允许通行
periods	String(256)	N	当 mode 为 2 时，定义允许通行的时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开。 最多 10 个时段

假日通行规则：**holiday(假日规则优先级高于周规则)**

字段	类型（长度）	必须	描叙
name	String（24）	Y	假日名称

mode	Byte（1）	Y	0:全天禁止通行 1:全天允许通行 2:分时段允许通行
periods	String（256）	N	当 mode 为 2 时，定义允许通行的时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开。 最多 10 个时段

返回 data：{}

示例：

Request:

Request:


```
{
    "message": "add_person_face",
    "data":{
        "personId":"100001",
        "personCode":"100001",
        "name":"张三",
        "type":1,
        "portrait":"fsfgdsgsdfsafds",
        "idNum":"",
        "icNum":"",
        "passNum": 10,
        "passPeriod": "08:00~09:30,11:30~14:00,17:00~19:00",
        "passMonday": true,
        "passTuesday": true,
        "passWednesday": true,
        "passThursday": true,
        "passFriday": true,
        "passSaturday": false,
        "passSunday": false,
        "startTime": "2019-01-01 00:00:00",
        "expireTime": "2020-01-01 00:00:00",
        "passMode": 255,
        "faces": [
            {
                "faceId":"3000004",
                "image":"0aB1D98bsdDSBDDS23sdfisfs"
            }
        ],
        "extPassRule": {
            "id":"规则 ID",
            "name":"规则名称",
            "updateTime":"yyyy-MM-dd HH:mm:ss",
            "weekly":{
                "mon":{
                    "mode":0
```

```
            },
            "tue":{
                "mode":1
            },
            "wed":{
                "mode":2,
                "periods":"00:00~05:00,13:30~18:00"
            },
            "thu":{
                "mode":1
            },
            "fri":{
                "mode":1
            },
            "sat":{
                "mode":1
            },
            "sun":{
                "mode":1
            }
        },
        "holiday":[
            {
                "name":"中秋节",
                "dayRange":"2019.12.25~2019.12.25",
                "mode":0
            },
            {
                "name":"国庆节",
                "dayRange":"2019.10.01~2019.10.03",
                "mode":2,
                "periods":"00:00~05:00,13:30~18:00"
            },

            ...
        ]
    }
  }
}
```

Response:

```
{
    “result”: 0,
    “desc”:“Success”,
    “data”:{}
}
```

4.4照片管理消息

4.4.1 添加照片（add_face）

方向： Server >> Device

message: add_face

data: {“personid”:””,”faceid”:””, “image”:”base64”}

字段	类型	长度	必须	描叙
personId	String	32	Y	人员唯一标识 ID，不能为空
faceId	String	32	Y	图片唯一标识 ID，不能为空
image	String		Y	Base64 编码

返回 data:{ }

示例：

Request:

```
{
    "message": "add_face",
    "data":{
        "personId":"100004",
        "faceId":"3000004",
        "image":"0aB1D98bsdDSBDDS23sdfisfs"
    }
}
```

Response:

```
{
    "result":0,
    "desc":"Success",
    "data":{}
}
```

4.4.2 删除照片（delete_face）

方向： Server >> Device

message: delete_face

data: {“faceId”:””,”personId”:””}

字段	类型	长度	必须	描叙
faceId	String	32	N	faceId 和 personId 只需指定一

personId	String	32	N	个，如果指定了 personId 表示删除该人员的所有照片

返回 data:{ }

示例：

Request:

```
{
    "message": "delete_face",
    "data":{
        "faceId":"3000004",
        "personId": "1000004"
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data":{}
}
```

4.5.1 设备重启（device_restart）

方向： Server >> Device

message: device_restart

参数 data: {}

返回 data: {}

示例：

Request:

```
{
    “message”: “device_restart”,
    “data”:{}
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.2 设备升级（device_upgrade）

方向： Server >> Device

message: device_upgrade

参数 data: {“url”: ” ”}

字段	类型	长度	描叙
url	String	128	升级包的 url 地址，最大不超过 128 字节
version	String	64	升级程序的版本号
check	String	32	升级文件的校验码 ,md5(“升级文件内容”),32 字节表示

返回 data: { }

示例：

Request:

```
{
    “message”: “device_upgrade”,
    “data”:{
        “url”:“http://ip:port/file.bin”,
        “version”:”V1.0.2_2018-11-10_12:20 xxx”,
        “check”:”e10adc3949ba59abbe56e057f20f883e”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.3 串口透传（serial_output）

方向： Server >> Device

message: serial_output

参数 data: {“index”:””,”content”:””}

字段	类型	长度	必须	描叙
index	int		Y	0：第一路 RS485 目前只有一路
content	String	256	Y	发送的数据用十六进制表示，其中的字母用大写； 如：发送字母“A”,内容为“41” 发送数字“1234”，内容为“31323334” 发送十六进制 0x1234EF，内容为“1234EF“

返回 data: {}

示例：

Request:

```
{
    “message”: “serial_output”,
    “data”:{
        “index”:0,
        “content”:”1234EF”
        }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.4 韦根输出（weigand_output）

方向： Server >> Device

message: weigand_output

参数 data: {“content”:”” }

字段	类型	长度	必须	描叙
content	String	8	Y	发送的数据用十六进制表示，其中的字母用大写； 如：发送字母“A”,内容为“41” 发送数字“1234”,内容为“31323334” 发送十六进制 0x1234EF，内容为

				“1234EF “ 发送的内容长度受格式限制 ,26bit 只能传 3 个字节数据，34bit 只能传 4 个字节

返回 data: {}

示例：

Request:

```
{
    “message”: “weigand_output”,
    “data”:{
            “content”:”1234EF”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.5 清空记录（record_clear）

方向： Server >> Device

message: record_clear

参数 data: {}

清空设备上所有业务记录，如：人脸比对记录

返回 data: {}

示例：

Request:

```
{
    “message”: “record_clear”,
    “data”:{}
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.6 远程开闸（open_door）

方向： Server >> Device

message: open_door

参数 data: {}

服务器向设备发送开闸（开门）指令

返回 data: {}

示例：

Request:

```
{
    “message”: “open_door”,
    “data”:{}
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.

方向： Server >> Device

message: online_openDoor

参数 data: {“online”: ,”mode”:}

字段	类型	必须	描叙
online	int	Y	注册远程控制端状态，0：离线 1：在线
mode	int	Y	0：不关注远程控制状态，设备控制开闸 1: 离线，设备控制开闸。在线，由远端控制是否开闸 2:不关注远程控制状态 设备不控制开闸 ,由远程控制端控制

返回 data: {}

示例：

Request:

```
{
    “message”: “online_open_door”,
    “data”:{
        “online”:1,
        “mode”:1
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.5.8 平台请求设备刷卡（swiping_card）

平台通过请求设备刷卡指令获取卡号，然后可以使用卡号来添加人员

方向： Server >> Device

message: swiping_card

data:

字段	类型	必须	描叙
timeOut	int	Y	刷卡超时（s）

返回 data：{}

字段	类型	必须	描叙
cardType	Int	Y	刷卡类型，0：身份证卡,1:永居证，2:港澳通行证 3:IC 卡
name	string	N	姓名
idNum	string	Y	证件号码
ethnic	string	N	民族
gender	string	N	性别
birthday	string	N	出生日期，格式：yyyyMMdd
address	String	N	地址
idIssuingOrgan	string	N	发证机关
idUsefulLife	string	N	有效期限 ,格式 :yyyyMMdd~yyyyMMdd

示例：

Request:

```
{
    "message": "swiping_card",
    "data":{
        "timeOut":10
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”:“Success”,
    “data”:{
            “cardType”:1,
            “name”:“Jim”,
             “idNum”:“420381198802141789”,
             “ethnic”:“汉”,
             “gender”:1,
             “birthday”:“1989-10-10”,
             “address”:“北京市朝阳区清湖路”,
             “idIssuingOrgan”:“北京市朝阳区公安局”,
             “idUsefulLife”:“2000-01-01~2030-01-01”
    }
}
```

4.5.9 设备通行规则配置（config_device_pass_rule）

配置设备的通行规则

方向： Server >> Device

message: config_device_pass_rule

协议版本号：4

data:

周规则：**weekly** （定义周一至周日基本通行规则）

字段	类型	必须	描叙
mode	Byte（1）	Y	0:全天禁止通行 1:全天允许通行 2:分时段允许通行
periods	String	N	当 mode 为 2 时，定义允许通行的时段，

	（256）		用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开。

假日规则：**holiday(假日规则优先级高于周规则)**

字段	类型	必须	描叙
name	String(24)	Y	假日名称
mode	Byte(1)	Y	0:全天禁止通行 1:全天允许通行 2:分时段允许通行
periods	String(256)	N	当 mode 为 2 时，定义允许通行的时段，用时分表示，如 08:00~09:30 表示早上八点到九点半允许通行，可以设置多段时间，各段用逗号隔开。 最多支持 10 个时间段

返回 data：{}

示例：Request:

```
{
    "message": "config_device_pass_rule",
    "data":{
        "id":"规则 ID（32）",
        "name":"规则名称（36）",
        "updateTime":"yyyy-MM-dd HH:mm:ss",
        "weekly":{
            "mon":{
                "mode":0
            },
            "tue":{
                "mode":1
            },
            "wed":{
                "mode":2,
                "periods":"00:00~05:00,13:30~18:00"
            },
            "thu":{
                "mode":1
            },
            "fri":{
                "mode":1
            },
            "sat":{
                "mode":1
```

```
            },
            "sun":{
                "mode":1
            }
        },
        "holiday":[
            {
                "name":"中秋节",
                "dayRange":"2019.12.25~2019.12.25",
                "mode":0
            },
            {
                "name":"国庆节",
                "dayRange":"2019.10.01~2019.10.03",
                "mode":2,
                "periods":"00:00~05:00,13:30~18:00"
            },

            ...
        ]
    }
}
```

Response:

```
{
    “result”: 0,
        “desc”:“Success”
}
```

4.5.10 系统信息设置与获取（config_system/get_configs）

4.5.10.1 版本信息--读取 (version)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 version，可支持:1 次获取多个组的配置信息。

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 version，可支持:1 次获取多个组的配置信息。
devType	String	32	Y	设备类型
appVer	String	32	Y	业务软件
sysVer	String	32	Y	文件系统
license	String	32	Y	序列号

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            "version"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "version",
                "devType": "ZG01-C",
                "appVer" : "2.0.8_2020-06-17_11:01 ZG01-C",
                "sysVer" : "2.1.0 2019-12-27 19:30\n",
                "license” : "be4f35b915a1aa9d",
            }],
    "RspMsg" : "get_version"
}
```

4.5.10.2 网络信息--获取(netinfo)

方向： Server >> Device

message: get_configs

参数 data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 netinfo，可支持:1 次获取多个组的配置信息。

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 netinfo，可支持:1 次获取多个组的配置信息。
name	String	8	Y	目前支持单/双网卡 命名为“eth0” “eth1”
macaddr	String	20	Y	Eth0/1 MAC 地址 Mac1 地址，暂时为空值
ipaddr	String	20	Y	Eth0/1 IP 地址
maskaddr	String	20	Y	Eth0/1 子网掩码
gateway	String	20	Y	默认网关
DNS	String	20	Y	DNS 服务器

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
           "netinfo"
       ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "netinfo",
                 "item": [{
                     "name": "eth0"
                     "macaddr ": "7E:22:4A:6E:FD:37"
                     "ipaddr" : "192.168.1.249"
                     "maskaddr" : "255.255.255.0"
                     "gateway" : "192.168.1.1"
                     "DNS": "192.168.1.1"
                 },{
                     "name": "eth1"
```

```
                        "macaddr ": ""
                        "ipaddr" : "192.168.1.249"
                        "maskaddr" : "255.255.255.0"
                        "gateway" : "192.168.1.1"
                        "DNS": "192.168.1.1"
                    }]
                }],
    "RspMsg" : "get_netinfo"
}
```

4.5.10.3 时间--获取(time)

方向： Server >> Device

message: get_configs

参数 data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 time

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 time
timezone	Int		Y	所在时区[取值范围 0~24] 例如：东八区北京：20
currenttime	String	20		当前时间
time_control	Int		Y	同步方式(0~2)： 0：不同步管理系统、NTP 服务器时间 1：同步管理系统时间 2：同步 NTP 服务器时间
ntpServer	String	64	Y	NTP 服务器
ntpPort	Int		Y	NTP 端口（0~99999）

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
           "time"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
        "groupKey": "time",
        "timezone": 20,
        "time_control": 1,
        "currenttime": "2020-07-01 16:56:15",
        "ntpServer": "192.168.1.250",
        "ntpPort": 456
    }],
    "RspMsg" : "get_time"
}
```

4.5.10.3 时间--设置 (time)

方向： Server >> Device

message: **config_system**

参数 Data: "time"

time：（请参照协议示例）：

字段	类型	长度	必须	描叙
currenttime	String	32	Y	当前时间
time_control	Int		Y	时间同步方式（3）： 0：不同步管理系统、不同步 NTP 服务器时间 1：同步管理系统时间 2：同步 NTP 服务器时间
ntpPort	Int		Y	NTP 端口
timezone	Int		Y	所在时区（共 25 个）：eg:（东八区）北京对应的值为 20
ntpServer	string	32	Y	NTP 服务器

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{
    "message": "config_system",
    "data": {
        "time": {
            "currenttime": "2020-07-01 16:56:15",
            "time_control": 1,
            "ntpPort": 3030,
            "timezone": 20,
            "ntpServer": "192.168.1.250"
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_time"
}
```

4.5.10.4 补光--获取(ledctl)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 **ledctl**

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 **ledctl**
mode	int		Y	补光模式[范围：0~2] 0：常灭 1：常亮 2：自动
closeScreen	int		Y	自动熄屏[范围：0，1，2，5，10] 0 ： 从不熄屏 1 ： 1 分钟 2 ： 2 分钟 5 ： 5 分钟 10：10 分钟

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            " ledctl"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
        "groupKey": " ledctl ",
        "mode": 2,
        "closeScreen": 1
    }],
    "RspMsg" : "get_ledctl"
}
```

4.5.10.4 补光--设置(ledctl)

参数 **Data:** “ledctl”

字段	类型	长度	必须	描叙
ledctl	String	32	Y	当前时间
mode	int		Y	补光模式[范围：0~2] 0：常灭 1：常亮 2：自动
closeScreen	int		Y	自动熄屏[范围：0，1，2，5，10] 0 ： 从不熄屏 1 ： 1 分钟 2 ： 2 分钟 5 ： 5 分钟 10：10 分钟

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "ledctl": {
            "mode": 2,
            "closeScreen": 0
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_ledctl"
}
```

4.5.10.5 串口--获取(serial)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 serial

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 serial，可支持:1 次获取多个组的配置信息。
baudrate	int		Y	波特率[取值范围：0, 1, 2, 3, 4, 5, 6, 7] 0：1200 1：2400 2：480 3：9600 4：19200 5：38400 6：5760 7: 115200
databit	int		Y	数据位 [取值范围：5, 6, 7, 8] 对应值：5，6，7，8
stopbit	int		Y	停止位[[取值范围：0, 1, 2] 对应值：0，1，2
checktype	int		Y	校验位[取值范围 0, 1, 2, 3, 4] 0：无校验 1：奇校验 2：偶校验 3：标志校验 4：空校验
putOutType	Int		Y	串口输出[取值范围：1, 2, 3, 10] 1：不输出 2：输出人员 ID 3：输出卡号 ID 4：用户自定义

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "serial"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                    "groupKey": "serial ",
                    "baudrate": 3,
                    "databit" :    8,
                    "stopbit" :    2,
                    "checktype" : 0,
                    "putOutType": 3
                }],
    "RspMsg" : "get_serial"
}
```

4.5.10.5 串口--设置(serial)

参数 **Data:** “serial”

字段	类型	长度	必须	描叙
serial	String	32	Y	当前时间
baudrate	int		Y	波特率[取值范围：0, 1, 2, 3, 4, 5, 6, 7] 0：1200 1：2400 2：480 3：9600 4：19200 5：38400 6：5760 7： 115200
databit	int		Y	数据位 [取值范围：5, 6, 7, 8] 对应值：5，6，7，8
stopbit	int		Y	停止位[[取值范围：0, 1, 2] 对应值：0，1，2
checktype	int		Y	校验位[取值范围 0, 1, 2, 3, 4] 0：无校验 1：奇校验 2：偶校验 3：标志校验 4：空校验
putOutType	String	32	Y	串口输出[取值范围：1, 2, 3, 10] 1：不输出 2：输出人员 ID 3：输出卡号 ID 4：用户自定义

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "serial": {
            "baudrate": 3,
            "databit": 8,
            "stopbit": 1,
            "checktype": 0,
            "putOutType": 3
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": " Success ",
    "data": {},
    "RspMsg": "set_serial"
}
```

4.5.10.6 声音--获取(audioctl)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 audioctl

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 audioctl
level	int		Y	音量[取值范围：0-15]

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "audioctl"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "audioctl",
                "level": 3
            }],
    "RspMsg" : "get_audioctl"
}
```

4.5.10.6 声音--设置(audioctl)

参数 **Data:** “ledctl”

字段	类型	长度	必须	描叙
audioctl	String	32	Y	当前时间
level	int		Y	音量[取值范围：0-15]

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "audioctl": {
           "level": 12
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_level"
}
```

4.5.10.7 韦根--获取(weigand)

方向： Server >> Device

message: get_configs

参数 data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 weigand

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 weigand
type	int		Y	韦根类型[取值范围：0, 1] 0：26 位 1：34 位
InputType	int		Y	韦根输入[取值范围：1, 2, 3] 1：做 HID/PID 转换 2：不做 HID/PID 转换 3：取后 2 个字节
putOutType	int		Y	韦根输出[取值范围: 0, 1, 2, 10] 0：不输出 1：输出人员 ID 2：输出卡号 ID 10：用户自定义

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "weigand"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                    "groupKey": "weigand",
                    "type ": 0,
                    "InputType" :  1,
                    "putOutType" :  1,
                }],
    "RspMsg" : "get_weigand"
}
```

4.5.10.7 韦根--设置(weigand)

参数 **Data:** “weigand”

字段	类型	长度	必须	描叙
weigand	String	32	Y	韦根
type	int		Y	韦根类型[取值范围：0, 1] 0：26 位 1：34 位
InputType	int		Y	韦根输入[取值范围：1, 2, 3] 1：做 HID/PID 转换 2：不做 HID/PID 转换 3：取后 2 个字节
putOutType	int		Y	韦根输出[取值范围: 0, 1, 2, 10] 0：不输出 1：输出人员 ID 2：输出卡号 ID 10：用户自定义

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
       "weigand": {
            "InputType": 1,
            "putOutType": 2,
            "type": 0
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_weigand"
}
```

4.5.10.8 人脸参数--获取(vaparam)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 vaparam

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 vaparam
SimilarThres	int		Y	人脸比对阈值[取值范围: 0~100] 默认值：80
DetThres	int		Y	人脸检测阈值[取值范围：0~100] 默认值：65
LiveThres	int		Y	活体检测阈值[取值范围：0~100] 默认值：50
LiveSwitch	int		Y	活体识别等级[取值范围：0, 1, 2] 0：快速识别，不拒绝照片 1：能拒绝照片+视频 2：能拒绝部分照片+视频
Distance	Int		Y	人脸识别距离[范围：0, 1, 2, 3, 4, 5] 0: 无限制（**默认值：0**） 1: 0.5 米以内 2: 1 米以内 3: 1.5 米以内 4: 2 米以内 5: 3 米以内
safetyhatDecect	bool			安全帽检测 true: 检测 false：不检测

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            "vaparam"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "vaparam",
                "SimilarThres": 80,
                "DetThres" :  65,
                "LiveThres" :  50,
                "LiveSwitch" : 0,
                "Distance": 1，
                "safetyhatDecect": false
            }],
    "RspMsg" : "get_vaparam"
}
```

4.5.10.8 人脸参数--设置(vaparam)

参数 Data: “vaparam”

字段	类型	长度	必须	描叙
vaparam	String	32	Y	人脸参数
SimilarThres	int		Y	人脸比对阈值[取值范围: 0~100] 默认值：80
DetThres	int		Y	人脸检测阈值[取值范围：0~100] 默认值：65
LiveThres	int		Y	活体检测阈值[取值范围：0~100] 默认值：50
LiveSwitch	int		Y	活体识别等级[取值范围：0, 1, 2] 0：快速识别，不拒绝照片 1：能拒绝照片+视频 2：能拒绝部分照片+视频
Distance	Int		Y	人脸识别距离[范围：0, 1, 2, 3, 4, 5] 0: 无限制（**默认值：0**） 1: 0.5 米以内 2: 1 米以内 3: 1.5 米以内 4: 2 米以内 5: 3 米以内
safetyhatDecect	bool		Y	安全帽检测 true: 检测 false：不检测

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "vaparam": {
            "SimilarThres": 80,
            "DetThres": 65,
            "LiveThres": 50,
            "LiveSwitch": 0
            "Distance": 0,
            "safetyhatDecect": false,
           }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_vaparam"
}
```

4.5.10.9 开关量--获取(switchCtrl)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 switchCtrl

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 switchCtrl
switchOutput	bool		Y	开关量输出: true: 输出 false: 不输出
switchOutMode	int		Y	输出模式[取值范围：0, 1] 0: 高电平开闸 1: 低电平开闸
outputInterval	int		Y	抬闸脉冲时长(ms) [范围：0~10000]
outputDelay	int		Y	延迟输出开闸时长(ms) [范围：0~5000]

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "switchCtrl"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                    "groupKey": "switchCtrl",
                    "switchOutput": true,
                    "switchOutMode" :   65,
                    "outputInterval" :   2000,
                    "outputDelay" : 0,
                }],
    "RspMsg" : "get_switchCtrl"
}
```

4.5.10.9 开关量--设置(switchCtrl)

参数 **Data:** “switchCtrl”

字段	类型	长度	必须	描叙
switchCtrl	String	32	Y	开关量
switchOutput	bool		Y	开关量输出: true: 输出 false: 不输出
switchOutMode	int		Y	输出模式[取值范围：0, 1] 0: 高电平开闸 1: 低电平开闸
outputInterval	int		Y	抬闸脉冲时长(ms) [范围：0~10000]
outputDelay	int		Y	延迟输出开闸时长(ms) [范围：0~5000]

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "switchCtrl": {
            "switchOutput": true,
            "switchOutMode": 0,
            "outputInterval": 2000,
            "outputDelay": 0,
         }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_switchCtrl"
}
```

4.5.10.10 终端语音提示--获取(VoiceTips)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙

groupKey	String Array	32	Y	本组关键字：为 VoiceTips

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 VoiceTips
SuccessTips	int		Y	识别成功[范围: 1, 2, 3, 4, 5, 6, 7, 8] 1: 识别成功 2: 请通行 3: 验证通过 4: 欢迎光临 5: 欢迎回家 6: 请慢走 7: 一路平安 8: 请戴安全帽
FailTips	int		Y	识别失败[取值范围：1, 2] 0: 未授权 1: 陌生人

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "VoiceTips"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                    "groupKey": "VoiceTips",
                    "SuccessTips": 1,
                    "FailTips" :   2
                }],
    "RspMsg" : "get_VoiceTips"
}
```

4.5.10.10 终端语音--设置(VoiceTips)

参数 **Data:** “VoiceTips”

字段	类型	长度	必须	描叙
VoiceTips	String	32	Y	终端语音
SuccessTips	int		Y	识别成功[范围: 1, 2, 3, 4, 5, 6, 7, 8] 1: 识别成功 2: 请通行 3: 验证通过 4: 欢迎光临 5: 欢迎回家 6: 请慢走 7: 一路平安 8: 请戴安全帽
FailTips	int		Y	识别失败[取值范围： 1, 2] 0: 未授权 1: 陌生人

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "VoiceTips": {
            "SuccessTips": 2
            "FailTips": 1,
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_VoiceTips"
}
```

4.5.10.11 产品名称--获取(ProductNameSet)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 ProductNameSet

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 ProductNameSet
UIName	String	100	Y	产品名称

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "ProductNameSet"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "ProductNameSet",
                "UIName": "动态人脸识别终端 005"
            }],
    "RspMsg" : "get_ProductNameSet"
}
```

4.5.10.11 产品名称--设置(ProductNameSet)

参数 **Data:** “ProductNameSet”

字段	类型	长度	必须	描叙
ProductNameSet	String	32	Y	产品名称设置
UIName	String	100	Y	产品名称

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "ProductNameSet": {
            "UIName": "动态人脸识别终端"
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_ProductNameSet"
}
```

4.5.10.12 通行模式--获取 (OpenDoorModeSet)

方向： Server >> Device

message: get_configs

请求：data: {}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字 :为 OpenDoorModeSet，

响应 data: {}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 OpenDoorModeSet
OpenDoor	int		Y	0：刷脸模式， 1：人卡合一模式， 2：刷卡模式， 3：人证模式， 4：人证白名单模式, 5: 人脸或卡 6: 人脸或人证
在“人卡合一模式” 下额外有以下字段				
CardVerify	int		Y	校验模式[取值范围:0,1,2] 0: 不限制 1: 先脸后卡 2: 先卡后脸
SwipeCardTip	bool		Y	刷卡提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
SwipeFaceTip	bool		Y	刷脸提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
在“人证模式”下额外有以下字段				
IdCardVerify	int		Y	校验模式[取值范围:0,1,2] 0: 不限制 1: 先脸后证 2: 先证后脸
SwipeIdCardTip	bool		Y	刷证提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
SwipeFaceTip	bool		Y	刷脸提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            "OpenDoorModeSet"
        ]}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data":[{
        "groupKey": "OpenDoorModeSet",
        "OpenDoor": 1,
        "CardVerify":0,
         "SwipeCardTip":0,
         "SwipeFaceTip":0
     }],
    "RspMsg": "get_OpenDoorModeSet"
}
```

扩展说明：若**要 1 次，读取多个组的配置**，可照如下方式，添加多个组的对应关键字（即 **groupKey**），具体如下：

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            "version",
            "OpenDoorModeSet"
         ]}
}
```

4.5.10.12 通行模式--设置 (OpenDoorModeSet)

方向： Server >> Device

message: config_system

参数 data: {"":""}

参考请求配置消息，支持请求配置中的子配置的单独设置

返回 data: {}

参数 Data: "OpenDoorModeSet"

OpenDoorModeSet： （请参照协议示例）：

字段	类型	长度	必须	描叙
OpenDoor	int		Y	通行模式[取值范围：0，1，2，3，4，5，6] 0：刷脸模式 1：人卡合一模式 2：刷卡模式 3：人证模式 4：人证白名单模式 5：人脸或卡 6: 人脸或人证
在“人卡合一模式” 下额外有以下字段				
CardVerify	int		Y	校验模式[取值范围:0,1,2] 0: 不限制 1: 先脸后卡 2: 先卡后脸
SwipeCardTip	bool		Y	刷卡提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
SwipeFaceTip	bool		Y	刷脸提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
在“人证模式”下额外有以下字段				
IdCardVerify	int		Y	校验模式[取值范围:0,1,2] 0: 不限制 1: 先脸后证 2: 先证后脸
SwipeIdCardTip	bool		Y	刷证提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音
SwipeFaceTip	bool		Y	刷脸提示音[取值范围:0,1] 0：打开提示音 1：关闭提示音

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{
    "message":"config_system",
    "data":{
        "OpenDoorModeSet":{
            "OpenDoor":1,
            "CardVerify":0,
            "SwipeCardTip":0,
            "SwipeFaceTip":0
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": {},
    "RspMsg": "config_OpenDoorModeSet"
}
```

4.5.10.15 陌生人--获取(StrangerParam)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 StrangerParam

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 StrangerParam
strangerIdentifyEn	bool		Y	识别使能[取值范围：0, 1] false: 不使能 true: 使能
strangerOpenDoorEn	bool		Y	允许通行[取值范围：0, 1]

				false: 不允许 true: 允许

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "StrangerParam"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "StrangerParam",
                "strangerIdentifyEn": true,
                "strangerOpenDoorEn" : true
            }],
    "RspMsg" : "get_StrangerParam"
}
```

4.5.10.15 陌生人--设置(StrangerParam)

参数 **Data:** “StrangerParam”

字段	类型	长度	必须	描叙
StrangerParam	String	32	Y	陌生人
strangerIdentifyEn	bool		Y	识别使能[取值范围：0, 1] false: 不使能 true: 使能
strangerOpenDoorEn	bool		Y	允许通行[取值范围：0, 1] false: 不允许 true: 允许

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "StrangerParam": {
            "strangerOpenDoorEn": false,
            "strangerIdentifyEn": true
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_StrangerParam"
}
```

4.5.10.16 HTTP 推送--获取(HttpPush)

方向： Server >> Device

message: get_configs

参数 data: {“”:””}

data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 HttpPush

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 HttpPush
httpPushEnable	bool		Y	启用: true: 启用 false: 不启用
httpServerAddr	String	64	Y	服务器地址
httpServerAddrStandby	String	64	Y	服务器备用地址
httpServerPort	Int		Y	服务器端口[取值范围：0~65535]
timeout	Int		Y	超时时间(s) [取值范围：0~1000]
charcode	Int		Y	字符编码[取值范围：0] 0: UTF-8
heartbeat_interval	Int		Y	心跳间隔(s) [取值范围：0~65535]
private_protocol	Int		Y	私有协议[取值范围：0~65535]
sslenable	bool		Y	ssl 连接启用 true: 启用 false：不启用
sslport	Int		Y	ssl 端口[取值范围：0~65535]
appid	String	127	Y	
appSecret	String	127	Y	
AuthSn	String	32	Y	

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "HttpPush"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                "groupKey": "HttpPush",
                "httpPushEnable": false,
                "httpServerAddr" : "192.168.1.235" ,
                "httpServerAddrStandby" : "192.168.1.240",
                "httpServerPort" : 80,
                "timeout": 4，
                "charcode": 0,
                "heartbeat_interval" : 30,
                "private_protocol":    0，
                "sslenable": false,
                "sslport": 4，
                "appid": 0,
                "appSecret" : 30,
                "AuthSn":    ""
            }],
    "RspMsg" : "get_configs"
}
```

4.5.10.16 HTTP 推送--设置(HttpPush)

参数 **Data:** “HttpPush”

字段	类型	长度	必须	描叙
HttpPush	String	32	Y	当前时间
httpPushEnable	bool		Y	启用: true: 启用 false: 不启用
httpServerAddr	String	64	Y	服务器地址
httpServerAddrStandby	String	64	Y	服务器备用地址
httpServerPort	Int		Y	服务器端口[取值范围：0~65535]
timeout	Int		Y	超时时间(s) [取值范围：0~1000]
charcode	Int		Y	字符编码[取值范围：0] 0: UTF-8
heartbeat_interval	Int		Y	心跳间隔(s) [取值范围：0~65535]
private_protocol	Int		Y	私有协议[取值范围：0~65535]
sslenable	bool		Y	ssl 连接启用 true: 启用 false：不启用
sslport	Int		Y	ssl 端口[取值范围：0~65535]
appid	String	127	Y	
appSecret	String	127	Y	
AuthSn	String	32	Y	

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "HttpPush": {
            "httpPushEnable": true,
            "httpServerAddr": "192.168.2.8",
            "httpServerAddrStandby": "192.168.2.1",
            "httpServerPort": 80,
            "timeout": 5,
            "charcode": 0,
            "heartbeat_interval": 0,
            "private_protocol": 0,
            "sslenable": false,
            "sslport": 443
            "appid": "12355",
            "appsecret": "afaf",
            "AuthSn": "fafafa",
            }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_HttpPush"
}
```

4.5.10.17 系统重启--获取(system_reboot)

方向： Server >> Device

message: get_configs

参数 data: {“groupKey”:[]}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 system_reboot

返回 data:{ }

字段	类型	长度	必须	描叙
groupKey	String	32	Y	本组关键字：为 system_reboot
auto_reboot	bool		Y	自动重启系统: true：自动重启 false：不自动重启
time	String	8	Y	启动时间：例如 03:02:01
week	int		Y	星期[范围：[0, 1, 2, 3, 4, 5, 6, 7] 0： 每天 1： 星期天 2： 星期一

				3： 星期二 4： 星期三 5： 星期四 6： 星期五 7： 星期六
log_level				日志级别[范围：[0, 1, 2] 0：最多 1：中等 2：最少

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
    "groupKey":[
            "system_reboot"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": [{
                    "groupKey": "system_reboot",
                    "auto_reboot": false,
                    "time" :    "03:02:00",
                    "week" :    5,
                    "log_level" : 0
                }],
    "RspMsg" : "get_system_reboot"
}
```

4.5.10.17 系统重启--设置(system_reboot)

参数 Data: “system_reboot”

字段	类型	长度	必须	描叙
system_reboot	String	32	Y	当前时间
auto_reboot	bool		Y	自动重启系统: true: 自动重启 false: 不自动重启
time	String	8	Y	启动时间：例如 03:02:01
week	int		Y	星期[范围：[0, 1, 2, 3, 4, 5, 6, 7] 0： 每天 1： 星期天 2： 星期一 3： 星期二 4： 星期三 5： 星期四 6： 星期五 7： 星期六
log_level				日志级别[范围：[0, 1, 2] 0：最多 1：中等 2：最少

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{   "message": "config_system",
    "data": {
        "system_reboot": {
            "week": 4,
            "auto_reboot": true,
            "log_level": 0,
            "time": "03:00:00"
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "操作成功",
    "data": {},
    "RspMsg": "set_system_reboot"
}
```

4.5.10.18 识别结果上报配置--获取 (VerifyRetRspSet)

方向： Server >> Device

message: get_configs

请求：data: {}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 VerifyRetRspSet

响应 data: {}

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 VerifyRetRspSet
SnapshotPic	bool		Y	false：不带抓拍照片，true： 带抓拍照片

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            " VerifyRetRspSet"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data":[{
        "groupKey": "VerifyRetRspSet",
        "OpenDoor": 1
     }],
    "RspMsg": "get_ VerifyRetRspSet"
}
```

4.5.10.18 识别结果上报配置--设置 (VerifyRetRspSet)

方向： Server >> Device

message: config_system

参数 data: {"":""}

参考请求配置消息，支持请求配置中的子配置的单独设置

返回 data: {}

参数 Data: "VerifyRetRspSet"

VerifyRetRspSet：（请参照协议示例）：

字段	类型	长度	必须	描叙

SnapshotPic	bool		Y	false：不带抓拍照片，true： 带抓拍照片

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

Request:

```
{
    "message":"config_system",
    "data":{
        "VerifyRetRspSet":{
            "SnapshotPic": true
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": {},
    "RspMsg": "config_VerifyRetRspSet "
}
```

4.5.10.19 身份证号限制--获取 (ForbidIdCardSet)

方向： Server >> Device

message: get_configs

请求：**data: {}**

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 ForbidIdCardSet

响应 **data: {}**

字段	类型	长度	必须	描叙
groupKey	String Array	32	Y	本组关键字：为 ForbidIdCardSet
NumPrefix	String Array	12	Y	受限的身份证号前缀数组

示例：

Request:

```
{
    "message":"get_configs",
    "data":{
        "groupKey":[
            "ForbidIdCardSet"
        ]}
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data":[{
        "groupKey": "ForbidIdCardSet",
        "NumPrefix":["44","4405"]
     }],
    "RspMsg": "get_ ForbidIdCardSet"
}
```

4.5.10.19 身份证号限制设置(ForbidIdCardSet)

方向： Server >> Device

message: config_system

参数 data: {"":""}

参考请求配置消息，支持请求配置中的子配置的单独设置

返回 data: {}

参数 Data: "ForbidIdCardSet"

ForbidIdCardSet：（请参照协议示例）：

字段	类型	长度	必须	描叙
NumPrefix	String Array	12	Y	受限的身份证号前缀数组

返回 data:{ }

字段	类型	长度	必须	描叙
RspMsg	String	32	N	与请求字段对应

示例：

```
{
    "message":"config_system",
    "data":{
        "ForbidIdCardSet":{
            "NumPrefix":["44","4405"]
        }
    }
}
```

Response:

```
{
    "result": 0,
    "desc": "Success",
    "data": {},
    "RspMsg": "config_ForbidIdCardSet"
}
```

4.5.11 设置重推识别结果（reSend_record_set)

方向： Server >> Device

message: reSend_record_set

请求：data: {}

字段	类型	长度	必须	描叙
recordType	int	4	Y	识别记录类型 0:识别失败，1：识别成功，2：所有记录
snapStartTime	String Array	20	Y	起始时间 2021-01-01 08:30:20
snapEndTime	String Array	20	Y	结束时间 2021-01-02 08:30:20

示例：

Request:

```
{
    "message":"reSend_record_set",
    "data":{
    "recordType":1,
    "snapStartTime":"2021-01-01 08:30:20",
    "snapEndTime":"2021-01-02 08:30:20"
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{}
}
```

4.6设备上报消息

4.6.1 上报识别结果（snapshot_face）

方向： Device >> Server

message: snapshot_face

参数 data:

字段	类型	长度	必须	描叙
personId	String	32	Y	抓拍人员的 ID，ID 是添加人员时平

				台下发的，如果人员不是平台添加，可以不用
faceId	String	32	Y	对比头像的 ID，ID 是添加照片时平台下发的，如果照片不是平台添加，可以不用
fullname	String	24	Y	Utf8 编码，人员姓名，如果没指定 personId，必须指定姓名
type	Integer		Y	人员类型，1：白名单，2：黑名单，3：访客
idNum	String	20	Y	身份证号码，如：，如 420381200010013857
gender	String	32	N	Utf8 编码，性别，人证设备必填
ethnic	String	32	N	Utf8 编码，民族，人证设备必填
birthday	String	32	N	Utf8 编码，证件上出生日期，人证设备必填
address	String	256	N	Utf8 编码，证件上地址，人证设备必填
expiredate	String	32	N	Utf8 编码，证件过期时间，人证设备必填
icNum	String	8	Y	人员 IC 卡号，如：157342643
comparePic	String		Y	对比图片的 Base64 数据 ,如果没指定 faceId，必须指定，如果指定了 faceId，将忽略
snapPic	String		Y	抓拍图片的 Base64 数据
score	Integer		Y	对比分数，0~100
result	Integer		Y	对比结果， 0 失败(不匹配)， 1 成功（匹配、未过期）， 2：比对成功，授权已经过期 3：比对成功，授权未过期但不在通行时段内
snaptime	String	20	Y	抓拍时间，字符串格式，如“2018-11-16 11:05:00”
bodyTemp	Float		Y	人体温度 如 "36.7"

返回 data：{}

字段	类型	必须	描叙

snaptime	String	N	抓拍时间，字符串格式，如“2018-11-16 11:05:00”，可以不返回
opendoor	int	N	开闸控制，0：关闸，1：开闸,不控制闸机，不要返回这个信息
screenDisp	String	N	屏幕显示内容,如"余额不足",最多 20 个字节，不显示屏幕，不要返回这个信息

示例：

Request:

```
{
“message”: “snapshot_face”,
“data”:
    {
    “personId”: “5907d9b6bede435faf0828177e5bce07”,
    “faceId”: “cf78bfabb138474c8b2b3a0077c9bec1”,
    “fullname”: “韩梅梅”,
    “type”:1,
    “idNum”: “430726199801042312”,
    “icNum”: “123475",
    “snapPic”:“/9j/2wBDABQODxIPDRQSEBIXFRQYHjIhHhwcHj0sLiQy
”,
    “score”:92,
    “result”:1,
    “snaptime”: “2019-08-02 15:42:48”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{
        “snaptime”:"2018-11-16 16:10:00",
    }
}
```

4.6.2 设备时间同步(sync_time)

方向： Device >> Server

设备与平台建立连接后，会主动发起时间对时，5 分钟后会再次同步；

message: sync_time

参数 data: { ”timestamp”:””}

字段	类型	长度	必须	描叙
timestamp	String	20	Y	设备端当前时间,格式:yyyyMMddHHmmssSSS,精确到毫秒

返回 data：{}

示例：

Request:

```
{
    “message”: “sync_time”,
    “data”:{
        “timestamp”: “20190531145826123”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{
        “timestamp”: “20190531145826456”
    }
}
```

timestamp 服务器返回的时间,格式:yyyyMMddHHmmssSSS,精确到毫秒

4.6.3 设备上报二维码信息(qrcode_data)

设备上报二维码数据，平台通过此消息来确认是否开关门。

方向： Device >> Server

message: qrcode_data

参数 data: { ”content”:””}

字段	类型	必须	描叙
content	String	Y	二维码数据(不固定长度)

返回 data：{}

字段	类型	必须	描叙
opendoor	int	Y	开闸：1 关闸：0

示例：

Request:

```
{
    “message”: “qrcode_data”,
    “data”:{
        “content”: “adddfffsss”
    }
}
```

Response:

```
{
    “result”: 0,
    “desc”: “Success”,
    “data”:{
        “opendoor”: 1
    }
}
```

4.7可视对讲（只有可视对讲设备才支持）

4.7.1 设备通过房号获取呼叫参数（device_call_param）

通过第三方 SDK 进行呼叫时,获取呼叫参数,包括登录第三方的 TOKEN、呼叫对端 TOKEN、手机号码等

方向：Device >> Server

message: device_call_param

协议版本号 = 5

Request:

```
{
        "message":"device_call_param",
        "data":{
            "callNo":"304",
            "callRequestId":"",
            "picture":""
        }
}
```

字段	类型	必须	描叙
callNo	string	Y	呼叫编号(房号)
callRequestId	string	Y	呼叫请求 ID(使用 uuid),每次请求唯一。
picture	string	Y	呼叫时设备端拍摄的前端照片,JPG 格式图片 BASE64 编码

Response:

```
{
        "message":"device_call_param",
        "result":0,
        "data":{
            "callRequestId":"",
            "callNo":"",
            "deviceId":"",
            "deviceToken":"",
            "deviceName":"",
            "callMaxSeconds":60,
            "calleds":[
                {
                    "userId":"",
                    "token":"",
                    "phoneNo":""
                },
                ......
            ]
        }
}
```

字段	类型	必须	描叙
result	int	Y	结果 0：成功 104：呼叫编号错误（房号不正确） 105：无人接听
callRequestId	string	Y	呼叫请求 ID,原样返回
callNo	string	Y	呼叫编号（房号）
deviceId	string	Y	设备 ID
deviceToken	string	Y	第三方呼叫平台主叫方设备 TOKEN
deviceName	string	Y	设备名称,通常用于在 APP 端的显示呼叫位置
callMaxSeconds	int	Y	最长通话多少秒
called	array	N	被叫方参数
userId	string	Y	被叫方用户 ID
token	string	N	被叫方 TOKEN(第三方 TOKEN),有可能为空,为空时表示该用户没有登录过 APP,无

			法进行可视对讲呼叫。
phoneNo	string	N	被叫方手机号

4.7.2 设备呼叫结果上报(device_call_result)

呼叫行为发生后,上报呼叫结果（统计数据）

方向：Device >> Server

message: device_call_result

协议版本号 = 5

Request:

```
{
        "message":"device_call_result",
        "data":{
            "callNo":"",
            "callRequestId":"",
            "callRecords":
                [
                    {
                        "userId":"",
                        "callResult":0/1/2/3/...,
                        "callType":0/1/2,
                        "openStatus":0/1/2,
                        "startTime":"",
                        "endTime":""
                    },
                    ......
                ]
        }

}
```

字段	类型	必须	描叙
callNo	string	Y	呼叫编号(房号)
callRequestId	string	Y	呼叫请求 ID,与 device_call_param 中的 callRequestId 配对。 同一个呼叫请求,可以上传多个呼叫结果。
callRecords	object	Y	呼叫信息
userId	string	N	PSTN 串呼时可以提供,其它情况为空
callResult	int	Y	呼叫结果 0：未知 1：设备端挂断（还未接通时挂断） 2：呼叫功能异常 3：APP 呼叫成功（有一人接听）

			4：APP 不在线（所有用户不在线） 5：APP 呼叫失败（所有用户未接或拒绝） 6：APP 呼叫异常 7：PSTN 呼叫成功（有一个接听） 8：PSTN 呼叫失败（未接或拒绝） 9：PSTN 异常
callType	int	Y	呼叫类型 0：未接通 1：APP 2：PSTN(落地电话）
openStatus	int	Y	0 未开门 1 开门成功 2 开门失败
startTime	string	Y	接通开始时间:yyyyMMddHHmmss
endTime	string	Y	挂断时间:yyyyMMddHHmmss

Response:

```
{
        "message":"device_call_result",
        "result":0
}
```

1 错误状态码

状态码	描述信息
-1	操作异常
0	操作成功
100	请重新注册
101	对象已存在
102	对象不存在
103	命令不存在
104	参数无效
105	用户不存在
106	密码错误
107	超时
108	添加用户失败

109	系统忙，禁止操作，在返回的 data 属性中可以定义 waitSeconds（请等待多少秒），如
	{
	“result”: 109,
	“desc”:“Success”,
	“data”:{
	“waitSeconds”:3

	} }
110	图片提取特征失败
200	无效的 url
201	url 无法连接
202	升级文件下载失败
203	升级文件校验失败