

# 车牌识别一体机 TCP 通讯协议说明手 册

(版权所有,翻版必究)

非常感谢您使用我们公司的产品，我们将竭诚为您提供最好的服务。  
本手册可能包含技术上不准确的地方或文字错误，欢迎您的纠正。  
本手册内容将做定期的更新，更新内容将在本手册的新版本中加入。  
我们随时会改进或更新本手册中描述的产品或程序。

## 目录

1. 文档说明 .....	1
2. 修订记录 .....	1
3. 协议说明 .....	2
4. 基本说明 .....	2
4.1. 收发命令包与数据包格式 .....	2
4.2. 具体命令 .....	3
4.2.1. 数据类型说明 .....	3
4.2.2. 协议通用格式 .....	3
5. Tcp 协议命令 .....	5
5.1. 设备维护管理 .....	5
5.1.1. 恢复设备默认配置: set_factorydefault .....	5
5.1.2. 获取设备的序列号: getsn .....	5
5.1.3. 获取设备的硬件版本信息: get_hw_board_version .....	6
5.1.4. 获取设备当前时间戳: get_device_timestamp .....	7
5.1.5. 设置系统时间: set_time .....	7
5.1.6. 设置网络参数: set_networkparam .....	7
5.1.7. 获取网络参数: get_networkparam .....	8
5.1.8. 设置中心服务器网络参数: set_centerserver_net .....	9
5.1.9. 设置当前配置为用户默认配置: set_user_default_cfg .....	9
5.1.10. 修改设备 admin 密码: set_adminpass .....	10
5.1.11. 重启设备: reboot_dev .....	11
5.1.12. 建安顺 APP 过期设置: jasauthtime, jasauthtime_rsp .....	11
5.1.13. 对 DM8127 的单板: 获取降噪模式和降噪强度: get_denoise .....	12
5.1.14. 对 DM8127 的单板: 设置降噪模式和降噪强度: set_denoise .....	12
5.1.15. 开始自动聚焦: 不支持 .....	13
5.1.16. 停止自动聚焦: 不支持 .....	14
5.1.17. 获取设备版本信息: get_product_info .....	14
5.1.18. 获取设备 4G 参数信息: get_4g_param .....	15
5.1.19. 设置设备 4G 参数信息: set_4g_param .....	16
5.1.20. 设置设备名称: set_dev_name .....	17
5.1.21. 设置 ftp 服务启用状态: set_vsftpd_enable .....	17
5.1.22. 获取 ftp 服务启用状态: get_vsftpd_enable .....	18
5.1.23. 修改账号密码(linux 账号和 ftpserver 对应的账号) .....	18
5.2. 车牌识别 .....	19
5.2.1. 配置推送数据方式: ivsresult .....	19
1.1.1. 配置快速推送数据方式: quick_ivsresult 【仅 E5 2.0 支持】 .....	24
5.2.2. 获取最近一次识别结果: getivsresult .....	26
5.2.3. 手动触发车牌识别: trigger .....	26
5.2.4. 获取记录最大 id: get_max_rec_id .....	26
5.2.5. 获取历史记录: get_record .....	27
5.2.6. 获取记录图片: get_image、get_offline_image .....	27
5.2.7. 抓取当前图片 get_snapshot .....	28
5.2.8. 获取视频播放的 uri: get_rtsp_uri .....	28
5.2.9. 获取虚拟线圈参数: get_virloop_para .....	29
5.2.10. 设置虚拟线圈参数: set_virloop_para .....	30
5.2.11. 获取虚拟线圈属性: get_virloop_prop .....	31
5.2.12. 获取识别区域参数: get_reco_para .....	32
5.2.13. 设置识别区域参数: set_reco_para .....	33

5.2.14. 获取识别区域属性: get_reco_prop .....	34
5.2.15. 获取算法识别参数: get_alg_result_para .....	35
5.2.16. 设置算法识别参数: set_alg_result_para .....	36
5.2.17. 获取算法识别参数属性: get_alg_result_prop .....	36
5.2.18. 设置 LED 参数: set_led_para .....	43
5.2.19. 获取 LED 参数: get_led_para .....	44
5.2.20. 获取 LED 属性: get_led_prop .....	44
5.2.21 获取道闸状态 barr_gate_status_notify .....	45
5.3. 设备硬件参数配置 .....	46
5.3.1. 设备端主动推送 Gpio 输入状态消息: gpio_trigger .....	46
5.3.2. 控制 IO 输出: ioctl, ioctl_resp .....	47
5.3.3. 获取 IO 输入状态: get_gpio_value .....	47
5.3.4. 获取 IO 输出状态: get_gpio_out_value .....	48
5.3.5. 自动聚焦: 不支持 .....	48
5.3.6. 获取存储设备信息: get_diskinfo .....	49
5.3.7. 获取串口参数: get_serial_para .....	50
5.3.8. 设置串口参数: set_serial_para .....	51
5.3.9. 设置客户定制 sn 序列号 .....	51
5.3.10. 获取客户定制 SN 序列号 .....	52
5.3.11. 设置 Gpio 口锁定状态 .....	52
5.3.12. 获取 gpio 口锁定状态 .....	53
5.3.13. 获取交通灯状态【SJ R5D 定制】 .....	53
5.3.14. 设置交通灯功能【SJ R5D 定制】 .....	54
5.3.15. 获取云存历史记录（如果没有上云查询本地存储记录） .....	54
5.4. 设备连接 .....	55
5.4.1. 通知在线消息: response_online, response_online_rsp .....	55
5.4.2. 主动断开连接消息: close_socket .....	56
5.5. 配置透明通道: ttransmission .....	56
5.6. 白名单 .....	58
5.6.1. 脱机检查 .....	58
5.6.2. 注册脱机功能: reg_offline_check .....	58
5.6.3. 取消脱机注册: reg_offline_check .....	58
5.6.4. 注册脱机事件: register_offline_event .....	59
5.6.5. 获取脱机状态: get_offline_status .....	59
5.6.6. 注册 OpenSDK 监听推送: register_push_channel .....	60
5.6.7. 请求 OpenSDK Push: push_msg_to_opensdk, 此协议不需要测试, 用于给 opensdk 发送透传消息的。 .....	60
5.6.8. 操作白名单: white_list_operator .....	61
5.6.9. 增加或者更新白名单: update_or_add .....	61
5.6.10. 以车牌号查询白名单: plate .....	62
5.6.11. 白名单删除: delete .....	64
5.7. 识别结果加密 .....	64
5.7.1. 获取加密方式: get_ems .....	65
5.7.2. 获取用户密码: get_encrypt_key .....	66
5.7.3. 重新设置用户密码: reset_encrypt_key .....	67
5.7.4. 修改用户密码: change_encrypt_key .....	68

5.7.5. 开启是否加密: enable_encrypt .....	69
5.7.6. 设置设备有效时间: device_active_settings .....	70
5.7.7. 获取设备有效时间: device_active_settings .....	71
5.7.8. 设置用户私有数据: set_user_data .....	71
5.7.9. 获取用户私有数据: get_user_data .....	72
5.8. 设备组网: dg_json_request .....	72
5.8.1. 得到当前设备 vzid: get_cdvzid .....	73
5.8.2. 得到当前设备名称: get_current_device_name .....	74
5.8.3. 得到在线设备信息, 不含自己: get_ovzid .....	75
5.8.4. 得到在线设备信息, 含自己: online_devices .....	76
5.8.5. 得到所有连接设备信息: get_avzid .....	77
5.8.6. 得到当前组网内所有设备信息: get_agdi .....	77
5.8.7. 得到当前设备记录 size: current_records_size .....	78
5.8.8. 得到入口设备记录: records_sparate_input .....	79
5.8.9. 得到出口设备记录: records_sparate_output .....	82
5.8.10. 使能设备组网: enable_devicegroup .....	85
5.8.11. 设置设备组网类型及参数: set_device_type_enable .....	86
5.8.12. 查找车牌信息: search_plate .....	87
5.8.13. 获取组网匹配模式: get_device_match_mode .....	89
5.8.14. 设置组网匹配模式: set_device_match_mode .....	90
5.8.15. 得到组网共享 IO: get_group_shared_io .....	90
5.8.16. 设置组网共享 IO: set_group_shared_io .....	91
5.8.17. 组网识别结果消息 enable_dg_result .....	91
5.8.18. 清除组网数据: reset_database .....	95
5.8.19. 得到当前组网内部所有设备配置: get_group_cfg .....	96
5.8.20. 设置当前组网内部所有设备配置: set_group_cfg .....	101
5.8.21. 根据 ID 获取组网图片: get_img_by_id .....	101
5.9. 用户登录 .....	103
5.9.1. 请求开始登录:start_login .....	103
5.9.2. 登录认证: login_authentication .....	104
5.10. 语音协议 .....	104
5.10.1. 获取当前语音文件列表: playserver_json_request .....	105
5.10.2. 设置语音默认参数: 不支持 .....	106
5.10.3. 获取语音默认参数: 不支持 .....	106
5.10.4. 播放语音参数: playserver_json_request .....	107
5.10.5. 请求语音对讲: start_talk .....	107
5.10.6. 刷新当前语音文件列表: playserver_json_request .....	109
5.10.7. 播放指定路径的音频: playserver_json_request .....	110
5.10.8. 播放 tts 文本内容: playserver_json_request .....	110
5.11. 屏显协议: R5D R52.0 RX5 支持 .....	111
5.11.1. 设置 LED 显示内容: set_led_show .....	111
5.11.2. 获取 LED 显示内容: get_led_show .....	112
5.11.3. 获取 LED 数据传输使用串口号: get_led_serial_port .....	113
5.11.4. 设置 LED 数据传输使用串口号: set_led_serial_port .....	114

5.11.5. 字段信息以及相关枚举 .....	114
5.12. LCD 配置: R5D R52.0 RX5 支持 .....	116
5.12.1. 设置 LCD 广告音量 .....	116
5.12.2. 查询 LCD 广告音量 .....	117
5.12.3. 设置 LCD 背光亮度 .....	118
5.12.4. 查询 LCD 背光参数 .....	119
5.12.5. 设置 LCD 忙时半屏显示 .....	120
5.12.6. 设置 LCD 闲时下半屏显示 .....	121
5.12.7. 设置 LCD 全屏忙时显示 .....	121
5.12.8. 设置 LCD 全屏广告显示 .....	122
5.13. 云台控制: R5D R52.0 RX5 支持 .....	128
5.14. 设置用户自定义 OSD (视频) .....	129
5.15. 图片 osd: 仅 E5 支持 .....	131
5.15.1. 设置图片 osd: AVS_SET_PIC_OSD_PRM .....	131
5.15.2. 获取图片 osd 配置: AVS_GET_PIC_OSD_PRM .....	140
5.16. 用户自定义设备信息 (R3 不支持) .....	142
5.16.1. 获取自定义信息: get_custom_user_info .....	142
5.17. 用户通过 R3 获取伴侣机 Rg 的相关配置信息 .....	145
5.17.1. 通过 R3 获取指定伴侣机的配置信息: user_get_mate_info .....	145
5.17.2. 通过 R3 获取伴侣机的 Rtsp 代理信息: user_get_rtsp_info .....	146
5.17.3. 通过 R3 获取伴侣机的语音代理信息: user_request_talkback .....	146
5.17.4. 通过 Tcp 获取日志指定的日志: log_search .....	148
5.17.5. 设置 Rg 人车滞留事件上报使能: stay_event_enable .....	149
6. 码表 .....	153
6.1. Gpio 状态码表 .....	153

# 1. 文档说明

除特别说明内容，本文档内容适用于车牌识别一体机所有 R 系列设备。

本文档对设备提供的 TCP 接口能力进行定义及描述，便于开发技术人员通过此文档开发程序与设备进行对接。

# 2. 修订记录

版本号	拟制日期	版本描述	存档编号
1.0.17	2025.3.28	增加 LCD 屏显自由模式	
1.0.16	2024.08.12	新增语音协议 5.10.8 播放 tts 文本内容	
1.0.15	2024.01.06	新增"error_msg"、"state"、"response"字段说明	
1.0.14	2023.01.05	修改查找车牌信息:search_plate 返回消息的结构定义	
1.0.13	2022.06.01	LCD 屏显控制命令	
1.0.12	2021.11.29	增加交通灯控制协议 Ftpserver 启停命令 语音列表刷新 播放指定路径的 wav 音频文件	
1.0.11	2021.6.21	增加设置获取客户定制 sn 增加设置获取 gpio 加锁状态	
1.0.10	2018.10.23	1. 增加 NTP 配置获取; 2. 增加 NTP 配置设置;	
1.0.9	2017.03.29	1. 增加获取/设置用户自定义数据 2. 增加获取/设置串口参数 3. 增加获取 IO 输出状态 4. 获取设备存储信息	
1.0.8	2017.12.12	1. 增加算法识别参数的获取/设置 2. 增加算法识别属性的获取 3. 增加 LED 控制参数的设置/获取 4. 增加 LED 控制属性的获取 5. 增加恢复设备默认配置	
1.0.7	2017.11.24	1. 增加设置/获取虚拟线圈参数 2. 增加设置/获取识别区域参数	
1.0.6	2017.10.16	1 设备维护管理相关 2 车牌识别中获取视频播放 uri 3 注册脱机事件 4 设备组网相关 5 用户登录相关	
1.0.5	2017.6.16	1 清除组网数据 2 得到当前组网内部所有设备配置 3 获取组网图片	
1.0.4	2016.6.13	1 增加设置、获取网络参数 2 增加设置中心服务器配置	

1.0.3	2016.3.25	Tcp 协议纠错	
1.0.2	2016.2.16	1 新增 Linux 环境下 java 的 Demo 2 从此版本开始，版本号以三位整数进行发布	
1.0.0.1	2016.1.12	1 开发手册对接口说明，进行分类，方便用户查看	
1.0.0.2	2023.2.23	删除 colorValue 字段	
1.0.0.2	2023.10.26	增加 OSD 设置	
1.0.0.3	2024.02.28	获取存储设备信息 devtype 字段增加 emmc 类型	
1.0.0.4	2024.03.01	更新白名单时间段字段 seg_time_start,seg_time_end，删除 seg_time 字段	
1.0.0.5	2024.03.11	更新 ivsresult 返回结果示例及字段描述	
1.0.0.6	2024.05.11	增加道闸状态	
1.0.0.7	2024.06.05	设置图片 OSD 协议支持动态叠加	
1.0.0.8	2024.06.06	增加道闸连接状态，功能开关状态上报	
1.0.0.9	2024.08.23	5.2.18.设置 LED 参数：set_led_para 小节 LED 模式配置中 删除"1 常亮" 选项	
1.0.0.10	2024.12.5	5.1.3 添加与鸿蒙相关的产品系列字段	
1.0.0.11	2024.12.24	5.17 增加防回退协议	
1.0.0.12	2025.01.21	新增车辆颜色（米、橙、紫），详细定义见 VzClientSDK_LPDefine.h	

### 3. 协议说明

Tcp 协议主要用于用户配置摄像机参数以及获取识别结果等操作。

### 4. 基本说明

设备作为服务端，通过 TCP 协议将识别结果发送到上位机；上位机作为客户端，使用普通 socket 与设备通信。服务端监听端口默认为 8131。

客户端先使用 socket 主动连接 tcp server 服务端（端口号 8131），连接成功后，便可使用如下一系列的 tcp 协议操作我们的设备，以满足用户的需求。

#### 4.1. 收发命令包与数据包格式

0	1	2	3	4	5	6	7	标尺
‘V’	‘Z’	包类型	包序号	包数据长度				数据头
Data								数据

上表是一个标准的 TCP 数据包的格式,数据主要分为两部分:一个 8 个字节的数据包头。接下来是数据,数据长度在前面的包头里面。

接下来我们重点介绍数据包头,数据包头一共 8 个字节,其含义如下:

- 字节位 0、ASCII 字符'V', 整数 86。
- 字节位 1、ASCII 字符'Z', 整数 90。
- 字节位 2、Data 数据类型
  - 0X00 数据包。
  - 0X01 心跳包, 数据长度为 0, DATA 为空
- 字节位 3、包序号, 包序列号为递增的数值, 用于对应请求命令与返回命令; 例如发送一个请求命令到服务器端, 服务器端在返回结果时, 会将请求命令中的包序列号填充到返回数据包的包序列号中, 便于客户端这边将返回结果与请求命令进行对应。如果不存在对应问题, 则设置为 0 即可。如果编号到 255, 则从 0 开始重新编号。
- 字节位 4-7、一共四位, 代表接下来的数据长度。这个数据是网络字节序, 在接收的时候务必调用 htonl 这样的函数将网络字节序转换成主机字节序。数据的长度不要大于  $1024 * 1024 = 1\text{MB}$ 。服务器将不会接收超过 1MB 大小的数据, 同样服务器也不会发送超过 1MB 大小的数据包。理论上, 除了心跳包之外, 其它所有数据的长度都不会是 0。
- DATA 数据位、根据前面的包头, 有不同的数据。

## 4.2. 具体命令

命令采用 json 格式, 注意不论请求还是回复, 所有的 JSON 字符全部使用小写字符。

### 4.2.1. 数据类型说明

此章节定义协议中用到的各类数据类型。

表 1 数据类型定义表

数据类型标识	备注
int	整数
uint	无符号整数
float	浮点数
double	浮点数
string(N)	字符串。其中 N 表示字符串的最大长度, 单位: 字节
base64	base64 编码格式的字符串
bool	布尔值: true, false
obj	JSON 对象
time_t	时间戳, 单位: 秒
msecond	时间戳, 单位: 毫秒
array	JSON 数组。数组中默认存放的是 obj, 若是其它类型需要备注说明, 如 array<float> 表示数组中存放的是浮点数数组。

### 4.2.2. 协议通用格式

表 2 请求协议定义表

层级	字段编码	数据类型	备注
	cmd	string	消息命令字, 标识当前属于哪种消息



	body	string	JSON 格式的字符串数据，此为消息数据内容
	id	string	消息序列号，标识具体的一次请求

表 2 应答协议定义表

层级	字段编码	数据类型	备注
1	cmd	string	与请求中的相应字段的值保持一致
2	id	string	与请求中的相应字段值保持一致，标识是对哪一次请求进行应答
3	state_code	int	错误码，标识操作是否成功。参见表 4
4	error_msg	string	失败或者成功的详细说明
5	state	int	同 state_code, 兼容其他协议的返回需要，有的协议之前用的是 state，这个是历史遗留问题，无法统一用一个。
6	response	string	返回说明，目前只有登录认证会用到，无加密的时候才会返回其他时候不会。

协议通用格式如下：

**request:**

```
{
  "cmd": "xxx"
  "body": "{}"
}
```

**response:**

```
{
  "cmd": "xxx",
  "id": "1234",
  "state_code": 200
  "body": "{}"
}
```

字段名	说明
cmd	命令字符串
body	传递的数据，可为空
state_code	返回结果状态，状态定义如下。 状态码定义：(参照 http 的返回状态码定义的) 200 请求的命令执行成功 400 客户端请求出错（因为错误的语法导致服务器无法理解请求信息） 401 客户没有权限，请求需要用户验证 404 请求的资源不存在 405 请求的命令不存在 408 请求超时 500 服务器内部错误

## 5. Tcp 协议命令

### 5.1. 设备维护管理

#### 5.1.1. 恢复设备默认配置: set\_factorydefault

请求命令参数

```
{
  "cmd": "set_factorydefault",
  "id": "1548",
  "body": {
    "factorydefault": 0
  }
}
```

结果返回

```
{
  "cmd": "set_factorydefault",
  "id": "1548",
  "state_code": 200
}
```

字段名	说明
factorydefault	恢复设备默认值 <ul style="list-style-type: none"> <li>0 完全恢复</li> <li>1 部分恢复</li> </ul>

#### 5.1.2. 获取设备的序列号: getsn

请求命令格式

```
{
  "cmd": "getsn",
  "id": "123456"
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "getsn",
  "id": "123456",
  "state_code": 200,
  "value": "XXXXXXXX-XXXXXXXX"
}
```

如果设备内部有问题, 则会返回如下数据:

```
{
  "cmd": "getsn",
```

```
{
  "value": "unknown"
}
```

字段名	说明
value	设备序列号：正确值为 17 位长的字符串，前 8 位 + '-' + 后 8 位

### 5.1.3. 获取设备的硬件版本信息: get\_hw\_board\_version

请求命令格式

```
{
  "cmd": "get_hw_board_version",
  "id": "12"
}
```

结果返回值

如果一切正常则返回如下数据：

```
{
  "body": {
    "board_version": 3435973836,
    "exdataSize": 3435973836,
    "product_series": "NA"
  },
  "cmd": "get_hw_board_version",
  "id": "12",
  "state_code": 200
}
```

如果设备内部有问题，返回如下数据：

```
{
  "body": {
    "board_version": ,
    "exdataSize": ,
    "product_series":
  },
  "cmd": "get_hw_board_version",
  "id": "12",
  "state_code": 500
}
```

字段名	说明
board_version	硬件版本号
exdataSize	TODO
product_series	产品系列(与鸿蒙版本相关)，默认 NA
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误（内部设置错误）</li> </ul>

### 5.1.4. 获取设备当前时间戳: get\_device\_timestamp

请求命令参数

```
{
  "cmd": "get_device_timestamp",
  "id": "999999"
}
```

结果返回

```
{
  "cmd": "get_device_timestamp",
  "id": "999999",
  "state_code": 200,
  "timestamp": 1443213456
}
```

字段名	说明
timestamp	时间 (格林威治时间, 单位秒)

### 5.1.5. 设置系统时间: set\_time

请求命令参数

```
{
  "cmd": "set_time",
  "id": "132156",
  "timestring": "2015-03-17 20:47:02"
}
```

结果返回

```
{
  "cmd": "set_time",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
state_code:	状态码 <ul style="list-style-type: none"> <li>200: 成功</li> <li>400: 客户端请求出错 (timestring 错误)</li> <li>500: 服务器内部错误 (内部设置错误)</li> </ul>
timestring	<ul style="list-style-type: none"> <li>时间字符串, 格式必须是: "XXXX-XX-XX XX:XX:XX"</li> </ul>

### 5.1.6. 设置网络参数: set\_networkparam

请求命令参数

```
{
  "cmd": "set_networkparam",
  "id": "132156",
  "body": {
```

```

"ip": "192.168.1.177",
"netmask": "255.255.255.0",
"gateway": "192.168.1.1",
"dns": "0.0.0.0",
"source": 0
}
}

```

#### 结果返回

```

{
"cmd": "set_networkparam",
"id": "132156",
"error_msg": "success",
"state_code": 200
}

```

字段名	说明
state_code	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 500: 服务器内部错误</li> </ul>
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	需要传递的网络数据（ip ,netmask ,gateway ,dns 字段均为可选的，即需要设置哪一项数据就列出哪一项）
ip	Ip 地址
netmask	子网掩码
gateway	网关
dns	dns 服务器
source	网口号 0 1：非必选，没有默认为 0，兼容老接口

### 5.1.7. 获取网络参数：get\_networkparam

#### 请求命令参数

```

{
"cmd": "get_networkparam",
"id": "132156",
"source": 0
}

```

#### 结果返回

```

{
"body": {
"dhcp_enable": 0,
"dns": "0.0.0.0",
"gateway": "192.168.3.1",
"ip": "192.168.3.48",
"netmask": "255.255.255.0"
},
"cmd": "get_networkparam",
"error_msg": "success",
"id": "132156",

```

```
{
  "state_code" : 200
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	获取到的网络数据
ip	Ip 地址
netmask	子网掩码
gateway	网关
dns	Dns 服务器
source	网口号 0 1：非必选，没有默认为 0，兼容老接口

### 5.1.8. 设置中心服务器网络参数：set\_centerserver\_net

请求命令参数

```
{
  "cmd" : "set_centerserver_net",
  "id" : "132156",
  "body":{
    "hostname":"192.168.1.106",
    "port":80,
    "enable_ssl":false,
    "ssl_port":443,
    "http_timeout":5
  }
}
```

结果返回

```
{
  "cmd" : "set_centerserver_net",
  "id" : "132156",
  "state_code" : 200
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递中心服务器参数数据
hostname	中心服务器地址
port	中心服务器端口
enable_ssl	开启 ssl 连接
ssl_port	ssl 端口
http_timeout	超时时间

### 5.1.9. 设置当前配置为用户默认配置：set\_user\_default\_cfg

请求命令参数

```
{
  "cmd" : "set_user_default_cfg",
}
```

```
"id": "132156"
}
```

**结果返回**

```
{
  "cmd": "set_user_default_cfg",
  "id": "132156",
  "error_msg": "success",
  "state_code": 200,
  "state": 200
}
```

字段名	说明
state_code	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>
state	<ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>
id	序列字符串 (唯一), 字符串长度 < 30, 等于请求时的 id
error_msg	错误信息, 失败时返回错误原因;

**5.1.10. 修改设备 admin 密码: set\_adminpass****请求命令参数**

```
{
  "cmd": "set_adminpass",
  "id": "132156",
  "body": {
    "old_pass": "asgwe4AGSAD45",
    "new_pass": "fdas213asfdgad"
  }
}
```

**结果返回**

```
{
  "cmd": "set_adminpass",
  "id": "132156",
  "state_code": 200,
  "error_msg": "success"
}
```

字段名	说明
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>

id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
old_pass	经过 64 位编码后的 Admin 旧密码；
new_pass	经过 64 位编码后的 Admin 新密码；

### 5.1.11. 重启设备:reboot\_dev

请求命令参数

```
{
  "cmd": "reboot_dev",
  "id": "132156"
}
```

结果返回且设备重启

```
{
  "cmd": "reboot_dev",
  "id": "132156",
  "state_code": 200,
}
```

字段名	说明
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误（内部设置错误）</li> </ul>
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id

### 5.1.12. 建安顺 APP 过期设置: jasauthtime, jasauthtime\_rsp

请求命令参数

```
{
  "cmd": "jasauthtime",
  "id": "132156",
  "year": 2017,
  "mon": 12,
  "day": 12,
  "hour": 13,
  "min": 14,
  "sec": 49
}
```

结果返回

当发送 CMD 为: jasauthtime\_rsp 的消息时有如下回复:

```
{
  "cmd": " jasauthtime_rsp ",
  "id": "132156",
  "state_code": 200,
}
```



字段名	说明
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>
year	年
mon	月
day	日
hour	时
min	分钟
sec	秒

### 5.1.13. 对 DM8127 的单板：获取降噪模式和降噪强度： get\_denoise

请求命令参数

```
{
  "cmd": "get_denoise",
  "id": "132156"
}
```

结果回复

```
{
  "cmd": "get_denoise",
  "id": "132156",
  "state_code": 200,
  "error_msg": "success",
  "body": {
    "mode": 1,
    "strength": 10
  }
}
```

字段名	说明
mode	降噪模式
strength	降噪强度
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>
id	序列字符串 (唯一), 字符串长度 < 30, 等于请求时的 id

### 5.1.14. 对 DM8127 的单板：设置降噪模式和降噪强度： set\_denoise

请求命令参数

模式设置:

```
{
  "cmd": "set_denoise",
  "id": "132156",
  "body": {
    "mode": 1
  }
}
```

强度设置:

```
{
  "cmd": "set_denoise",
  "id": "132156",
  "body": {
    "strength": 0
  }
}
```

结果返回

```
{
  "cmd": "set_denoise",
  "id": "132156",
  "state_code": 200,
  "error_msg": "success"
}
```

服务器内部错误

```
{
  "cmd": "set_denoise",
  "id": "132156",
  "state_code": 500,
  "error_msg": "service inner error"
}
```

字段名	说明
mode	降噪模式
strength	降噪强度
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 400: 客户端请求出错</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>
id	序列字符串 (唯一), 字符串长度 < 30, 等于请求时的 id

### 5.1.15. 开始自动聚焦: 不支持

请求命令参数

```
{
  "cmd": "startfocusandzoom",
  "id": "1548",
  "body": {
```

```
"value" : 1
}
}
```

结果返回

```
{
"cmd" : "startfocusandzoom",
"id" : "1548",
"state_code" : 200
}
```

字段名	说明
value	自动变倍/调焦 <ul style="list-style-type: none"><li>● 0 停止</li><li>● 1 focus 焦增加</li><li>● 2 focus 焦减小</li><li>● 3 zoom 变倍增加</li><li>● 4 zoom 变倍减小</li></ul>

5.1.16. ~~停止自动聚焦：不支持~~

请求命令参数

```
{
"cmd" : "stopfocusandzoom",
"id" : "1548"
}
```

或者

```
{
"cmd" : "startfocusandzoom",
"id" : "1548",
"body":{
"value" : 0
}
}
```

结果返回

```
{
"cmd" : "stopfocusandzoom",
"id" : "1548",
"state_code" : 200
}
```

5.1.17. 获取设备版本信息: get\_product\_info

请求命令格式

```
{
"cmd": "get_product_info",
"id": "123456"
```

```
}

```

**结果返回值**

如果一切正常则返回如下数据:

```
{
  "body": {
    "boardver": 12363,
    "majorver": 2,
    "productver": "RX"
  },
  "cmd": "get_product_info",
  "error_msg": "Sucess",
  "id": "123456",
  "state_code": 200
}
```

字段名	说明
boardver	Board_version, 整数
majorver	软件主版本, 整数
productver	Product_ver, 字符串, 如 "RX" "RM"

**5.1.18. 获取设备 4G 参数信息: get\_4g\_param****请求命令格式**

```
{
  "cmd": "get_4g_param",
  "id": "123456"
}
```

**结果返回值**

如果一切正常则返回如下数据:

```
{
  "body": {
    "ip": "",
    "apn": "192.168.1.11",
    "apn_enable": false,
    "apn_mode": 0,
    "net_cgatt": false,
    "net_cop": "",
    "net_creg": false,
    "rssi": 0,
    "sim_id": "",
    "state_4g": true,
    "state_sim": false
  },
  "cmd": "get_4g_param",
  "error_msg": "Sucess",
  "id": "12345",
  "state_code": 200
}
```

字段名	说明
state_4g	4G 模块状态 4g_module, 表示是否检测到 4G 模块插入, bool
state_sim	SIM 卡槽状态 sim_slot, 表示是否插卡, bool
sim_id	SIM 卡信息 sim_id, 表示有卡时的填充字段, string
net_cop	当前移动运营商 net_cop, 表示当前移动运营商, string
rsi	移动网络信号强度 rssi, 表示当前信号强度 dBm 值, int
net_creg	移动网络注册信息 net_creg, 表示是否注册到网络中, bool
net_cgatt	移动网络附着和分离 net_cgatt, 表示当前网络是否挂在 GPRS/4G 上还是剥离的只剩下 2G, bool
IP	IP, 表示当前分配到的 IP 地址, string
apn_enable	APN_mode, 表示是否使用 APN 接入点模式, bool
apn_mode	APN_mode, 表示是否使用 APN 接入点模式, bool
apn	APN, 表示 APN 地址, 使用 APN 模式时的填充字段, string

### 5.1.19. 设置设备 4G 参数信息: set\_4g\_param

请求命令格式

```
{
  "cmd": "set_4g_param",
  "id": "12345",
  "body": {
    "sub_cmd": "set_apn",
    "apn_param": {
      "apnaddr": "192.168.1.12",
      "username": "apn_user",
      "passwd": "apn_pass",
      "authentication": 0
    }
  }
}
```

或

```
{
  "cmd": "set_4g_param",
  "id": "12345",
  "body": {
    "sub_cmd": "set_4g_reset"
  }
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "set_4g_param",
  "error_msg": "Sucess",
  "id": "12345",
  "state_code": 200
}
```

字段名	说明
sub_cmd	配置 4g 参数子命令： "set_apn"表示设置 apn 参数 "set_4g_reset"表示 4g 重连
apn_param	apn 参数 "apnaddr"表示 apn 服务器地址 "username"表示 apn 认证用户名 "passwd"表示 apn 认证密码 "authentication"表示 apn 模式，取值： 0: 不启用 1: PAP 2: CHAP 3: PAP_OR_CHAP

### 5.1.20. 设置设备名称: set\_dev\_name

请求命令格式

```
{
  "cmd": "set_dev_name",
  "id": "12345",
  "body": {
    "title": "lvs"
  }
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "set_dev_name",
  "error_msg": "Sucess",
  "id": "12345",
  "state": 200,
  "state_code": 200
}
```

字段名	说明
title	设置设备名称。大小限制：最大 60 个字节。

### 5.1.21. 设置 ftp 服务启用状态: set\_vsftpd\_enable

请求命令格式

```
{
  "cmd": "set_vsftpd_enable",
  "id": "12345",
  "body": {
    "enable": 1
  }
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "set_vsftpd_enable",
  "error_msg": "Sucess",
  "id": "12345",
  "state_code": 200
}
```

字段名	说明
enable	1:开启; 0:关闭

### 5.1.22. 获取 ftp 服务启用状态: get\_vsftpd\_enable

请求命令格式

```
{
  "cmd": "get_vsftpd_enable",
  "id": "12345"
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "get_vsftpd_enable",
  "error_msg": "Sucess",
  "id": "12345",
  "state_code": 200,
  "body": {
    "enable": 1
  }
}
```

字段名	说明
enable	1:开启; 0:关闭

### 5.1.23. 修改账号密码(linux 账号和 ftpserver 对应的账号)

请求命令格式

```
{
  "cmd": "set_user_password",
  "id": "132156",
  "body": {
    "account_number": "admin",
    "old_password": "MTIzNDU2",
    "new_password": "YWRtaW4="
  }
}
```

结果返回值

如果一切正常则返回如下数据:

```
{
  "cmd": "set_user_password",
```

```
{
  "error_msg": "Sucess",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
account_number	账号: admin 目前只支持 admin 账号修改密码
old_password	旧密码数字和可见字符组成 base64 编码
new_password	新的密码数字和可见字符组成 base64 编码

## 5.2. 车牌识别

### 5.2.1. 配置推送数据方式: ivsresult

```
{
  "cmd": "ivsresult",
  "id": "123",
  "enable": true,
  "format": "json",
  "image": true,
  "image_type": 0
}
```

请求命令参数

字段名	说明
enable	是否允许推送识别结果，默认值: false <ul style="list-style-type: none"> <li>• true 表示允许推送识别结果</li> <li>• false 表示不推送识别结果</li> </ul>
format	推送识别结果数据格式，默认值: json <ul style="list-style-type: none"> <li>• bin 表示识别结果数据格式为二进制数据格式</li> <li>• json 表示识别结果数据格式为 json 数据格式</li> </ul>
image	识别结果是否包含图片，默认值: true <ul style="list-style-type: none"> <li>• true 表示识别结果包含图片</li> <li>• false 表示识别结果不包含图片</li> </ul>
image_type	识别的图片类型，默认值: 0 <ul style="list-style-type: none"> <li>• 0 表示返回识别结果全图</li> <li>• 1 表示只返回车牌区域小图</li> <li>• 2 表示返回两种图片</li> <li>• 3 表示只返回车牌区域二值化小图</li> <li>• 4 表示返回识别结果全图+车牌区域小图+二值化车牌区域小图</li> </ul>

当配置完成后（配置为允许推送识别结果的情况 enable: true），并且在服务器端有 ivsresult 识别结果产生后，tcp server 会主动推送 ivsresult 识别结果给客户端，推送的识别结果格式如下

结果返回值

```
{
  "cmd": "ivsresult",
```



```
"id" : "123",
"state_code" : 200
}
```

配置完成后，接收到识别结果为配置值：  
正常情况下，一个完整的识别结果（返回 3 种图片的情况）推送消息格式如下：

‘V’	‘Z’	0	0	长度（数据包的大小）
识别结果字符串+字符串结束符‘\0’				
全图的 jpg 格式数据				
小图片（车牌区域图片）的 jpg 格式数据				
二值化小图片（车牌区域图片）的 1bit bmp 格式数据 (无 bmp 文件头)				

例如下面就是一个正常的 JSON 识别推送结果：

```
{
  "PlateResult": {
    "bright": 0,
    "carBright": 0,
    "carColor": 0,
    "car_brand": {
      "brand": 37,
      "type": 0,
      "year": 4017
    },
    "car_location": {
      "RECT": {
        "bottom": 0,
        "left": 0,
        "right": 0,
        "top": 0
      }
    },
    "colorType": 1,
    "colorValue": 0,
    "confidence": 91,
    "direction": 0,
    "enable_encrypt": 0,
    "fake_plate": 0,
    "featureCode": true,
    "license": "青 PTW3Z3",
    "license_ext_type": 0,
    "location": {
      "RECT": {
        "bottom": 1509,
        "left": 1374,
        "right": 1578,
        "top": 1444
      }
    }
  },
}
```

```
"plate_distance": 0,
"plate_true_width": 205,
"plates": [
  {
    "binImgSize": 0,
    "clipImgSize": 3483,
    "color": 1,
    "is_danger": 0,
    "license": "青 PTW3Z3",
    "plate_width": 205,
    "pos": {
      "bottom": 1509,
      "left": 1374,
      "right": 1578,
      "top": 1444
    },
    "type": 1
  }
],
"timeStamp": {
  "Timeval": {
    "sec": 1708675913,
    "usec": 519076
  }
},
"timeUsed": 0,
"triggerType": 8,
"trigger_time_end": 0,
"trigger_time_start": 0,
"type": 1,
"white_list_info": 0,
"begin_time": 1725871942,
"end_time": 1725871952,
"start_time": 1686299994687,
"unique_id": "dac8d1d3-248ce33e_1686299994687"
},
"active_id": 0,
"clipImgSize": 3483,
"cmd": "ivs_result",
"fullImgSize": 232950,
"id": 230,
"imageformat": "jpg",
"rule_id": 0,
"timeString": "2024-02-23 16:11:53"
}
```

其中各字段的含义如下表所示：

字段名	说明
license	车牌号码（汉字为 GB2312 编码）
enable_encrypt	车牌是否加密（0 不加密，1 加密）
colorType	车牌颜色序号，详见车牌颜色定义 LC_X

colorValue	车牌颜色
type	车牌类型, 详见车牌类型定义 LT_X
confidence	车牌可信度
bright	亮度评价
direction	运动方向, 详见运动方向定义 DIRECTION_X
license_ext_type	新式小型车牌扩展, 仅在车牌为小型车牌的时候有效, 详情: NewCarTypeExtEnum
location: rect	车牌位置
timeUsed	识别所用时间
carBright	车的亮度
carColor	车的颜色, 详见车辆颜色定义 LCOLOUR_X
timeStamp	识别时间点
triggerType	<p>触发结果的类型如下:</p> <ul style="list-style-type: none"> <li>● 1 自动识别触发</li> <li>● 2 外部输入触发 (IO)</li> <li>● 4 软件触发(SDK)</li> <li>● 8 虚拟线圈触发</li> <li>● 64 车滞留事件</li> <li>● 65 车滞留恢复事件</li> <li>● 66 车折返事件</li> <li>● 68 跟车事件</li> <li>● 69 车辆拥堵</li> <li>● 70 车辆拥堵恢复</li> <li>● 71 人滞留事件</li> <li>● 72 人滞留恢复</li> <li>● 73 人员拥堵</li> <li>● 74 人员拥堵恢复</li> <li>● 75 非机动车滞留</li> <li>● 76 非机动车滞留恢复</li> <li>● 77 道闸正常</li> <li>● 78 道闸异常</li> <li>● 79 道闸回落</li> <li>● 80 道闸抬升</li> <li>● 85 车头车尾号牌不一致</li> <li>● 86 车牌识别</li> <li>● 87 非机动车拥堵</li> <li>● 88 非机动车拥堵解除</li> </ul> <p>注意: 类型 69~88 目前仅 x2 产品涉及</p>
cmd	当前指令名称
id	识别记录的编号
imageformat	图片格式
timeString	触发时间字符串, 格式如: 2015-01-02 03:04:05
fullImgSize	整幅大图的尺寸 (字节数)
clipImgSize	车牌区域图片的尺寸 (字节数)

active_id	车牌加密方式
plate_true_width	车牌真实宽度
plate_distance	车牌距离
fake_plate	是否是伪车牌
car_location	车辆位置
Car_Brand	车辆品牌信息
plates	数组，车牌信息
featureCode	特征码(16 位字符串)
white_list_info	白名单信息：0-临时车 1-月租车 2-月租车到期 9-特殊车辆
trigger_time_start	触发开始时间
trigger_time_end	触发结束时间
rule_id	算法识别规则 id
begin_time	事件开始时间， 秒
end_time	事件结束时间， 秒
start_time	事件触发时间， 毫秒
unique_id	事件唯一 id， 格式为 sn_start_time， 如： "12345678-87654321_159728586500"

Car\_Brand 中个字段信息

brand	车辆品牌信息
type	车辆类型
year	年份

plates 的定义

license	车牌号码 (utf8 编码)。若开启加密功能，此为加密数据后的车牌数据的 base64 格式。	
color	车牌颜色序号，详见车牌颜色定义 LC_X	
type	车牌类型，详见车牌类型定义 LT_X	
clipImgSize	车牌区域图片的尺寸（字节数）。如果此值为 0，表示没有相应的小图。	
binImgSize	车牌区域二值化图片的尺寸（字节数）。如果此值为 0，表示没有相应的二值化小图。	
is_danger	是否是危险车辆，0：否，1：是，-1：未开启改功能（仅高速版才有该功能）	
plate_width	车牌宽度	
pos	车辆位置	
	bottom	车牌在图片中位置
	left	车牌在图片中位置
	right	车牌在图片中位置
	top	车牌在图片中位置

另，LC\_X，LT\_X，DIRECTION\_X，LCOLOUR\_X，TH\_TRIGGER\_TYPE\_BIT 详细定义请参考头文件 VzClientSDK\_LPDefine.h。

如果用户设置的是二进制识别结果，那么将会收到如下的二进制数据：

'V'	'Z'	2	0	长度（整个包的大小）
'I'	'R'	1	0	长度（识别结果结构体的大小）
识别结果结构体(TH_PlateResult)				
'I'	'R'	2（大图片）	0	长度（大图像数据的大小）
全图的照片数据(完整的 jpg 格式)				
'I'	'R'	3（小图片）	0	长度（小图像数据的大小）
小图片（车牌区域图片）数据(完整的 jpg 格式)				

注：二进制识别结果协议不再更新，如需对接新功能，请使用 JSON 识别结果

推送的识别结果结构体定义为 TH\_PlateResult，这一个结构体的定义请参考头文件 VzClientSDK\_LPDefine.h。

### 1.1.1. 配置快速推送数据方式: quick\_ivsresult

请求命令参数

{
"cmd" : "quick_ivsresult",
"id" : "123",
"enable" : true
}

字段名	说明
enable	是否允许快速推送识别结果，默认值: false <ul style="list-style-type: none"> <li>● true 表示允许快速推送识别结果</li> <li>● false 表示不使用快速推送识别结果</li> </ul>

结果返回值

{
"cmd" : "quick_ivsresult",
"error_msg" : "Success",
"id" : "123",
"state_code" : 200
}

当配置完成后（配置为允许快速推送识别结果的情况 enable: true），并且在服务器端有 quick\_ivsresult 识别结果产生后，tcp server 会主动推送 quick\_ivsresult 识别结果给客户端，推送的识别结果格式如下

例如下面就是一个正常的快速识别推送 JSON 结果：

{
"PlateResult": {
"bright": 0,
"carBright": 0,
"carColor": 0,
"colorType": 1,
"confidence": 99,
"direction": 0,
"license": "青 PTW3Z3",
}
}

```
"enable_encrypt": 0,
"location": {
  "RECT": {
    "bottom": 392,
    "left": 690,
    "right": 834,
    "top": 350
  }
},
"timeStamp": {
  "Timeval": {
    "sec": 1458882234,
    "usec": 921325
  }
},
"timeUsed": 0,
"triggerType": 1,
"type": 1,
"plate_true_width": 1,
"plate_distance": 50,
"fake_plate": 1,
"car_location": {
  "RECT": {
    "bottom": 392,
    "left": 690,
    "right": 834,
    "top": 350
  }
},
"car_brand": {
  "brand": 1,
  "type": 2,
  "year": 2017
},
"featureCode": "asdghb4AGD",
"plates": [
  {
    "color": 1,
    "is_danger": 0,
    "license": "青 PTW3Z3",
    "pos": {
      "bottom": 704,
      "left": 487,
      "right": 801,
      "top": 598
    },
    "type": 1
  }
],
"begin_time": 1725871942,
"end_time": 1725871952,
"start_time": 1686299994687,
"unique_id": "dac8d1d3-248ce33e_1686299994687"
```

```

    },
    "active_id": 0,
    "cmd": "ivs_result",
    "id": 0,
    "timeString": "2016-03-25 13:03:54"
  }

```

其中各字段的含义请参考 ivsresult 推送信息字段含义表  
(注：快速推送的数据结果里无图像相关信息)

### 5.2.2. 获取最近一次识别结果: getivsresult

请求命令参数

```

{
  "cmd": "getivsresult",
  "image": true,
  "format": "json"
}

```

字段名	说明
image	是否接收识别结果图片，默认值：false <ul style="list-style-type: none"> <li>● true 需要识别结果图片</li> <li>● false 不需要识别结果图片</li> </ul>
format	推送识别结果数据格式，默认值：json <ul style="list-style-type: none"> <li>● bin 表示识别结果数据格式为二进制数据格式</li> <li>● json 表示识别结果数据格式为 json 数据格式。</li> </ul>

结果返回值

这个命令的结果返回值，参见配置推送数据方式里面的识别结果返回值。

### 5.2.3. 手动触发车牌识别: trigger

请求命令参数

```

{
  "cmd": "trigger"
}

```

结果返回值

这个命令的结果返回值，参见配置推送数据方式里面的识别结果返回值。

### 5.2.4. 获取记录最大 id: get\_max\_rec\_id

主要用于在脱机之后，断线之后使用。

请求命令参数

```

{
  "cmd": "get_max_rec_id",
  "id": "123"
}

```

**结果返回**

```
{
  "cmd": "get_max_rec_id",
  "id": "123",
  "state_code": 200,
  "max_id": 1024
}
```

字段名	说明
max_id	识别结果记录中最大的 id 值

**5.2.5. 获取历史记录: get\_record**

根据 id 获取识别记录

**请求命令参数**

```
{
  "cmd": "get_record",
  "id": 2,
  "format": "json",
  "image": true
}
```

字段名	说明
id	识别结果记录的 id 值
format	推送识别结果数据格式，默认值：json bin 表示识别结果数据格式为二进制数据格式，当前不支持 bin 形式。 json 表示识别结果数据格式为 json 数据格式。
image	识别结果是否包含图片，默认值：true true 表示识别结果包含图片 false 表示识别结果不包含图片

**结果返回**

这个命令的结果返回值，参见配置推送数据方式里面的识别结果返回值。

**5.2.6. 获取记录图片: get\_image、get\_offline\_image**

按记录 ID 获取识别结果对应的图片

**请求命令参数**

```
{
  "cmd": "get_image",
  "id": 2
}
```

字段名	说明
id	识别结果记录的 id，可在识别记录结构体中找到。

**结果返回**

```
{
  "cmd": "get_image",
```



```
{
  "id": 2,
  "size": 57344,
  "state_code": 200
}
```

字段名	说明
id	识别结果记录的 id, 可在识别记录结构体中找到。前面的获取脱机记录中的 ID 也一样
size	图片的大小

注释：这个 JSON 后面跟了一个 57344 大小的二进制图片，其中 JSON 字符串以 ' \0' 结束。

### 5.2.7. 抓取当前图片 get\_snapshot

请求命令参数

```
{
  "cmd": "get_snapshot",
  "id": "123456"
}
```

结果返回

```
{
  "cmd": "get_snapshot",
  "state_code": 200,
  "id": "123456",
  "imgformat": "jpg",
  "imgdata": "14ewrfswtergfhbfr=="
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码
imgformat	图片格式 (jpg)
imgdata	图片数据：经过 base64 转码后的数据

### 5.2.8. 获取视频播放的 uri: get\_rtsp\_uri

请求命令参数

```
{
  "cmd": "get_rtsp_uri",
  "id": "132156"
}
```

结果返回

```
{
  "cmd": "get_rtsp_uri",
  "id": "132156",
  "state_code": 200,
  "uri": "rtsp://192.168.7.7:8557/h264"
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码
uri	当前播放视频的 URI

### 5.2.9. 获取虚拟线圈参数：get\_virloop\_para

请求命令参数

```
{
  "cmd": "get_virloop_para",
  "id": "132156"
}
```

结果返回

```
{
  "body": {
    "virtualloop": {
      "max_plate_width": 400,
      "min_plate_width": 45,
      "dir": 0,
      "trigger_gap": 10,
      "virtualloop_num": 1,
      "loop": [
        {
          "id": 0,
          "enable": true,
          "point_num": 4,
          "point": [
            {
              "x": 2400,
              "y": 12000
            },
            {
              "x": 13984,
              "y": 12000
            },
            {
              "x": 15984,
              "y": 14000
            },
            {
              "x": 400,
              "y": 14000
            }
          ]
        }
      ]
    }
  },
  "cmd": "get_virloop_para",
  "error_msg": "success",
}
```

```

    "id": "132156",
    "state_code": 200
  }

```

Body 中个字段含义

字段名	说明
virtualloop	虚拟线圈
max_plate_width	最大车牌尺寸
min_plate_width	最小车牌尺寸
dir	运动方向
trigger_gap	相同车牌的触发时间间隔
virtualloop_num	虚拟线圈数量

Loop 中字段含义：

字段名	说明
id	线圈 ID
enable	是否可用
point_num	顶点个数
Poin	顶点坐标 <ul style="list-style-type: none"> <li>● X 横向坐标，从左至右，从 0 递增</li> <li>● Y 纵向坐标，从上至下，从 0 递增</li> </ul>

### 5.2.10. 设置虚拟线圈参数：set\_virloop\_para

请求命令参数

```

{
  "body": {
    "virtualloop": {
      "max_plate_width": 400,
      "min_plate_width": 45,
      "dir": 0,
      "trigger_gap": 10,
      "virtualloop_num": 1,
      "loop": [
        {
          "id": 0,
          "enable": true,
          "point_num": 4,
          "point": [
            {
              "x": 2400,
              "y": 12000
            },
            {
              "x": 13984,
              "y": 12000
            },
            {
              "x": 15984,
              "y": 14000
            }
          ]
        }
      ]
    }
  }
}

```

```

        {
            "x": 400,
            "y": 14000
        }
    ]
}
},
"cmd": "set_virloop_para",
"id": "132156"
}

```

#### 结果返回

```

{
    "cmd": "set_virloop_para",
    "id": "132156",
    "error_msg": "success",
    "state_code": 200
}

```

参数含义请参考 2.2.9 上方;

### 5.2.11. 获取虚拟线圈属性: get\_virloop\_prop

#### 请求命令参数

```

{
    "cmd": "get_virloop_prop",
    "id": "132156"
}

```

#### 结果返回

```

{
    "body": {
        "virtualloop": {
            "plate_width": {
                "max": 600,
                "min": 45,
                "default_max": 400,
                "default_min": 25
            },
            "trigger_gap": {
                "max": 255,
                "min": 0,
                "default": 10
            },
            "dir": {
                "default": 0,
                "types": [
                    {
                        "content": "双向",
                        "type": 0
                    }
                ]
            }
        }
    }
}

```

```
    },
    {
      "content": "从上至下",
      "type": 1
    },
    {
      "content": "从下至上",
      "type": 2
    }
  ]
}
}
}
"cmd": "get_virloop_prop",
"error_msg": "success",
"id": "132156",
"state_code": 200
}
```

字段名	说明
plate_width	车牌宽度 <ul style="list-style-type: none"><li>max 最大宽度</li><li>min 最小宽度</li><li>default_max 默认最大宽度</li><li>default_min 默认最小宽度</li></ul>
trigger_gap	相同车牌的触发时间间隔 <ul style="list-style-type: none"><li>max 最长间隔时间</li><li>min 最短间隔时间</li><li>default 默认间隔时间</li></ul>
dir	运动方向 <ul style="list-style-type: none"><li>content" 双向" , type,0</li><li>content" 从上至下" ,type,1</li><li>content" 从下至上" ,type2</li><li>default 默认值</li></ul>

5.2.12. 获取识别区域参数：get\_reco\_para

请求命令参数

```
{
  "cmd": "get_reco_para",
  "id": "132156"
}
```

结果返回

```
{
  "body": {
    "recognition_area": {
      "polygon_num": 1,
      "polygon": [
        {
```

```
        "id": 1,
        "enable": true,
        "point_num": 4,
        "point": [
            {
                "x": 4096,
                "y": 4096
            },
            {
                "x": 12288,
                "y": 4096
            },
            {
                "x": 13926,
                "y": 13926
            },
            {
                "x": 2457,
                "y": 13926
            }
        ]
    },
    "cmd": "get_reco_para",
    "id": "132156",
    "error_msg": "success",
    "state_code": 200
}
```

recognition\_area 识别区域中字段含义

字段名	说明
polygon_num	规则数量
enable	是否可用
point_num	顶点个数
Poin	顶点坐标 <ul style="list-style-type: none"><li>● X 横向坐标, 从左至右, 从 0 递增</li><li>● Y 纵向坐标, 从上至下, 从 0 递增</li></ul>

5.2.13. 设置识别区域参数：set\_reco\_para

请求命令参数

```
{
    "body": {
        "recognition_area": {
            "polygon_num": 1,
            "polygon": [
                {
                    "id": 1,
                    "enable": true,
                    "point_num": 4,
```

```

        "point": [
            {
                "x": 4096,
                "y": 4096
            },
            {
                "x": 12288,
                "y": 4096
            },
            {
                "x": 13926,
                "y": 13926
            },
            {
                "x": 2457,
                "y": 13926
            }
        ]
    },
    "cmd": "set_reco_para",
    "id": "132156"
}

```

#### 结果返回

```

{
  "cmd": "set_reco_para",
  "id": "132156",
  "error_msg": "success",
  "state_code": 200
}

```

参数字段含义参见 2.2.13 上方

### 5.2.14. 获取识别区域属性: get\_reco\_prop

#### 请求命令参数

```

{
  "cmd": "get_reco_prop",
  "id": "132156"
}

```

#### 结果返回

```

{
  "body": {
    "polygon_num": { "max": 1, "min": 0 },
    "polygon": [ {
      "id": 1,
      "enable": true,
      "point_num": 4,
      "point": [
        { "x": 4096, "y": 4096 },
        { "x": 12288, "y": 4096 },

```

```
{ "x": 13926, "y": 13926 },
{ "x": 2457, "y": 13926 }
}
}
"cmd": "get_reco_prop",
"id": "132156",
"error_msg": "success",
"state_code": 200
}
```

字段名	说明
polygon_num	规则数量 <ul style="list-style-type: none"> <li>● Max 最大数量</li> <li>● Min 最小数量</li> </ul>

### Polygon 中各字段含义

id	规则 ID
enable	是否可用
point_num	顶点个数
Poin	顶点坐标 <ul style="list-style-type: none"> <li>● X 横向坐标, 从左至右, 从 0 递增</li> <li>● Y 纵向坐标, 从上至下, 从 0 递增</li> </ul>

## 5.2.15. 获取算法识别参数: get\_alg\_result\_para

### 请求命令参数

```
{
  "cmd": "get_alg_result_para",
  "id": "132156"
}
```

### 回复

```
{
  "cmd": " get_alg_result_para",
  "id": "132156",
  "state_code": 200,
  "body": {
    "snap_resolution": 0,
    "snap_image_quality": 100,
    "out_result_type": 1,
    "recognition_type": 1,
    "province": 1,
    "run_mode": 0,
    "alg_version": "2016LPALG",
    "time_zone": 0,
    "reco_dis": 0
  }
}
```

### 字段含义

字段名	说明
snap_resolution	抓拍图片分辨率
snap_image_quality	抓拍图片质量, 单位百分比;
out_result_type	输出类型;



recognition_type	识别类型
province	预设省份
run_mode	运行模式
alg_version	算法版本
time_zone	时区
reco_dis	识别距离 0: 1-6 米 1: 大于 6 米

### 5.2.16. 设置算法识别参数: set\_alg\_result\_para

请求命令参数

```
{
  "cmd": "set_alg_result_para",
  "id": "132156",
  "body": {
    "snap_resolution": 0,
    "snap_image_quality": 100,
    "out_result_type": 1,
    "recognition_type": 1,
    "province": 1,
    "run_mode": 0,
    "alg_version": "2016LPRALG",
    "time_zone": 0,
    "reco_dis": 0
  }
}
```

回复

```
{
  "cmd": "set_alg_result_para",
  "id": "132156",
  "state_code": 200
}
```

字段含义

字段名	说明
snap_resolution	抓拍图片分辨率
snap_image_quality	抓拍图片质量, 单位百分比;
out_result_type	输出类型;
recognition_type	识别类型
province	预设省份
run_mode	运行模式
alg_version	算法版本
time_zone	时区
reco_dis	识别距离 0: 1-6 米 1: 大于 6 米

### 5.2.17. 获取算法识别参数属性: get\_alg\_result\_prop

请求命令参数

```
{
  "cmd": "get_alg_result_prop",
}
```

```
"id": "132156"
}
```

### 回复

```
{
  "body": {
    "out_result_type": {
      "default": 4,
      "types": [
        {
          "content": "6Jma5ouf57q/5Zyl",
          "type": 1
        },
        {
          "content": "56iz5a6a6K+G5Yir",
          "type": 2
        },
        {
          "content": "5aSW6YOo6Kem5Y+R",
          "type": 4
        },
        {
          "content": "5a6e5pe257uT5p6c",
          "type": 8
        },
        {
          "content": "5byA5YWz6YePL+eUteW5s+i+k+WFpTE=",
          "type": 16
        },
        {
          "content": "5byA5YWz6YePL+eUteW5s+i+k+WFpTI=",
          "type": 32
        }
      ]
    },
    "province": {
      "default": 0,
      "types": [
        {
          "content": "5Lqs",
          "type": 0
        },
        {
          "content": "5rSI",
          "type": 1
        },
        {
          "content": "5YaA",
          "type": 2
        },
        {
          "content": "5pmL",
          "type": 3
        }
      ]
    }
  }
}
```

```
{
  "content": "6JKZ",
  "type": 4
},
{
  "content": "6L69",
  "type": 5
},
{
  "content": "5ZCJ",
  "type": 6
},
{
  "content": "6buR",
  "type": 7
},
{
  "content": "5rKq",
  "type": 8
},
{
  "content": "6luP",
  "type": 9
},
{
  "content": "5rWZ",
  "type": 10
},
{
  "content": "55qW",
  "type": 11
},
{
  "content": "6Ze9",
  "type": 12
},
{
  "content": "6LWj",
  "type": 13
},
{
  "content": "6bKB",
  "type": 14
},
{
  "content": "6LGr",
  "type": 15
},
{
  "content": "6YSC",
  "type": 16
},
{
  "content": "5rmY",
```

```
        "type": 17
    },
    {
        "content": "57Kk",
        "type": 18
    },
    {
        "content": "5qGC",
        "type": 19
    },
    {
        "content": "55C8",
        "type": 20
    },
    {
        "content": "5rid",
        "type": 21
    },
    {
        "content": "5bed",
        "type": 22
    },
    {
        "content": "6LS1",
        "type": 23
    },
    {
        "content": "5LqR",
        "type": 24
    },
    {
        "content": "6JeP",
        "type": 25
    },
    {
        "content": "6ZmV",
        "type": 26
    },
    {
        "content": "55SY",
        "type": 27
    },
    {
        "content": "6Z2S",
        "type": 28
    },
    {
        "content": "5a6B",
        "type": 29
    },
    {
        "content": "5paw",
        "type": 30
    }
}
```

```

    ]
  },
  "recognition_type": {
    "default": 1599546,
    "types": [
      {
        "content": "5pyq55+l",
        "type": 1
      },
      {
        "content": "6JOd54mM",
        "type": 2
      },
      {
        "content": "6buR54mM",
        "type": 4
      },
      {
        "content": "5Y2V5o6S6buE54mM",
        "type": 8
      },
      {
        "content": "5Y+M5o6S6buE54mM",
        "type": 16
      },
      {
        "content": "6K2m6L2m",
        "type": 32
      },
      {
        "content": "5q2m6K2m6L2m",
        "type": 64
      },
      {
        "content": "5Liq5oCn5YyW6L2m54mM",
        "type": 128
      },
      {
        "content": "5Y2V5o6S5Yab6L2m54mM",
        "type": 256
      },
      {
        "content": "5Y+M5o6S5Yab6L2m54mM",
        "type": 512
      },
      {
        "content": "5L2/6aaG6L2m",
        "type": 1024
      },
      {
        "content": "6aaZ5riv6L+b5Ye65Lit5Zu95aSn6ZmG6L2m54mM",
        "type": 2048
      },
    ]
  }

```

```

        "content": "5Yac55So6L2m54mM",
        "type": 4096
    },
    {
        "content": "5pWZ57uD6L2m54mM",
        "type": 8192
    },
    {
        "content": "5r6z6Zeo6L+b5Ye65Lit5Zu95aSn6ZmG6L2m54mM",
        "type": 16384
    },
    {
        "content": "5Y+M5bGC5q2m6K2m6L2m54mM",
        "type": 32768
    },
    {
        "content": "5q2m6K2m5oC76Zif6L2m54mM",
        "type": 65536
    },
    {
        "content": "5Y+M5bGC5q2m6K2m5oC76Zif6L2m54mM",
        "type": 131072
    },
    {
        "content": "5rCR6liq6L2m54mM",
        "type": 262144
    },
    {
        "content": "5paw6lO95rqQ6L2m54mM",
        "type": 524288
    },
    {
        "content": "5aSn5Z6L5paw6lO95rqQ6L2m54mM",
        "type": 1048576
    },
    {
        "content": "5bqU5oCl",
        "type": 2097152
    },
    {
        "content": "6aKG6aaG6L2m54mM",
        "type": 4194304
    },
    {
        "content": "5paw5qCH5YeG5bCP5Z6L54eD5rK56L2m",
        "type": 8388608
    },
    {
        "content": "5paw5qCH5YeG5bCP5Z6L5paw6lO95rqQ6L2m",
        "type": 16777216
    },
    {
        "content": "5py65Zy66L2m54mM",
        "type": 33554432
    }

```

```

        },
        {
            "content": "5rW35aSW6L2m54mM",
            "type": 67108864
        },
        {
            "content": "5YGH6L2m54mM",
            "type": 536870912
        },
        {
            "content": "5peg54mM6L2m",
            "type": 2147483648
        }
    ]
},
"snap_image_quality": {
    "default": 70,
    "max": 100,
    "min": 10
},
"snap_resolution": {
    "default": 10,
    "types": [
        {
            "content": "NjQwKjM2MA==",
            "type": 7
        },
        {
            "content": "NzA0KjU3Ng==",
            "type": 8
        },
        {
            "content": "MTI4MCo3MjA=",
            "type": 9
        },
        {
            "content": "MTkyMCoxMDgw",
            "type": 10
        },
        {
            "content": "MjMwNCoxMjk2",
            "type": 11
        },
        {
            "content": "MjU2MCoxNDQw",
            "type": 12
        },
        {
            "content": "MjY4OCoxNTEy",
            "type": 14
        },
        {
            "content": "Mjg4MCo3NjIw",
            "type": 15
        }
    ]
}

```

```
        }
    ]
}
},
"cmd": "get_alg_result_prop",
"error_msg": "Sucess",
"id": "132156",
"state_code": 200
}
```

5.2.18. 设置 LED 参数: set\_led\_para

请求命令参数

```
{
  "body": {
    "led_level": 2,
    "led_mode": 3,
    "time_ctrl": [
      {
        "id": 0,
        "led_level": 2,
        "time_begin": "00:00:00",
        "time_end": "02:03:34",
        "timectrl_enable": true
      },
      {
        "id": 1,
        "led_level": 6,
        "time_begin": "02:03:34",
        "time_end": "09:02:23",
        "timectrl_enable": true
      }
    ]
  },
  "cmd": "set_led_para",
  "id": "132156"
}
```

回复

```
{
  "cmd": "set_led_para",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
led_mode	LED 模式 <ul style="list-style-type: none"><li>0 智能</li><li>2 常灭</li><li>3 时间段</li></ul>



led_level	LED 等级[0,5]
time_ctrl	时间段控制
id	时间段 ID
time_begin	时间段起始时间
time_end	时间段结束时间
timectrl_enable	时间段使能 (0 去使能 1 使能)

5.2.19. 获取 LED 参数: get\_led\_para

请求命令参数

```
{
  "cmd" : "get_led_para",
  "id" : "132156"
}
```

回复

```
{
  "body": {
    "led_level": 2,
    "led_mode": 3,
    "time_ctrl": [
      {
        "id": 0,
        "led_level": 2,
        "time_begin": "00:00:00",
        "time_end": "01:02:23",
        "timectrl_enable": true
      },
      {
        "id": 1,
        "led_level": 6,
        "time_begin": "01:02:24",
        "time_end": "09:02:23",
        "timectrl_enable": true
      }
    ]
  },
  "cmd": "get_led_para",
  "id": "132156",
  "state_code": 200
}
```

5.2.20. 获取 LED 属性: get\_led\_prop

请求命令参数

```
{
  "cmd" : "get_led_prop",
  "id" : "132156"
}
```

## 回复

```

{
  "cmd": " get_led_prop ",
  "id": "132156",
  "state_code": 200,
  "body": {
    "led_mode": {
      "default": 0,
      "types": [
        {
          "content": "智能",
          "type": 0
        },
        {
          "content": "常亮",
          "type": 1
        },
        {
          "content": "常灭",
          "type": 2
        },
        {
          "content": "时间段",
          "type": 3
        }
      ]
    },
    "led_level": {
      "default": 4,
      "max": 5,
      "min": 0
    }
  },
}

```

## 5.2.21 获取道闸状态 barr\_gate\_status\_notify

## 请求命令参数示例

```

{
  "cmd": "barr_gate_status_notify",
  "enable": true,
  "id": "123"
}

```

## 结果返回

```

{
  "cmd": "barr_gate_status_notify",
  "error_msg": "Sucess",
  "id": "123",
  "state_code": 200
}

```

获取状态设置为 **true** 成功后，会收到当前的道闸状态，此后道闸状态发生改变时，设备也会推送新的道闸状态

道闸状态实例

```
{
  "body": {
    "connect_status":1,
    "enable":1,
    "gate_ctrl_id" : 1,
    "gate_status" : 1
  },
  "cmd" : "barr_gate_status",
  "type" : "BUS_NOTIFY_BARR_GATE_STATUS"
}
```

字段名	类型	说明
gate_ctrl_id	Int	道闸控制器型号编号
gate_status	Int	道闸当前状态 0: 关到位 1: 开到位 2: 中间状态
connect_status	Int	道闸连接状态 0: 未连接 1: 已连接
enable	Int	道闸功能开关状态 0: 关闭 1: 开启

5.3. 设备硬件参数配置

5.3.1. 设备端主动推送 Gpio 输入状态消息: gpio\_trigger

```
{
  "cmd" : "gpio_trigger",
  "gpio" : 0,
  "value" : 1
}
```

字段名	类型	说明
gpio	Int	对应的输出 IO 编号，必需值，正常情况下只有 0，1 两个输出编号，其他 2，3 等对具体产品规格来确定

value	Int	输入状态参考, Gpio 状态码表 1
-------	-----	---------------------

### 5.3.2.控制 IO 输出: ioctl, ioctl\_resp

请求命令参数示例

```
{
  "cmd": "ioctl",
  "id": "132156",
  "delay": 500,
  "io": 0,
  "value": 2
}
```

```
{
  "cmd": "ioctl_resp",
  "id": "132156",
  "delay": 500,
  "io": 0,
  "value": 2
}
```

字段名	说明
io	对应的输出 IO 编号, 必需值, 正常情况下只有 0, 1 两个输出编号, 其他 2, 3 等对具体产品规格来确定
value	输出 IO 的状态值 <ul style="list-style-type: none"> <li>● 0 断</li> <li>● 1 通</li> <li>● 2 先通后断 (一般做开闸使用)</li> </ul>
delay	先通后断的延迟时间

结果返回

当发送 CMD 为: ioctl\_resp 时有如下返回:

```
{
  "cmd": " ioctl_resp ",
  "id": "132156",
  "state_code": 200
}
```

### 5.3.3.获取 IO 输入状态: get\_gpio\_value

请求命令参数示例

```
{
  "cmd": "get_gpio_value",
  "id": "123",
  "gpio": 0
}
```

**结果返回**

```
{
  "cmd": "get_gpio_value",
  "error_msg": "Sucess",
  "id": "123",
  "state_code": 200,
  "gpio": 0,
  "value": 1
}
```

字段名	说明
gpio	对应的输入 IO 编号，必需值，正常情况下只有 0, 1 两个输入编号 其他 2, 3 等对具体产品规格来确定
value	输入 IO 的状态值 <ul style="list-style-type: none"> <li>● 0 低电平</li> <li>● 1 高电平</li> </ul>

**5.3.4. 获取 IO 输出状态: get\_gpio\_out\_value****请求命令参数示例**

```
{
  "cmd": "get_gpio_out_value",
  "id": "123",
  "gpio": 0
}
```

**结果返回**

```
{
  "cmd": "get_gpio_out_value",
  "id": "123",
  "state_code": 200,
  "gpio": 0,
  "value": 1
}
```

字段名	说明
gpio	对应的输入 IO 编号，必需值，正常情况下只有 0, 1 两个输入编号
value	输入 IO 的状态值 <ul style="list-style-type: none"> <li>● 0 低电平</li> <li>● 1 高电平</li> </ul>

**5.3.5. 自动聚焦: 不支持****请求命令参数**

```
{
  "cmd": "auto_focus"
}
{
  "cmd": "auto_focus_rsp",
  "id": "123"
}
```

```
}

```

**结果返回**

可观察到设备有自动聚焦的动作发生

当 cmd 为: auto\_focus\_rsp 时, 返回如下:

```
{
  "cmd": " auto_focus_rsp ",
  "id" : "123",
  "state_code" : 200
}
```

**5.3.6. 获取存储设备信息: get\_diskinfo****请求命令参数**

```
{
  "cmd" : "get_diskinfo",
  "id": "123"
}
```

**四代结果返回**

```
{
  "body": [
    {
      "devname": "/dev/mmcblk0",
      "devparts": [
        {
          "formatpercent": 0,
          "partname": "/dev/mmcblk0p1",
          "partspace": {
            "left": 7602,
            "total": 7932,
            "used": 330
          },
          "partstate": 5
        }
      ],
      "devtotal": 7948,
      "devtype": 0
    }
  ],
  "cmd": "get_diskinfo",
  "error_msg": "success",
  "id": "123",
  "state_code": 200
}
```

**三代结果返回**

```
{
  "body" : [
    {
      "devname" : "/dev/mmcblk0",
      "devstate" : 5,
      "devtotal" : 7945,
```

```

        "devtype" : 0
    }
},
"cmd" : "get_diskinfo",
"error_msg" : "Sucess",
"id" : "123",
"state" : 200,
"state_code" : 200
}

```

字段名	说明
devname	当前磁盘名称
formatpercent	格式化百分比, 此值不用关注
devparts	三代无此字段
left	剩余磁盘空间
total	磁盘总空间
used	已使用磁盘空间
partstate devstate	分区状态 0. 错误 1. 未格式化 2. 未挂载 3. 正在格式化 4. 正在删除文件 5. 正常工作 6. 只读 7. 慢速磁盘 8. 磁盘损坏
devtype	磁盘类型 0. SD 卡 1. HD 卡 2. FLASH 3. EMMC
Devtotal	设备总内存

### 5.3.7. 获取串口参数: get\_serial\_para

请求命令参数

```

{
    "cmd": "get_serial_para",
    "id": "123",
    "serial_port": 0
}

```

结果返回

```

{
    "body": {
        "baud_rate": 9600,
        "data_bits": 8,

```

```

    "parity": 0,
    "stop_bits": 1
  },
  "cmd": "get_serial_para",
  "id": "123",
  "serial_port": 1,
  "state_code": 200
}

```

字段名	说明
serial_port	串口号
baud_rate	波特率: 2400/4800/9600/19200/38400/57600/115200
data_bits	数据位: 固定 8
parity	校验位: 0. 无校验 1. 奇校验 2. 偶校验
stop_bits	停止位: 1 停止位 1 2 停止位 2

### 5.3.8. 设置串口参数: set\_serial\_para

请求命令参数

```

{
  "body": {
    "baud_rate": 37100,
    "data_bits": 8,
    "parity": 1,
    "stop_bits": 2
  },
  "cmd": "set_serial_para",
  "id": "123",
  "serial_port": 0
}

```

结果返回

```

{
  "cmd": "set_serial_para",
  "error_msg": "Sucess",
  "id": "123",
  "state_code": 200
}

```

### 5.3.9. 设置客户定制 sn 序列号

请求命令参数



```
{
  "cmd": "set_oem_sn_info",
  "id": "1324567",
  "body": {
    "oem_sn": "1234-5689"
  }
}
```

**结果返回**

```
{
  "cmd": "set_oem_sn_info",
  "error_msg": "Sucess",
  "id": "1324567",
  "state_code": 200
}
```

字段名	说明
oem_sn	客户定制的序列号

**5.3.10. 获取客户定制 SN 序列号****请求命令参数**

```
{
  "cmd": "get_oem_sn_info",
  "id": "1324567"
}
```

**结果返回**

```
{
  "cmd": "get_oem_sn_info",
  "id": "1324567",
  "state_code": 200,
  "body": {
    "oem_sn": "1234-5678"
  }
}
```

**5.3.11. 设置 Gpio 口锁定状态****请求命令参数**

```
{
  "cmd": "set_io_lock_status",
  "id": "1234568",
  "body": [
    {
      "ioout": 0,
      "status": 0
    }
  ]
}
```

```
}

```

**结果返回**

```
{
  "cmd": "set_io_lock_status",
  "error_msg": "Sucess",
  "id": "1234568",
  "state_code": 200
}
```

字段名	说明
ioout	ioout 输出口 0 或者 1
status	0 解锁 1 高电平锁定 2 低电平锁定

**5.3.12. 获取 gpio 口锁定状态****请求命令参数**

```
{
  "cmd": "get_io_lock_status",
  "id": "1234569"
}
```

**结果返回**

```
{
  "body": [
    {
      "ioout": 0,
      "status": 1
    }
  ],
  "cmd": "get_io_lock_status",
  "error_msg": "Sucess",
  "id": "1234569",
  "state_code": 200
}
```

**5.3.13. 获取交通灯状态【SJ R5D 定制】****请求命令参数**

```
{
  "cmd": "get_traffic_lights"
}
```

**结果返回**

```
{
  "body": {
    "enable": 1,
    "green_gpio": 1,
    "red_gpio": 0,

```

```

    "time" : 15000
  },
  "cmd" : "get_traffic_lights",
  "error_msg" : "Succeed",
  "id" : "999999",
  "state_code" : 200
}

```

字段名	说明
enable	是否启用交通灯控制 0: 不启用 1: 启用
green_gpio	绿灯控制 IO 口
red_gpio	红灯控制 IO 口
time	开闸绿灯延时时间(单位: ms)

### 5.3.14. 设置交通灯功能【SJ R5D 定制】

请求命令参数

```

{
  "cmd" : "set_traffic_lights",
  "body" : {
    "enable" : 1,
    "green_gpio" : 1,
    "red_gpio" : 0,
    "time" : 15000
  }
}

```

结果返回

```

{
  "cmd" : "set_traffic_lights",
  "error_msg" : "Succeed",
  "id" : "999999",
  "state_code" : 200
}

```

### 5.3.15. 获取云存历史记录 (如果没有上云查询本地存储记录)

请求命令参数

```

{
  "cmd" : "get_recg_record",
  "body" : {
    "page": 1,
    "size": 100,
    "start_time": 1715788800,
    "end_time": 1715875199,
    "plate": "5bedWDU0NjlzMQ=="
  }
}

```

## 结果返回

```
{
  "cmd": "get_recg_record",
  "error_msg": "Succeed",
  "id": "999999",
  "state_code": 200,
  "body": {
    "page": 1,
    "size": 100,
    "total": 43,
    "item": [{
      "data_source": 2,
      "trigger": 4,
      "plate": "5bedWDU0NjIzMQ==",
      "recg_time_str": "1715838936",
      "image_absolute_path":
"http://192.168.54.58/record_pic?id=VzIPCCap_C27_R4294967305",
      "image_relative_path": "VzIPCCap_C27_R4294967305"
    }, {
      "data_source": 1,
      "trigger": 4,
      "plate": "5bedWDU0NjIzMQ==",
      "recg_time_str": "1715838936",
      "image_absolute_path":
"http://192.168.6.111:9030/api/v1/ctobs/vap-dev/cbs/imgs/cc1142bc-a4d3cfa3/20240515/1715
775854_cc1142bc-a4d3cfa3_1394_18_full.jpg?token=eyJja20iOiJEsInBhaWQiOnRydWUsInN1Yil6I
mRldmIjZSIslmV4cCI6MTcxNTg0MTY4Miwic2l6ZSI6MzQ3MDY1LCJvaWQiOiJlE5NzQyOCwiY2lkIjox
MjQ5fQ.5aK8j4_NArYY1ccqxxVhl71EeTg",
      "image_relative_path":
"imgs/cc1142bc-a4d3cfa3/20240515/1715775854_cc1142bc-a4d3cfa3_1394_18_full.jpg"
    }
  ]
}
```

## 5.4. 设备连接

### 5.4.1. 通知在线消息:response\_online, response\_online\_rsp

#### 请求命令参数

```
{
  "cmd": "response_online",
  "id": "123"
}
{
  "cmd": "response_online_rsp",
  "id": "123"
}
```

#### 结果返回

当发送 cmd 为: response\_online\_rsp 时有如下返回:

```
{
  "cmd": "response_online",
  "id": "123",
  "state_code": 200
}
```

5.4.2. 主动断开连接消息:close\_socket

请求命令参数

无请求，服务器主动发送给客户端，发送完此命令后服务器会主动断开客户端。

服务器发送给客户端

```
{
  "cmd": "close_socket ",
  "state_code": 129,
  "error_msg": "close socket"
}
```

字段名	说明
cmd	命令，固定为 close_socket
state_code	错误码： <ul style="list-style-type: none"><li>● 129 应用层异步发送缓慢出现堵包</li><li>● 130 应用层写数据时连接断开</li><li>● 131 应用层连续出现错误数据断开连接</li><li>● 132 应用层接收包数据小于 0（底层读错误）</li><li>● 其他 系统层错误，对应的 code 参考 linux 下的 error code</li></ul>
error_msg	错误信息

5.5. 配置透明通道: ttransmission

请求命令参数

```
{
  "cmd": "ttransmission",
  "id": "999999",
  "subcmd": "send",
  "datalen": 6,
  "data": "QUJDREVG",
  "comm": "rs485-3"
}
{
  "cmd": "ttransmission",
  "id": "999999",
  "subcmd": "init",
  "data": "rs485-3"
}
```

```

}
{
  "cmd": "ttransmission",
  "id" : "999999",
  "subcmd": "uninit"
}

```

字段名	说明
subcmd	子命令 <ul style="list-style-type: none"> <li>● init 初始化</li> <li>● uninit 取消初始化</li> <li>● send 发送数据</li> </ul>
datalen	数据长度, 实际数据长度 (即编码前的数据长度)
data	字符串数据, 经过 base64 编码后的数据
comm	comm 口 <ul style="list-style-type: none"> <li>● rs485-1</li> <li>● rs485-2</li> </ul>
id	32 位无符号整数, 保留字段

#### 结果返回

如果初始化错误, 则返回:

```

{
  "cmd": "ttransmission",
  "id" : "999999",
  "subcmd": "init",
  "state_code" : 400,
  "response": "failed"
}

```

如果初始化成功, 则返回:

```

{
  "cmd": "ttransmission",
  "id" : "999999",
  "state_code" : 200,
  "subcmd": "init",
  "response": "ok"
}

```

如果取消初始化失败, 则返回

```

{
  "cmd": "ttransmission",
  "id" : "999999",
  "subcmd": "uninit",
  "state_code" : 400,
  "response": "failed"
}

```

如果取消初始化成功, 则返回:

```

{
  "cmd": "ttransmission",

```

```

{id": "999999",
"state_code": 200,
"subcmd": "uninit",
"response": "ok"
}

```

发送数据没有任何返回值。

字段名	说明
subcmd	子命令 <ul style="list-style-type: none"> <li>init 初始化</li> <li>uninit 取消初始化</li> <li>send 发送数据</li> </ul>
response	返回结果

## 5.6. 白名单

### 5.6.1. 脱机检查

#### 5.6.2. 注册脱机功能: reg\_offline\_check

将当前 TCP 客户端注册为脱机检测的响应终端。

请求命令参数示例

```

{
"cmd": "reg_offline_check",
"id": "999999",
"interval": 2
}

```

结果返回

```

{
"cmd": "offline",
"id": "999999",
"state_code": 200,
"response": "ok"
}

```

字段名	说明
interval	脱机响应的超时时间（秒为单位），如果超时未响应，则设备转为脱机状态。
response	返回结果

#### 5.6.3. 取消脱机注册: reg\_offline\_check

取消脱机功能

```

{
"cmd": "reg_offline_check",

```

```
"id" : "999999",
"sucmd" : "cancel"
}
```

无结果返回

5.6.4. 注册脱机事件: register\_offline\_event

请求命令参数示例

```
{
  "cmd": "register_offline_event",
  "id": "132156",
  "body": {
    "register_status": 1
  }
}
```

结果返回

```
{
  "cmd": "register_offline_event",
  "id": "132156",
  "state_code": 200,
  "error_msg": "Sucess"
}
```

未注册脱机，返回错误:

```
{
  "cmd": "register_offline_event",
  "id": "132156",
  "state_code": 400,
  "error_msg": "The session do not reg OfflineCheck,please reg it first"
}
```

当脱机状态改变且 register\_status!=0 时，收到消息:

```
{
  "cmd": "offline_status_change",
  "body": {
    "offline_status": 0
  }
}
```

字段名	说明
register_status	注册状态，当发生改变且 register_status!=0 时，会收到改变响应消息；

备注：注册脱机事件之前，需要先注册脱机功能，否则失败；

5.6.5. 获取脱机状态: get\_offline\_status

请求命令参数示例

```
{
  "cmd": "get_offline_status",
  "id": "132156"
}
```



结果回复:

```
{
  "cmd": "get_offline_status",
  "id": "132156",
  "state_code": 200,
  "body": {
    "offline_status": 1
  }
}
```

字段名	说明
offline_status	当前脱机状态; 0: 脱机, 1: 在线

### 5.6.6. 注册 OpenSDK 监听推送: register\_push\_channel

请求命令参数示例

```
{
  "cmd": "register_push_channel",
  "id": "132156",
  "body": {
    "register_status": 1
  }
}
```

结果返回

```
{
  "cmd": "register_push_channel",
  "id": "132156",
  "state_code": 200,
  "error_msg": "Success"
}
```

有推送消息且 register\_status!=0 时, 会收到推送消息:

```
{
  "cmd": "opensdk_push_message",
  "body": {
    XXXXXXXX
  }
}
```

字段名	说明
register_status	注册 opensdk 状态, 当有推送消息且 register_status!=0 时, 会收到推送消息;

### 5.6.7. 请求 OpenSDK Push: push\_msg\_to\_opensdk, 此协议不需要测试, 用于给 opensdk 发送透传消息的。

请求命令参数示例

```
{
  "cmd": "push_msg_to_opensdk",
```

```

    "id": "132156",
    "body": {
      XXXXXXX
    }
  }

```

#### 结果返回

```

{
  "cmd": "push_msg_to_opensdk",
  "id": "132156",
  "state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码

### 5.6.8.操作白名单：white\_list\_operator

#### 5.6.9. 增加或者更新白名单：update\_or\_add

##### 请求命令参数示例

```

{
  "cmd": "white_list_operator",
  "id": "123",
  "operator_type": "update_or_add",
  "dldb_rec": {
    "create_time": "2015-10-10 12:30:40",
    "enable_time": "2015-10-20 12:30:40",
    "overdue_time": "2016-10-20 12:30:40",
    "enable": 1,
    "plate": "京 A12345",
    "time_seg_enable": 0,
    "seg_time_start": "00:00:00",
    "seg_time_end": "00:00:00",
    "need_alarm": 1,
    "vehicle_code": "3254ASFDSFSD",
    "vehicle_comment": "HELOO woradf",
    "customer_id": 144413212
  }
}

```

先根据车牌号进行查找，如果在白名单中找到数据，则为更新记录。如果没有在白名单中找到数据则为添加数据。

其中时间要特别注意，必须是标准的格式：“XXXX-XX-XX XX:XX:XX”。这样才能够和数据库相互对应。

字段名	说明
operator_type	子命令 <ul style="list-style-type: none"> <li>• update_or_add 增加或更新白名单</li> <li>• delete 删除白名单</li> </ul>

	● Select 白名单查询
create_time	白名单创建时间
enable_time	白名单生效时间
overdue_time	白名单失效时间
enable	是否启动这条规则
plate	车牌号
time_seg_enable	是否启用时间段, 如果是, 那么下面 seg_time, 就会生效
seg_time	如果上面启动了时间段。这个字段就开始生效了。最长 1024 个字符
seg_time_start	示例:"00:00:00"
seg_time_end	示例:"00:00:00" 如果启用时间段, seg_time_start 和 seg_time_end 不一致, 启用时间段生效
need_alarm	是否需要报警
vehicle_code	未使用, 不需要设置
vehicle_comment	用户自定义的注释, 也是一个字符串。最长 16 个字节
customer_id	用户自己定义 ID, 是一个整数, 数据库内不保证唯一性

增加或者更新返回值:

```
{
  "cmd": "white_list_operator",
  "id": "123",
  "state_code": 200,
  "operator_type": "update_or_add",
  "state": "succeed"
}
```

字段名	说明
operator_type	子命令 <ul style="list-style-type: none"> <li>● update_or_add 增加或更新白名单</li> <li>● delete 删除白名单</li> <li>● select 白名单查询</li> </ul>
state	返回结果

### 5.6.10. 以车牌号查询白名单: plate

请求命令参数示例

```
{
  "cmd": "white_list_operator",
  "id": "999999",
  "operator_type": "select",
  "plate": "川 A07273",
  "sub_type": "plate"
}
```

这个查询是模糊查询, 如果车牌号为空的话, 那么就会查询到所有的记录。如果只有一部分, 那么就可能查询到相似的车牌信息。

字段名	说明
operator_type	子命令 <ul style="list-style-type: none"> <li>● update_or_add 增加或更新白名单</li> </ul>

	<ul style="list-style-type: none"> <li>● delete 删除白名单</li> <li>● Select 白名单查询</li> </ul>
sub_type	子命令 <ul style="list-style-type: none"> <li>● plate 根据车牌号进行查询</li> </ul>
plate	车牌号

以车牌号查询白名单返回结果：

```
{
  "cmd": "white_list_operator",
  "id": "999999",
  "state_code": 200,
  "dldb_rec": [
    {
      "create_time": "2016-03-25 14:23:58",
      "customer_id": 144413212,
      "enable": 6,
      "enable_time": "2015-12-25 11:00:00",
      "index": 1,
      "need_alarm": 1,
      "overdue_time": "2015-1-20 11:00:00",
      "plate": "川A07273",
      "seg_time_end": "12:00:00",
      "seg_time_start": "00:00:00",
      "time_seg_enable": 1,
      "vehicle_code": "1",
      "vehicle_comment": "HELOO woradf"
    }
  ],
  "operator_type": "select",
  "state": "succeed"
}
```

字段名	说明
operator_type	子命令 <ul style="list-style-type: none"> <li>● update_or_add 增加或更新白名单</li> <li>● delete 删除白名单</li> <li>● select 白名单查询</li> </ul>
index	此条记录在数据库中的 id
create_time	白名单创建时间
enable_time	白名单生效时间
overdue_time	白名单失效时间
enable	是否启动这条规则
plate	车牌号，编码格式：utf8 编码格式
time_seg_enable	是否启用时间段，如果是，那么下面 seg_time，就会生效
seg_time	如果上面启动了时间段。这个字段就开始生效了。最长 1024 个字符
seg_time_start	开始时间
seg_time_end	结束时间。如果启用时间段，seg_time_start 和 seg_time_end 不一致，启用时间段生效。
need_alarm	是否需要报警

vehicle_code	用户自定义代码，是一个字符串。最长 32 个字符，不允许重复
vehicle_comment	用户自定义的注释，也是一个字符串。最长 16 个字符
customer_id	用户自己定义 ID，是一个整数，数据库内不保证唯一性
state	返回结果状态

### 5.6.11. 白名单删除：delete

请求命令参数示例

```
{
  "cmd": "white_list_operator",
  "id": "999999",
  "operator_type": "delete",
  "plate": "京 A12345"
}
```

通过匹配车牌号删除白名单，如果车牌号为空，则清空整个白名单，否则只删除当前匹配成功的记录。

白名单删除返回结果：

```
{
  "cmd": "white_list_operator",
  "id": "999999",
  "operator_type": "delete",
  "state": "succeed",
  "state_code": 200
}
```

字段名	说明
operator_type	子命令 <ul style="list-style-type: none"> <li>● update_or_add 增加或更新白名单</li> <li>● delete 删除白名单</li> <li>● select 白名单查询</li> </ul>
state	返回结果状态
plate	车牌号

## 5.7. 识别结果加密

从安全角度考虑或者不希望第三方获取到一些信息，可调用如下接口对识别结果加密。

当前加密功能一共提供了四个接口 get\_ems、reset\_encrypt\_key、change\_encrypt\_key、enable\_encrypt。用户如果使用这些接口传输自己的明文密码的话，将会非常的不安全，所有的信息都是以明文的形式进行传输。恶意用户很容易通过抓包就能够得到设备的加密密码甚至主密码，所以对这几个接口传输的密码信息进行了加密处理。

接口中部分字段的解释：

active\_id：这一个字段指明当前选择的加密字段为多少。其中如果为 0 则为不加密，如果为其它的则为加密

signature：是一个十六位的字符串，全部的值以`0 - 9 A - Z a - z`字符构成。用户在未来的设置密码的时候，应该将这个字符串作为“消息”并结合秘钥（encrypt\_key 或者 prime\_key）使用`HMAC-SHA1`算法进行加密，再进行`Base64`编码。然后传输给设备，设备

才能够开通或者关闭加密或者修改密码或者重置密码。调用加密功能的这四个接口都会得到一个 signature，且每一次用户得到的值都会不一样。signature 一旦更新了之后，用户必须确保使用新的 signature，才能够进行接下来的认证。

**注意：**用户密码长度应该限制在[4, 16]位之前，不然会失败

可借助如下网址中的加密工具来验证设备的加密功能相关接口的正确性

HMAC-SHA 算法加密：<http://1024tools.com/hmac>

Base64 编码：[http://tomeko.net/online\\_tools/hex\\_to\\_base64.php?lang=en](http://tomeko.net/online_tools/hex_to_base64.php?lang=en)

AES 128 位算法：<http://aes.online-domain-tools.com/>

识别结果加密一般流程：

1. 获取设备支持的加密方式：get\_ems
2. 用户可修改密码：对出厂时的默认用户密码进行修改或者对已经修改过的用户密码再次修改：change\_encrypt\_key
3. 用户可重置密码：用户在忘记用户密码时也可根据主密钥重新设置用户密码：主密钥需要向我们销售要：reset\_encrypt\_key
4. 根据用户密码开启对识别结果的加密：enable\_encrypt

使用主密钥+用户密码加密的原因：如果只使用一个密码（用户密码），当用户忘记这个密码时，需要重置成默认密码，假如默认密码是“000000”，那么代表着所有设备重置一下，密码都变为“000000”。大家都知道这个情况后，设备也将不再安全。例如 A 用户的设备，为了防止 B 用户获取到 A 用户设备上的一些信息，A 用户对他的设备返回结果加了密，B 用户在不知道 A 用户密码的情况下，无法解密加密后的文件，就无法获取到有用的信息。但是当 B 知道了设备重置一下，密码就会变成“000000”，那么 B 用户可向 A 用户的设备发送重置密码命令，然后再修改成自己的密码，就可用自己的密码解密加密后的文件，就获取到了 A 用户设备上的有用信息。主密钥由我们统一管理，且每台设备的主密钥唯一

### 5.7.1. 获取加密方式：get\_ems

请求命令参数

```
{
  "cmd": "get_ems",
  "id": "999999"
}
```

结果返回

```
{
  "active_id": 1,
  "cmd": "get_ems",
  "ems": [
    {
      "m_id": 0,
      "name": "none encrypt"
    },
    {
      "m_id": 1,
      "name": "aes-128-ecb-padding-zeros"
    },
    {
      "m_id": 2,
      "name": "level 2"
    }
  ],
  "error_msg": "Sucess",
}
```

```

    "id" : "999999",
    "signature" : "MjJVbF68gsrYL8Sm",
    "state_code" : 200
}

```

字段名	说明
active_id	当前加密方式的索引
ems	加密方式列表 <ul style="list-style-type: none"> <li>● <b>m_id</b> 加密方式的索引</li> <li>● <b>name</b> 加密方式的名字</li> </ul>
signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口中的任意一个接口，signature 都会变，重置/修改密码和开启加密需采用最新的 signature）
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> </ul>

### 5.7.2. 获取用户密码：get\_encrypt\_key

用户根据主密钥获取用户密码；

请求命令参数

```

{
  "cmd" : "get_encrypt_key",
  "id" : "999999",
  "prime_key" : "Vg0MgpeBCOzzCxbsQ68V2NHd0Zk="
}

```

字段名	说明
prime_key	主密钥字符串（加密之后的密码） <ul style="list-style-type: none"> <li>● prime_key 的加密方法：由 HMAC-SHA 算法再加上原始的 prime_key，对 signature 进行加密生成数据，再进行 Base64 生成的字符串。Signature 可通过 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口获取，需采用最新的 signature</li> </ul>

注意：prime\_key 的加密与 new\_encrypt\_key 的加密可借助上面提到的网址得到结果  
结果返回

```

{
  "cmd" : "reset_encrypt_key",
  "id" : "999999",
  "signature" : "Rr17TBjPHNYXJR80",
  "encrypt_key" : "OB46x4SOg0ZTXa9BDFGh/i5LAfQ=",
  "state_code" : 200
}

```

字段名	说明
-----	----

signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口中的任意一个接口，signature 都会变，重置/修改密码和开启加密需采用最新的 signature）
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 401: 用户没有权限</li> <li>● 400: 用户参数设置错误（有可能密码太长或者太短）</li> <li>● 500: 服务器内部错误（内部设置错误）</li> </ul>
encrypt_key	encrypt_key 的加密方法：使用 AES 128 位算法，使用秘钥 prime_key（原始主密钥），对密码进行加密生成数据，再进行 Base64 生成的字符串

### 5.7.3. 重新设置用户密码：reset\_encrypt\_key

用户根据主密钥重新设置用户密码：用户在忘记自己设置的密码时可根据主密钥重新设置自己的密码

请求命令参数

```
{
  "cmd": "reset_encrypt_key",
  "id": "999999",
  "new_encrypt_key": "j6NrhO8gWFKU008mk8+A/g==",
  "prime_key": "Vg0MgpeBCOzzCxbsQ68V2NHd0Zk="
}
```

字段名	说明
prime_key	主密钥字符串（加密之后的密码） <ul style="list-style-type: none"> <li>● prime_key 的加密方法：由 HMAC-SHA 算法再加上原始的 prime_key（主密码），对 signature 进行加密生成数据，再进行 Base64 生成的字符串。Signature 可通过 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口获取，需采用最新的 signature</li> </ul>
new_encrypt_key	新设置的用户密码（加密之后的密码） <ul style="list-style-type: none"> <li>● new_encrypt_key 的加密方法：使用 AES 128 位算法，使用密码 prime_key（原始主密钥），对新的密码进行加密生成数据，再进行 Base64 生成的字符串</li> </ul>

注意：prime\_key 的加密与 new\_encrypt\_key 的加密可借助上面提到的网址得到结果  
结果返回

```
{
  "cmd": "reset_encrypt_key",
  "id": "999999",
  "signature": "Rr17TBjPHNYXJR80",
  "state_code": 200
}
```

字段名	说明
-----	----



signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口中的任意一个接口，signature 都会变，重置/修改密码和开启加密需采用最新的 signature）
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 401: 用户没有权限</li> <li>● 400: 用户参数设置错误（有可能密码太长或者太短）</li> <li>● 500: 服务器内部错误（内部设置错误）</li> </ul>

#### 5.7.4. 修改用户密码：change\_encrypt\_key

将旧的用户密码修改成新的用户密码或者将默认的用户密码修改成新的用户密码  
请求命令参数

<pre>{   "cmd": "change_encrypt_key",   "id": "9999999",   "encrypt_key": "OB46x4SOg0ZTXa9BDFGh/i5LafQ=",   "new_encrypt_key": "sZuBTdl+Fy0HjgFJ/yDmSA==" }</pre>
---

字段名	说明
encrypt_key	旧的用户密码（加密后的） <ul style="list-style-type: none"> <li>● encrypt_key 的加密方法：使用 HMAC-SHA1 算法对原始用户密码加密(encrypt_key)，加密物料使用 signature，再进行 Base64 生成字符串。</li> <li>● Signature 可通过 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口获取，需采用最新的 signature</li> </ul>
new_encrypt_key	新设置的用户密码（加密之后的密码） <ul style="list-style-type: none"> <li>● new_encrypt_key 的加密方法：使用 AES 128 位算法，使用原本的 encrypt_key（原始用户密码），对新的密码进行加密生成数据，再进行 Base64 生成的字符串</li> </ul>

注意：encrypt\_key 的加密与 new\_encrypt\_key 的加密可借助上面提到的网址得到结果  
结果返回

<pre>{   "cmd": "change_encrypt_key",   "signature": "8GZiU0f8u2slTCsp",   "id": "9999999",   "state_code": 200 }</pre>
---

字段名	说明
signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口中的任意一个接口，signature 都会变，重置/修改密码和开启加密需采用最新的 signature）

state_code:	状态码 <ul style="list-style-type: none"><li>● 200: 成功</li><li>● 401: 用户没有权限</li><li>● 400: 用户参数设置错误 (有可能密码太长或者太短)</li><li>● 500: 服务器内部错误 (内部设置错误)</li></ul>
-------------	---

如

```
std::string old_key;
std::string encrypt_key;
unsigned char digest_buffer[20] = {0};
HmacSha1((const unsigned char *) old_key.c_str(),old_key.size(),
(const unsigned char *) signature.c_str(),signature.size(), digest_buffer);
Base64::EncodeFromArray(digest_buffer, 20, &encrypt_key);
std::string new_key;
std::string new_encrypt_key;
AES128_ECB_encrypt((const unsigned char*)old_key.c_str()),(const unsigned char*)
new_key.c_str(), new_encrypt_key);
Base64::EncodeFromArray(new_encrypt_key.c_str(),new_encrypt_key.size(),&new_encrypt_key);
```

5.7.5.开启是否加密: enable\_encrypt

请求命令参数

注意: m\_id:0 为不加密, encrypt\_key 需传入用户密码 (加密后的), 而非主密钥

```
{
"cmd": "enable_encrypt",
"encrypt_key": "9k1t5PhnUXlfgJKgHmcswU3Plp8=",
"id": "999999",
"m_id": 1
}
```

字段名	说明
encrypt_key	用户密码 (加密后的) <ul style="list-style-type: none"><li>● encrypt_key 的加密方法: 使用 HMAC-SHA1 算法对原始用户密码加密(encrypt_key), 加密物料使用 signature, 再进行 Base64 生成字符串。</li><li>● 。Signature 可通过 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口获取, 需采用最新的 signature</li></ul>
m_id	加密方式的索引 (m_id:0 为不加密)

注意: encrypt\_key 的加密可借助上面提到的网址得到结果

结果返回

```
{
"cmd": "enable_encrypt",
"signature": "BUqCmKfikt7dYEOv",
"id": "999999",
"state_code": 200
}
```

字段名	说明
signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt 四个接口中的任意一个接口, signature 都会变，重置/修改密码和开启加密需采用最新的 signature)
State_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 401: 用户没有权限</li> <li>● 400: 用户参数设置错误 (m_id 错误)</li> <li>● 500: 服务器内部错误 (内部设置错误)</li> </ul>

### 5.7.6. 设置设备有效时间: device\_active\_settings

设置启用授权后，超过相机授权时间后，相机不再推送识别结果；

#### 请求命令参数

```
{
  "cmd": "device_active_settings",
  "id": "123",
  "body": {
    "type": "set_device_active_status",
    "active_status": 1,
    "active_time": 60,
    "authentication": "asdgzxb"
  }
}
```

#### 回复结果:

```
{
  "body": {
    "state": 200,
    "type": "set_device_active_status"
  },
  "cmd": "device_active_settings",
  "error_msg": "Sucess",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
active_status	是否启用相机授权 0 不启用 1 启用 启用授权后，在超过授权时间后，相机不再推送识别结果信息；
active_time	相机授权时间，单位秒；

authentication	加密后再经过 64 位编码的用户密码 使用 SHA1 算法对用户密码加密，加密物料使用签名，再进行 64 位编码后的字符串；
error_msg	记录失败或者成功的详细说明

### 5.7.7. 获取设备有效时间：device\_active\_settings

请求命令参数

```
{
  "cmd": "device_active_settings",
  "id": "123",
  "body": {
    "type": "get_device_active_status"
  }
}
```

回复结果：

```
{
  "body": {
    "active_status": 1,
    "active_time": 1200,
    "signature": "cR6dxChrE42JufOQ",
    "state": 200,
    "type": "get_device_active_status"
  },
  "cmd": "device_active_settings",
  "error_msg": "Sucess",
  "id": "123",
  "state_code": 200
}
```

### 5.7.8. 设置用户私有数据：set\_user\_data

请求命令参数

```
{
  "cmd": "set_user_data",
  "id": "123",
  "body": {
    "data": "MTMMyNDU2Nzg5"
  }
}
```

回复结果：

```
{
  "cmd": "set_user_data",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
data	需要设置的用户数据，经过 base64 位编码

### 5.7.9. 获取用户私有数据：get\_user\_data

请求命令参数

```
{
  "cmd": "get_user_data",
  "id": "123"
}
```

回复结果：

```
{
  "body": {
    "data": "MTMyNDU2Nzg5"
  },
  "cmd": "get_user_data",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
data	需要设置的用户数据，经过 base64 位编码

## 5.8. 设备组网：dg\_json\_request

设备组网不同于传统的将一台设备的信息提供给用户，需是自动的将一定区域内的所有设备信息提供给用户，以便应用层用户能够更加方便的接入我们的设备系统，更加快速的开发自己的应用。

设备组网功能主要应用在停车场环境，一个停车场里面的所有设备之间相互连接组成一个网络。当有一辆车进入或者离开整个停车场的时候，设备之间进行信息的共享和同步，并且根据这些数据分析出一个最优的结果，然后传输给用户，以方便应用开发，甚至直接收费。

设备组网协议主要分为两个大部分。第一步是关于识别结果的协议。第二部分是关于各种操作组网的协议。

请求命令参数如下格式：

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "%s"
  }
  ... //此处为组网请求细分消息中携带的参数，具体如下
}
```

结果返回

0. 处理成功：

```
{
```

```
"cmd": "dg_json_request",
"id": "132156",
"state_code": 200,
"body": {
    "state": 200,
    "type": "get_cdvzid",
    ..... // 具体请求具体回复
}
```

1. 格式错误:

```
{
"cmd": "dg_json_request",
"id": "132156",
"state_code": 400
}
```

2. 服务器处理超时错误:

```
{
"cmd": "dg_json_request",
"id": "132156",
"state_code": 408
}
```

3. 服务器内部错误:

```
{
"cmd": "dg_json_request",
"id": "132156",
"state_code": 500
}
```

5.8.1. 得到当前设备 vzid: get\_cdvzid

我们当前将每一个设备的名称定义为 vzid ，用来标记这一台设备  
请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_cdvzid"
  }
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据
type	操作组网模块的命令 <ul style="list-style-type: none"><li>get_cdvzid 获取当前设备 vzid</li></ul>

## 结果返回

```
{
  "body": {
    "state": 200,
    "type": "get_cdvzid",
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.7.22",
      "name": "bHk=&MTkyljE2OC43Ljly#",
      "sn": "56a8872e-2c5f7863",
      "type": "input"
    }
  },
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 get_cdvzid 获取当前设备 vzid
vzid	设备的名称，vzid 说明如下
state	状态码

vzid 字段说明：

字段名	说明
enable_group	开启组网功能 <ul style="list-style-type: none"> <li>● true 开启</li> <li>● false 关闭</li> </ul>
ip_addr	设备 ip
name	设备名字
sn	设备序列号
type	设备类型 <ul style="list-style-type: none"> <li>● input 入口设备</li> <li>● output 出口设备</li> </ul>

## 5.8.2. 得到当前设备名称：get\_current\_device\_name

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_current_device_name"
  }
}
```

```

    }
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● <code>get_current_device_name</code> 得到当前设备名称</li> </ul>

结果返回

返回结果为当前设备 `vzid` 的信息，和 2.8.1 返回结果一致，请参考：

### 5.8.3. 得到在线设备信息，不含自己：get\_ovzid

请求命令参数

```

{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_ovzid"
  }
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● <code>get_ovzid</code> 得到所有已连接的设备信息</li> </ul>

结果返回

```

{
  "body": {
    "state": 200,
    "type": "get_ovzid",
    "vzids": [
      {
        "enable_group": true,
        "ip_addr": "192.168.2.29",
        "name": "2.29 TEST",
        "sn": "0f692718-95bc0185",
        "type": "output"
      },
      {
        "enable_group": true,
        "ip_addr": "192.168.2.77",
        "name": "2.77 TEST",
        "sn": "ef69b450-21bf3bff",

```



```

        "type": "input"
    },
    {
        "enable_group": true,
        "ip_addr": "192.168.2.79",
        "name": "2.79 TEST",
        "sn": "c9455af2-68daf655",
        "type": "output"
    },
    {
        "enable_group": true,
        "ip_addr": "192.168.2.9",
        "name": "2.9 TEST",
        "sn": "037de919-76c16e36",
        "type": "output"
    }
]
},
"cmd": "dg_json_request",
"id": "132156",
"error_msg": "Sucess",
"state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>get_ovzid 得到所有已连接的设备信息</li> </ul>
vzid	设备信息列表（由多条设备信息组成），一条设备信息，即 vzid，vzid 上面已经提到过了

#### 5.8.4. 得到在线设备信息，含自己：online\_devices

请求命令参数

```

{
    "cmd": "dg_json_request",
    "id": "132156",
    "body": {
        "type": "online_devices"
    }
}

```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>online_devices 得到所有已连接的设备信息</li> </ul>

结果返回：

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200,
  "error_msg": "Sucess",
  "body": {
    "state": 200,
    "type": "online_devices",
    "online_devices": [
      {
        "enable_group": true,
        "ip_addr": "192.168.7.22",
        "name": "bHk=&MTkyLjE2OC43Ljly#",
        "sn": "56a8872e-2c5f7863",
        "type": "input"
      },
      {
        "enable_group": true,
        "ip_addr": "192.168.7.22",
        "name": "bHk=&MTkyLjE2OC43Ljly#",
        "sn": "56a8872e-2c5f7863",
        "type": "input"
      }
    ]
  }
}
```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"><li>● online_devices 得到所有已连接的设备信息</li><li>● online_devices 中的数据结构与上面 vziids 的数据一致，请参考</li></ul>

5.8.5. 得到所有连接设备信息：get\_avzid

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_avzid"
  }
}
```

结果返回：  
结果返回以及字段信息和上面 2.8.3 一致，请参考；

5.8.6. 得到当前组网内所有设备信息：get\_agdi

请求命令参数

```
{
  "cmd": "dg_json_request",
```

```
{
  "id": "132156",
  "body": {
    "type": "get_agdi"
  }
}
```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"><li>● <b>get_agdi</b> 得到当前组网内所有设备信息</li></ul>

结果返回：  
结果返回以及字段信息和上面 2.8.3 一致，请参考；

5.8.7. 得到当前设备记录 size: **current\_records\_size**

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "current_records_size"
  }
}
```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"><li>● <b>current_records_size</b> 得到当前设备记录的 size</li></ul>

结果返回：  
1. 入口设备：

```
{
  "body": {
    "state": 200,
    "type": "current_records_size",
    "input_records_size": 1024,
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.7.22",
      "name": "bHk=&MTkyLjE2OC43Ljly#",
      "sn": "56a8872e-2c5f7863",
      "type": "input"
    }
  },
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
}
```

2. 出口设备：

```
{
  "body": {
    "state": 200,
    "type": "current_records_size",
  }
}
```

```

"output_records_size": 1024,
"vzid" : {
  "enable_group" : true,
  "ip_addr" : "192.168.7.22",
  "name" : "bHk=&MtkyLjE2OC43Ljly#",
  "sn" : "56a8872e-2c5f7863",
  "type" : "output"
},
"cmd" : "dg_json_request",
"id" : "132156",
"state_code" : 200
}

```

字段名	说明
output_records_size	● 出口设备记录的 size
input_records_size	● 入口设备记录的 size
Vzid	● 同时返回当前设备 vzid 信息; vzid 信息参考前面;

### 5.8.8.得到入口设备记录: records\_sparate\_input

由于设备记录有很多，一般情况下很难一次性传输出来，所以，在得到设备记录的时候必须指定一个范围。在最开始的时候如果不知道记录有多少条，可以随便指定一个可以传输的范围，32 最好。然后在回复的消息中，会告诉当前设备有多少条记录。这样的分页机制，让用户可以更好的开发自己的应用

请求命令参数

```

{
  "body": {
    "input_begin_pos": 0,
    "input_end_pos": 8,
    "type": "records_sparate_input"
  },
  "cmd": "dg_json_request",
  "id": "123456789"
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据
type	请求回应消息

Body 内的字段说明:

字段名	说明
type	操作组网模块的命令 ● records_sparate_input 获取入口设备记录
input_begin_pos	获取记录的开始位置
input_end_pos	获取记录的结束位置

结果返回

```
{
```

```
"body": {
  "input_begin_pos": 0,
  "input_end_pos": 8,
  "input_records_size": 10213,
  "records": [
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxxx",
      "enter_time": 1494049723,
      "plate": "渝 AX4462",
      "state": 5
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxxxx",
      "enter_time": 1494049731,
      "plate": "琼 ECVWJB",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxx",
      "enter_time": 1494049739,
      "plate": "豫 JDZBHF",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxx",
      "enter_time": 1494049754,
      "plate": "冀 SCNGRR",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxxx",
      "enter_time": 1494049762,
      "plate": "新 GMKA3M",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxx",
      "enter_time": 1494049770,
      "plate": "赣 QFWXF5",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxx",
      "enter_time": 1494049778,
      "plate": "豫 X4JK1G",
      "state": 6
    }
  ]
}
```

```

    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxx",
      "enter_time": 1494049784,
      "plate": "豫 QWY8KH",
      "state": 6
    },
    {
      "enter_ip": "192.168.5.13",
      "enter_name": "xxxxxx",
      "enter_time": 1494049793,
      "plate": "沪 H1JLJ0",
      "state": 6
    }
  ],
  "state": 200,
  "type": "records_sparate_input",
  "vzid": {
    "enable_group": true,
    "ip_addr": "192.168.7.22",
    "name": "bHk=&MTkyLjE2OC43Ljly#",
    "sn": "56a8872e-2c5f7863",
    "type": "input"
  }
},
"cmd": "dg_json_request",
"id": "123456789",
"state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

Body 内的字段说明：

字段名	说明
input_begin_pos	获取记录的开始位置
input_end_pos	获取记录的结束位置
input_records_size	入口设备记录总条数
records	入口设备记录的列表（由多条入口记录组成）
state	状态码
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>records_sparate_input 获取入口设备记录</li> </ul>
vzid	设备的名称，vzid 说明如下

records 内的字段说明：

字段名	说明
enter_ip	ip 地址
enter_name	设备名称(base64 编码)

enter_time	识别时间点
plate	车牌号码 (汉字为 GB2312 编码)
state	设备的名称, vzyd 说明如下

vzyd 字段说明:

字段名	说明
enable_group	开启组网功能 <ul style="list-style-type: none"> <li>● true 开启</li> <li>● false 关闭</li> </ul>
ip_addr	设备 ip
name	设备名字
sn	设备序列号
type	设备类型 <ul style="list-style-type: none"> <li>● input 入口设备</li> <li>● output 出口设备</li> </ul>

### 5.8.9. 得到出口设备记录: records\_sparate\_output

请求命令参数

<pre>{   "body": {     "output_records_begin_pos": 0,     "output_records_end_pos": 8,     "type": "records_sparate_output"   },   "cmd": "dg_json_request",   "id": "123456789" }</pre>
--

字段名	说明
id	序列字符串 (唯一), 字符串长度 < 30
body	需要传递给组网模块的数据

Body 内的字段说明:

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● records_sparate_outpu 获取出口设备记录</li> </ul>
output_records_begin_pos	获取记录的开始位置
output_records_end_pos	获取记录的结束位置

结果返回

<pre>{   "body": {     "output_records_begin_pos": 0,     "output_records_end_pos": 8,     "output_records_size": 4549,     "records": [       {         "enter_ip": "192.168.4.79",         "enter_name": "5ryr5q2l5LqR56uv&amp;MTkyLjE2OC40Ljc5#", </pre>
---

```

        "enter_time": 1465496977,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1465496992,
        "plate": "粤 Z72D2 港",
        "state": 0
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1465497011,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1465497011,
        "plate": "粤 Z72D2 港",
        "state": 3
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1465497051,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1465497051,
        "plate": "粤 Z72D2 港",
        "state": 3
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1468906933,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468906933,
        "plate": "粤 Z72D2 港",
        "state": 3
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1468906941,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468906941,
        "plate": "粤 Z72D2 港",
        "state": 3
    },
    {
        "enter_ip": "192.168.4.79",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljc5#",
        "enter_time": 1468906962,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468906967,

```



```

        "plate": "粤 Z72D2 港",
        "state": 0
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1468907000,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468907000,
        "plate": "粤 Z72D2 港",
        "state": 3
    },
    {
        "enter_ip": "192.168.4.79",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljc5#",
        "enter_time": 1468907153,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468907367,
        "plate": "粤 Z72D2 港",
        "state": 0
    },
    {
        "enter_ip": "192.168.4.97",
        "enter_name": "5ryr5q2l5LqR56uv&MTkyLjE2OC40Ljk3#",
        "enter_time": 1468907448,
        "leave_ip": "192.168.7.22",
        "leave_name": "bHk=&MTkyLjE2OC43Ljly#",
        "leave_time": 1468907448,
        "plate": "粤 Z0V89 港",
        "state": 3
    }
],
"state": 200,
"type": "records_sparate_output",
"vzid": {
    "enable_group": true,
    "ip_addr": "192.168.7.22",
    "name": "bHk=&MTkyLjE2OC43Ljly#",
    "sn": "56a8872e-2c5f7863",
    "type": "output"
}
},
"cmd": "dg_json_request",
"id": "123456789",
"state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

Body 内的字段说明：

字段名	说明
output_records_begin_pos	获取记录的开始位置
output_records_end_pos	获取记录的结束位置
output_records_size	出口设备记录总条数
records	出口设备记录的列表（由多条出口记录组成，一条出口记录由此车牌的入口记录信息+此车牌的出口记录信息组成）
state	状态码
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● <b>records_sparate_output</b> 获取出口设备记录</li> </ul>
vzid	设备的名称

vzid 字段说明：

字段名	说明
enable_group	开启组网功能 <ul style="list-style-type: none"> <li>● true 开启</li> <li>● false 关闭</li> </ul>
ip_addr	设备 ip
name	设备名字
sn	设备序列号
type	设备类型 <ul style="list-style-type: none"> <li>● input 入口设备</li> <li>● output 出口设备</li> </ul>

#### 5.8.10. 使能设备组网：enable\_devicegroup

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "enable_devicegroup",
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.7.22",
      "name": "bHk=&MTkyLjE2OC43Ljly#",
      "sn": "56a8872e-2c5f7863",
      "type": "input"
    }
  }
}
```

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● <b>enable_devicegroup</b> 使能设备组网</li> </ul>
vzid	需要使能的设备的 vzid Vzid 中的字段意思，参考 2.8.1 中关于 vzid 的描述

结果返回

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
  "body": {
    "state": 200,
    "type": "enable_devicegroup"
  }
}
```

5.8.11. 设置设备组网类型及参数: set\_device\_type\_enable

功能和 2.8.10 一致  
请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "set_device_type_enable",
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.7.22",
      "name": "bHk=&MTkyLjE2OC43Ljly#",
      "sn": "56a8872e-2c5f7863",
      "type": "input"
    }
  }
}
```

Body 内的字段说明:

字段名	说明
type	操作组网模块的命令 ● set_device_type_enable 设置设备组网
vzid	需要使能的设备的 vzid
	Vzid 中的字段意思, 参考 2.8.1 中关于 vzid 的描述

结果返回

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
  "body": {
    "state": 200,
    "type": "set_device_type_enable"
  }
}
```

5.8.12. 查找车牌信息: search\_plate

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "search_plate",
    "plate": "川 A12345",
    "record_type": "input",
    "record_state": 5
  }
}
```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"><li>search_plate 查找车牌信息</li></ul>
Plate	车牌号
record_type	记录类型 入场设备; <ul style="list-style-type: none"><li>input : 入场</li><li>output:出场</li></ul> select 白名单查询
record_state	记录状态

结果返回

```
{
  "body": {
    "search_result": [
      {
        "input_record": {
          "device_name": {
            "ast_type": 0,
            "dev_name": "",
            "enable_group": true,
            "ip_addr": "192.168.55.90",
            "name": "6L2m5bqTNg==&MTkyLjE2OC41NS45MA==#",
            "sn": "e7e40437-580ec15e",
            "type": "input"
          },
          "ivs_result_param": {
            "bright": 0,
            "car_bright": 0,
            "car_color": 255,
            "confidence": 80,
            "direction": 4,
            "fragment_path": "VzIPCCap_C4328522066_R2",
            "id": 1,
            "image_path": "VzIPCCap_C4311744617_R1",
            "image_sd_path": "VzIPCCap_C4311744617_R1",
            "location": {
              "bottom": 928,
```

```

        "left": 290,
        "right": 684,
        "top": 797
    },
    "n_time": 0,
    "plate": "|| A12345",
    "plate_color": 1,
    "plate_type": 1,
    "timeval": {
        "tv_sec": 1672888437,
        "tv_usec": 327320
    },
    "trig_type": 8
},
"state": 6
},
"record_type": "input"
},
{
    "input_record": {
        "device_name": {
            "ast_type": 0,
            "dev_name": "",
            "enable_group": true,
            "ip_addr": "192.168.55.90",
            "name": "6L2m5bqTNg==MTkyLjE2OC41NS45MA==#",
            "sn": "e7e40437-580ec15e",
            "type": "input"
        },
        "ivs_result_param": {
            "bright": 0,
            "car_bright": 0,
            "car_color": 255,
            "confidence": 75,
            "direction": 4,
            "fragment_path": "VzIPCCap_C8623554948_R4",
            "id": 2,
            "image_path": "VzIPCCap_C8606777439_R3",
            "image_sd_path": "VzIPCCap_C8606777439_R3",
            "location": {
                "bottom": 908,
                "left": 319,
                "right": 710,
                "top": 803
            },
            "n_time": 0,
            "plate": "|| A12345",
            "plate_color": 1,
            "plate_type": 1,
            "timeval": {
                "tv_sec": 1672888664,
                "tv_usec": 482974
            },
            "trig_type": 4
        }
    }
}

```

```
        },
        "state": 6
    },
    "record_type": "input"
}
],
"state": 200,
"type": "search_plate"
},
"cmd": "dg_json_request",
"error_msg": "Sucess",
"id": "132156",
"state_code": 200
}
```

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"><li>● search_plate 查找车牌信息</li></ul>

search\_result 中字段含义:

字段名	说明
record_type	● 当前记录是出场还是入场记录
output_record	● 出场记录
input_record	● 入场记录
ivs_result_param	● 车辆信息, 结构体
State	● 当前车辆状态
device_name	● 设备名称

ivs\_result\_param 中字段含义:  
参考 2.2.1 中车牌识别结果中字段含义

5.8.13. 获取组网匹配模式: get\_device\_match\_mode

请求命令参数

```
{
    "cmd": "dg_json_request",
    "id": "132156",
    "body": {
        "type": "get_device_match_mode"
    }
}
```

结果返回

```
{
    "cmd": "dg_json_request",
    "id": "132156",
    "state_code": 200
    "body": {
        "state": 200,
        "type": " get_device_match_mode ",
        "mode": " fuzzy_mode "
    }
}
```

字段名	说明
mode	组网匹配模式： fuzzy_mode: 模糊匹配 exact_mode: 精确匹配

#### 5.8.14. 设置组网匹配模式: set\_device\_match\_mode

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "set_device_match_mode",
    "mode": "fuzzy_mode"
  }
}
```

结果返回

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
  "body": {
    "state": 200,
    "type": "set_device_match_mode"
  }
}
```

字段名	说明
mode	组网匹配模式： fuzzy_mode: 模糊匹配 exact_mode: 精确匹配

#### 5.8.15. 得到组网共享 IO: get\_group\_shared\_io

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_group_shared_io"
  }
}
```

结果返回

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
}
```

```

"body": {
  "state": 200,
  "type": "get_group_shared_io",
  "value": 1
}

```

字段名	说明
value	当前组网共享的 IO 的值

### 5.8.16. 设置组网共享 IO: set\_group\_shared\_io

请求命令参数

```

{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "set_group_shared_io",
    "value": 1
  }
}

```

结果返回

```

{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
  "body": {
    "state": 200,
    "type": "set_group_shared_io"
  }
}

```

### 5.8.17. 组网识别结果消息 enable\_dg\_result

识别结果消息分成了两种类型一种类型是入口设备的识别结果消息，一种是出口设备的识别结果消息。用户首先需要开启允许接收组网识别结果消息，未来，如果有车牌识别消息到来，用户才会接收到出/入口消息

请求命令参数

```

{
  "cmd": "enable_dg_result",
  "enable": true,
  "id": "123456"
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
enable	是否允许接收组网识别消息 <ul style="list-style-type: none"> <li>● true 允许接收组网识别消息</li> </ul>



	● false 不接收组网识别消息
--	-------------------

## 结果返回

```
{
  "cmd": "enable_dg_result",
  "id": "123456",
  "state_code": 200
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码

当开启接收组网识别结果后，当组网设备有识别到结果，会主动推送给用户，推送信息如下：

## 出口消息

```
{
  "body": {
    "record_type": "output",
    "input_record": {
      "device_name": {
        "enable_group": true,
        "ip_addr": "192.168.4.180",
        "name": "4.180Leave",
        "sn": "7667931c-7fa08a25",
        "type": "output"
      },
      "ivs_result_param": {
        "bright": 0,
        "car_bright": 0,
        "car_color": 0,
        "confidence": 98,
        "direction": 4,
        "fragment_path": "/tmp/app/html/snapshot/lpr/patch_tri_snap_4.jpg",
        "id": 0,
        "image_path": "/tmp/app/html/snapshot/lpr/tri_snap_4.jpg",
        "image_sd_path": "/media/mmcblk0p1/VzIPCCap/2015_11_12/1534433501_ 陕
D4FY61.jpg",
        "location": {
          "bottom": 309,
          "left": 965,
          "right": 1150,
          "top": 268
        },
        "n_time": 0,
        "plate": "陕 D4FY61",
        "plate_color": 1,
        "plate_type": 0,
        "related_plate": "陕 D4FY61",
        "timeval": {
          "decday": 5,
          "dechour": 15,
          "decmin": 26,
          "decmon": 5,
          "decsec": 14,
```

```

        "decyear": 2017,
        "tv_sec": 1493969174,
        "tv_usec": 416656
    },
    "trig_type": 1
},
"state": 0
},
"output_record": {
    "device_name": {
        "enable_group": true,
        "ip_addr": "192.168.4.180",
        "name": "4.180Leave",
        "sn": "7667931c-7fa08a25",
        "type": "output"
    },
    "ivs_result_param": {
        "bright": 0,
        "car_bright": 0,
        "car_color": 0,
        "confidence": 98,
        "direction": 4,
        "fragment_path": "/tmp/app/html/snapshot/lpr/patch_tri_snap_4.jpg",
        "id": 0,
        "image_path": "/tmp/app/html/snapshot/lpr/tri_snap_4.jpg",
        "image_sd_path": "/media/mmcblk0p1/VzIPCCap/2015_11_12/1534433501_ 陕
D4FY61.jpg",
        "location": {
            "bottom": 309,
            "left": 965,
            "right": 1150,
            "top": 268
        },
        "n_time": 0,
        "plate": "陕 D4FY61",
        "plate_color": 1,
        "plate_type": 0,
        "related_plate": "陕 D4FY61",
        "timeval": {
            "decday": 5,
            "dechour": 15,
            "decmin": 26,
            "decmon": 5,
            "decsec": 14,
            "decyear": 2017,
            "tv_sec": 1493969174,
            "tv_usec": 416656
        },
        "trig_type": 1
    },
    "state": 0
}
},
"cmd": "dg_plateinfo_result",

```

```

    "id": "132156",
    "state_code": 200
  }

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码
cmd	操作命令 <ul style="list-style-type: none"> <li>● dg_plateinfo_result 组网识别消息结果</li> </ul>

Body 内的字段说明：

字段名	说明
record_type	消息类型 <ul style="list-style-type: none"> <li>● output 出口消息</li> <li>● input 入口消息</li> </ul>
input_record	上面已经提到过了，请自行查找
output_record	上面已经提到过了，请自行查找

入口消息

```

{
  "body": {
    "record_type": "input",
    "input_record": {
      "device_name": {
        "enable_group": true,
        "ip_addr": "192.168.4.157",
        "name": "4.157Enter",
        "sn": "0ba76bc6-e26d52eb",
        "type": "input"
      },
      "ivs_result_param": {
        "bright": 0,
        "car_bright": 0,
        "car_color": 0,
        "confidence": 99,
        "direction": 0,
        "fragment_path": "/tmp/app/html/snapshot/lpr/patch_tri_snap_16.jpg",
        "id": 0,
        "image_path": "/tmp/app/html/snapshot/lpr/tri_snap_16.jpg",
        "image_sd_path": "/media/mmcblk0p1/VzIPCCap/2015_11_12/1533300901_陕
A8T00 学.jpg",
        "location": {
          "bottom": 590,
          "left": 587,
          "right": 696,
          "top": 571
        },
        "n_time": 0,
        "plate": "陕 A8T00 学",
        "plate_color": 13,
        "plate_type": 0,
        "related_plate": "陕 A8T00 学",
        "timeval": {

```

```

        "decday": 5,
        "dechour": 15,
        "decmin": 26,
        "decmon": 5,
        "decsec": 14,
        "decyear": 2017,
        "tv_sec": 1493969174,
        "tv_usec": 416656
    },
    "trig_type": 1
},
"state": 1
}
},
"cmd": "dg_plateinfo_result",
"id": "132156",
"state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code	状态码
cmd	操作命令 dg_plateinfo_result 组网识别消息结果

Body 内的字段说明：

字段名	说明
record_type	消息类型 output 出口消息 input 入口消息
input_record	上面已经提到过了，请自行查找

### 5.8.18. 清除组网数据：reset\_database

请求命令参数

```

{
  "body": {
    "type": "reset_database"
  },
  "cmd": "dg_json_request",
  "id": "132156"
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据

Body 内的字段说明：

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>reset_database 清除组网数据</li> </ul>

## 结果返回

```
{
  "body": {
    "state": 200,
    "type": "reset_database"
  },
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

Body 内的字段说明:

字段名	说明
type	操作组网模块的命令 reset_database 清除组网数据
state	状态码

## 5.8.19. 得到当前组网内部所有设备配置: get\_group\_cfg

## 请求命令参数

```
{
  "body": {
    "type": "get_group_cfg"
  },
  "cmd": "dg_json_request",
  "id": "132156"
}
```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据

Body 内的字段说明:

字段名	说明
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>get_group_cfg 得到当前组网内部所有设备配置</li> </ul>

## 结果返回

```
{
  "body": {
    "group_cfg": {
      "group_vzids": [
        {
          "connect_status": 5,
          "enable_group": true,
          "ip_addr": "192.168.105.39",
          "name": ""
        }
      ]
    }
  }
}
```

```

        "sn": "018cd419-5e0702a4",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.103.16",
        "name": "",
        "sn": "0a0248b7-f9121dba",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.8.94",
        "name": "",
        "sn": "10391362-00e22a20",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.4.186",
        "name": "6L2m5bqTNQ==&55u46L6FNTk=&MTkyLjE2OC41LjEx#dHJIZURlbW9fMTU=",
        "sn": "169d39b6-34645de7",
        "type": "input"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.4.132",
        "name": "",
        "sn": "3acd054f-6b3c0af3",
        "type": "input"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.1.7",
        "name": "",
        "sn": "702cdcaa-597a919f",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.102.40",
        "name": "",
        "sn": "783f0ea9-90485466",
        "type": "unkown"
    },
    {
        "connect_status": 5,

```

```

        "enable_group": true,
        "ip_addr": "192.168.22.71",
        "name": "",
        "sn": "7a827083-62ad93cd",
        "type": "input"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.3.3",
        "name": "",
        "sn": "7aea8606-8d88dd06",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.102.4",
        "name": "",
        "sn": "a9132b96-4af3b688",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.4.137",
        "name": "",
        "sn": "abf20a38-23f0b33d",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.1.187",
        "name": "",
        "sn": "abf775b8-bf50d0b2",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.105.38",
        "name": "",
        "sn": "af52ed95-8deaeadb",
        "type": "unkown"
    },
    {
        "connect_status": 5,
        "enable_group": true,
        "ip_addr": "192.168.102.41",
        "name": "
"ODEyNyOt5Yir5Yqo&55u46L6FMQ==&MTkyLjE2OC4xMDIuNDE=#dHJIZURibW9fNg==",
        "sn": "b189c806-faec3304",
        "type": "output"
    }

```

```

    },
    {
      "connect_status": 5,
      "enable_group": true,
      "ip_addr": "192.168.12.251",
      "name": "",
      "sn": "b1a7806e-326e00a2",
      "type": "unkown"
    },
    {
      "connect_status": 5,
      "enable_group": true,
      "ip_addr": "192.168.20.10",
      "name": "",
      "sn": "ed2045b2-30acf3a2",
      "type": "unkown"
    },
    {
      "connect_status": 5,
      "enable_group": true,
      "ip_addr": "192.168.102.56",
      "name": "",
      "sn": "f0565a83-fd3be473",
      "type": "unkown"
    },
    {
      "connect_status": 5,
      "enable_group": true,
      "ip_addr": "192.168.105.22",
      "name": "",
      "sn": "fcf2e551-a44c1cce",
      "type": "unkown"
    },
    {
      "connect_status": 5,
      "enable_group": true,
      "ip_addr": "192.168.105.14",
      "name": "",
      "sn": "ffedb27b-cd76bd51",
      "type": "unkown"
    }
  ],
  "state": 200,
  "type": "get_group_cfg",
  "vzid": {
    "enable_group": true,
    "ip_addr": "192.168.7.22",
    "name": "bHk=&MTkyLjE2OC43Ljly#",
    "sn": "56a8872e-2c5f7863",
    "type": "output"
  }
},
"cmd": "dg_json_request",

```



```

    "id": "132156",
    "state_code": 200,
    "type": null
  }

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

body 内的字段说明：

字段名	说明
group_cfg	组网内所有设备配置
state	状态码
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● get_group_cfg 得到当前组网内部所有设备配置</li> </ul>
vzid	设备的名称

group\_cfg 内的字段说明：

字段名	说明
group_vzids	组网内所有设备配置信息)(由多条设备配置信息组成)

group\_vzids 内的字段说明：

字段名	说明
connect_status	连接状态
enable_group	开启组网功能 <ul style="list-style-type: none"> <li>● true 开启</li> <li>● false 关闭</li> </ul>
ip_addr	IP 地址
name	设备的名称
sn	设备序列号
type	设备类型 <ul style="list-style-type: none"> <li>● input 入口设备</li> <li>● output 出口设备</li> </ul>

vzid 字段说明：

字段名	说明
enable_group	开启组网功能 <ul style="list-style-type: none"> <li>● true 开启</li> <li>● false 关闭</li> </ul>
ip_addr	设备 ip
name	设备名字
sn	设备序列号
type	设备类型 <ul style="list-style-type: none"> <li>● input 入口设备</li> <li>● output 出口设备</li> </ul>

**5.8.20. 设置当前组网内部所有设备配置: set\_group\_cfg**

请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "set_group_cfg",
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.7.22",
      "name": "bHk=&MTkyLjE2OC43Ljly#",
      "sn": "56a8872e-2c5f7863",
      "type": "input"
    },
    "group_cfg": [
      {
        "enable_group": true,
        "ip_addr": "192.168.1.2",
        "name": "bHk=&asd4231#",
        "sn": "1238872e-2c5fasdv",
        "type": "input",
        "connect_status": 2
      },
      {
        "enable_group": true,
        "ip_addr": "192.168.22.12",
        "name": "bHk=&456sdg13#",
        "sn": "asxc887e-2cafasdv",
        "type": "input",
        "connect_status": 2
      }
    ]
  }
}
```

结果返回

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
  "body": {
    "state": 200,
    "type": "set_group_cfg"
  }
}
```

**5.8.21. 根据 ID 获取组网图片: get\_img\_by\_id**

请求命令参数

```
{
  "body": {
```

```

    "image_id": 79509,
    "image_type": 1,
    "sn": "ef69b450-21bf3bff",
    "type": "get_img_by_id"
  },
  "cmd": "dg_json_request",
  "id": "132156"
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30
body	需要传递给组网模块的数据

body 内的字段说明：

字段名	说明
image_id	图片 id
image_type	图片类型
sn	设备序列号
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● get_img_by_id 得到当前组网内部所有设备配置</li> </ul>

结果返回

```

{
  "body": {
    "image_id": 79509,
    "image_type": 1,
    "state": 404,
    "type": "get_img_by_id"
  },
  "cmd": "dg_json_request",
  "id": "132156",
  "state_code": 200
}

```

字段名	说明
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
body	组网模块处理后返回的数据
state_code	状态码

body 内的字段说明：

字段名	说明
image_id	图片 id
image_type	图片类型
state	状态码
type	操作组网模块的命令 <ul style="list-style-type: none"> <li>● get_img_by_id 得到当前组网内部所有设备配置</li> </ul>

## 5.9. 用户登录

### 5.9.1. 请求开始登录:start\_login

请求命令参数

```
{
  "cmd": "start_login",
  "id": "123456"
}
```

结果返回

```
{
  "active_id": 0,
  "cmd": "start_login",
  "device_sn": "b77b647f-277f2aa9",
  "ems": [
    {
      "m_id": 0,
      "name": "none encrypt"
    },
    {
      "m_id": 1,
      "name": "aes-128-ecb-padding-zeros"
    },
    {
      "m_id": 2,
      "name": "level 2"
    }
  ],
  "id": "123456",
  "signature": "Rr17TBjPHNYXJR80",
  "state_code": 200
}
```

字段名	说明
active_id	当前加密方式的索引
ems	加密方式列表 <ul style="list-style-type: none"><li>● <b>m_id</b> 加密方式的索引</li><li>● <b>name</b> 加密方式的名字</li></ul>
signature	一个十六位的字符串（用于对传输的密码进行加密，每调用一次 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt/start_login 四个接口中的任意一个接口, signature 都会变, 重置/修改密码和开启加密需采用最新的 signature）
state_code:	状态码 <ul style="list-style-type: none"><li>● 200: 成功</li></ul>

5.9.2. 登录认证: login\_authentication

请求命令参数

```
{
  "cmd": "login_authentication",
  "id": "13245",
  "authentication": "12345789"
}
```

结果返回

```
{
  "cmd": "login_authentication",
  "error_msg": "Sucess",
  "id": "13245",
  "response": "Current security is no need to login authenticat.",
  "state_code": 200
}
```

注：认证失败，直接断开会话连接，需重新建立会话连接

字段名	说明
authentication	用户密码; 使用 HMAC-SHA1 算法对 用户密码进行加密，加密物料使用 signature， 再对加密后的数据进行 Base64 生成的字符串。 Signature 可通过 get_ems/reset_encrypt_key/change_encrypt_key/enable_encrypt/start_login 接口获取，需采用最新的 signature
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
response	这个字段在不需要密码登录的时候的返回提示,建议自动化不要适配此字段，这个 字段其他情况是没有的。

如

```
Std::string user_pass;
Std::string signature;
Std::string authentication;
unsigned char digest_buffer[20] = {0};
HmacSha1((const unsigned char *) user_pass.c_str(), user_pass.size(),
(const unsigned char *) signature.c_str(),signature.size(), digest_buffer);
Base64::EncodeFromArray(digest_buffer, 20, &authentication);
```

5.10.语音协议

语音协议主要分为两部分，配置和播放；  
注 意：此处的语音为相机设备自带的语音，非外接；

请求命令参数如下格式：

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
```

```
    "type": "%s"
    ...
  }
}
```

回复格式为:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200,
  "body": {
    "type": "%s"
    ...
  }
}
```

5.10.1. 获取当前语音文件列表: playserver\_json\_request

请求命令:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_get_voice_info",
    "voice_type": 3
  }
}
```

回复参数:

```
{
  "body": {
    "type": "ps_get_voice_info",
    "voice": [
      {
        "file_name": "MC53YXY="
      },
      {
        "file_name": "6bKBLndhdg=="
      },
      {
        "file_name": "6buRLndhdg=="
      }
    ],
    "voice_type": 4
  },
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
-----	----

type	获取语音文件 ps_get_voice_info
file_name	语音文件名称，base64 字符串；
voice_type	文件类型： 用户上传语音目录结构为:/home/admin/video/male, /home/admin/video/female 0. 未知类型 1. 用户语音男声(用户通过 ftp 上传至/home/admin 目录下.wav 文件) 2. 用户语音女声(用户通过 ftp 上传至/home/admin 目录下.wav 文件) 3. 系统语音男声 4. 系统语音女声

5.10.2. 设置语音默认参数：不支持

请求命令：-

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_set_voice_config",
    "voice_defalut_interval": 1,
    "voice_defalut_volume": 1,
    "voice_defalut_male": 1
  }
}
```

回复参数：-

```
{
  "body": {
    "type": "ps_set_voice_config"
  },
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
voice_defalut_interval	语音文件默认播放间隔，单位为毫秒；
voice_defalut_volume	语音默认音量，范围为[0, 100],现在还没有用，待开放；
voice_defalut_male	默认语音类型：0 男声；1 女声；
type	ps_set_voice_config 设置语音

5.10.3. 获取语音默认参数：不支持

请求命令：-

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_get_voice_config"
  }
}
```

回复结果：-

```
{
  "body": {
    "type": "ps_get_voice_config",
    "voice_defalut_interval": 300,
    "voice_defalut_male": 1,
    "voice_defalut_volume": 100
  },
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200
}
```

5.10.4. 播放语音参数: playserver\_json\_request

请求命令:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_voice_play",
    "voice": "agaasg",
    "voice_interval": 1,
    "voice_speed": 100,
    "voice_volume": 1,
    "voice_male": 1
  }
}
```

回复结果:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200,
  "body": {
    "type": "ps_voice_play"
  }
}
```

字段名	说明
voice	语音信息, utf-8/GBK 编码的 BASE64 编码字符串
voice_interval	语音文件播放间隔
voice_speed	语速 0~100
voice_volume	语音文件音量大小
voice_male	语音类型: 0 男声; 1 女声
type	ps_voice_play 语音播放

5.10.5. 请求语音对讲: start\_talk

注意: 设置语音对讲后, 由相机回复出来的消息, 是广播到所有的会话连接;  
此语音对讲命令只是控制命令, 具体的语音对讲信息通过双方协商的端口通信;  
请求命令参数:

```
{
  "cmd": "start_talk",
```



```

{id": "1234",
"body": {
  "window_size": 320
}
}

```

结果返回:

```

{
  "body": {
    "encode_type": 3,
    "port": 2707,
    "status": 1,
    "sampling_rate": 8000,
    "window_size": 1024,
    "sample_point": 512
  },
  "cmd": "start_talk",
  "error_msg": "Success",
  "id": "1234",
  "state_code": 200
}

```

请求字段	说明
cmd	消息类型
id	消息 ID, 可以忽略。
window_size	每次发送数据包的大小, 建议设置为 320.
响应字段	说明
cmd	消息类型
Id	消息 ID, 可以忽略。
state_code	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 其他: 失败</li> </ul>
port	语音对讲通信 TCP 服务端口
window_size	每次发送数据包的大小, 摄像机返回, 缓冲的字节数, 固定值。
sample_point	采样点, 固定值 512。
encode_type	支持的编码类型, 目前 RG 设备仅支持 G711A <ol style="list-style-type: none"> <li>1. PCM</li> <li>2. G711</li> <li>3. G711A</li> </ol>
sampling_rate	采样率固定值 8000(默认) 16000, 表示 8khz 16khz。
status	对讲状态, 固定值 0 <ul style="list-style-type: none"> <li>0 空闲</li> <li>1 忙碌</li> </ul>

备注:

1. 首先通过协议获取音频参数

"encode\_type":3,  
"port":2707,  
"sampling\_rate": 8000,  
"window\_size":1024,  
"sample\_point":512

encode\_type 音频编码格式: 1:PCM; 2:G711;3:G711A  
port 音频 tcp 服务端口号  
sampling\_rate 采样率  
window\_size 缓存区大小  
sample\_point 采样点个数

2. 建立 tcp 连接到相机音频服务端口, 收发音频数据

相机向 tcp 客户端发送的音频数据为: 4 字节数据长度+G711A 音频数据,总长度为 sample\_point +4  
tcp 客户端向相机发送的音频数据与上一致, 同样为: 4 字节数据长度+G711A 音频数据,总长度为 sample\_point +4

5.10.6. 刷新当前语音文件列表: playserver\_json\_request

请求命令:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_refresh_voice_info"
  }
}
```

回复参数:

```
{
  "body": {
    "type": "ps_refresh_voice_info"
  },
  "cmd": "playserver_json_request",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
type	刷新语音文件 ps_refresh_voice_info

5.10.7. 播放指定路径的音频: playserver\_json\_request

请求命令:

```
{
  "cmd": "playserver_json_request",
  "id": "132156",
  "body": {
    "type": "ps_voice_play_file",
    "voice_immediately": 0,
    "voice_volume": 100,
    "voice_interval": 2,
    "voice_file": [
      {
        "file_path":
        "L3RtcC9hcHAvZXhlYy9hdWRpby9mZW1hbGUv5LiA6Lev5bmz5a6JLndhdg=="
      },
      {
        "file_path":
        "L3RtcC9hcHAvZXhlYy9hdWRpby9mZW1hbGUv5LiA6Lev5bmz5a6JLndhdg=="
      }
    ]
  }
}
```

回复参数:

```
{
  "body": {
    "type": "ps_voice_play_file"
  },
  "cmd": "playserver_json_request",
  "error_msg": "Sucess",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
file_path	带有绝对路径的语音信息, utf-8/GBK 编码的 BASE64 编码字符串 /tmp/app/exec/audio 是自带语音目录, /home/admin/video 是客户定制语音目录
voice_immediately	1 0;1 立即播放, 把之前未播放音频的任务全部清除, 播放当前命令音频文件;0 排队播放。默认 0
voice_volume	播放音量, 取值范围[1-100], 没有该字段默认 100
voice_file	需要播放的音频文件数组-最多 20 条信息, 超过 20 条, 也只能播放 20 条
voice_interval	语音文件播放间隔

5.10.8. 播放 tts 文本内容: playserver\_json\_request

请求命令:

```
{
  "cmd": "playserver_json_request,
```

```

    "id": "132156",
    "body": {
      "type": "ps_voice_play",
      "voice": "suLK1GdiMjMxMg==",
      "voice_interval": 0,
      "voice_male": 1,
      "voice_volume": 100
    }
  }
}

```

回复参数:

```

{
  "body": {
    "type": "ps_voice_play"
  },
  "cmd": "playserver_json_request",
  "error_msg": "Sucess",
  "id": "132156",
  "state_code": 200
}

```

字段名	说明
voice	文本内容 GB2312 格式的 BASE64 编码字符串
voice_volume	播放音量, 取值范围[1-100], 没有该字段默认 100
voice_male	语音的声音类型, 1-男声, 0-女声
voice_interval	语音文件播放间隔

## 5.11.屏显协议:

注 意:

- 1.仅当 led\_enable 为 1 时, led\_content 中的内容才有效;
2. line\_content 数组大小与 led\_line\_num 保持一致; 最大不超过 4;
- 3.仅当 voice\_mode 不为 0 时, voice\_content 中的内容有效;
- 4.与网页配置互斥, 通过 TCP 配置后, 网页配置不再生效, 网页配置后, TCP 配置失效;
- 5.请注意此处的语音为外接 LED 屏显自带的语音;

### 5.11.1. 设置 LED 显示内容: set\_led\_show

请求命令参数

```

{
  "cmd": "set_led_show",
  "id": "12365",
  "body": {
    "led_enable": 1,
    "led_content": {
      "led_proto": 1,

```

```

    "led_status": 1,
    "led_refresh_time": 1,
    "led_line_num": 4,
    "line_content": [
      {
        "show_mode": 1,
        "show_content": "5qyi6L+O5YWJ5Li0"
      },
      {
        "show_mode": 1,
        "show_content": "5qyi6L+O5YWJ5Li0"
      },
      {
        "show_mode": 1,
        "show_content": "5qyi6L+O5YWJ5Li0"
      },
      {
        "show_mode": 1,
        "show_content": "5qyi6L+O5YWJ5Li0"
      }
    ],
    "voice_mode": 1,
    "voice_content": {
      "voice_volume": 2,
      "voice_welcom": 1,
      "voice_tag": 1,
      "play_content": "5qyi6L+O5YWJ5Li0"
    },
    "car_info": {
      "park_time": 32,
      "payment_amount": 9,
      "car_type": 1,
      "car_plate": "5qyi6L+O5YWJ5Li0"
    }
  }
}

```

#### 结果返回

```

{
  "cmd": "set_led_show",
  "id": "12365",
  "state_code": 200
}

```

### 5.11.2. 获取 LED 显示内容: get\_led\_show

#### 请求命令参数

```

{
  "cmd": "get_led_show",
  "id": "12365"
}

```

## 结果返回

```

{
  "cmd": "get_led_show",
  "id": "12365",
  "state_code": 200
  "body": {
    "led_enable": 1,
    "led_content": {
      "led_proto": 1,
      "led_status": 1,
      "led_refresh_time": 1,
      "led_line_num": 4,
      "line_content": [
        {
          "show_mode": 1,
          "show_content": "5qyi6L+O5YWJ5Li0"
        },
        {
          "show_mode": 1,
          "show_content": "5qyi6L+O5YWJ5Li0"
        },
        {
          "show_mode": 1,
          "show_content": "5qyi6L+O5YWJ5Li0"
        },
        {
          "show_mode": 1,
          "show_content": "5qyi6L+O5YWJ5Li0"
        }
      ]
    },
    "voice_mode": 1,
    "voice_content": {
      "voice_volume": 2,
      "voice_welcom": 1,
      "voice_tag": 1,
      "play_content": "5qyi6L+O5YWJ5Li0"
    },
    "car_info": {
      "park_time": 32,
      "payment_amount": 9,
      "car_type": 1,
      "car_plate": "5qyi6L+O5YWJ5Li0"
    }
  }
}

```

### 5.11.3. 获取 LED 数据传输使用串口号：get\_led\_serial\_port

请求命令参数

```
{
  "cmd": "get_led_serial_port",
  "id": "12365"
}
```

#### 结果返回

```
{
  "body": {
    "use_serial_port": 1
  },
  "cmd": "get_led_serial_port",
  "error_msg": "OK",
  "id": "12365",
  "state": 200,
  "state_code": 200
}
```

### 5.11.4. 设置 LED 数据传输使用串口号: set\_led\_serial\_port

#### 请求命令参数

```
{
  "cmd": "set_led_serial_port",
  "body": {
    "use_serial_port": 1
  },
  "id": 12365
}
```

#### 结果返回

```
{
  "cmd": "set_led_serial_port",
  "error_msg": "OK",
  "id": "12365",
  "state": 200,
  "state_code": 200
}
```

### 5.11.5. 字段信息以及相关枚举

字段名	类型	说明
voice_mode	int	语音播放播放模式: LED_VOICE_PLAY_MODE
voice_content	无	voice_mode 不为 0 时有效
voice_volume	int	音量大小 (0 - 9)
voice_welcom	int	欢迎语: LED_VOICE_USER_DEFINE
voice_tag	int	结束语: LED_VOICE_USER_DEFINE
led_enable	int	使能屏显 (0:去使能, 1:使能)

led_content	无	led_enable 为 1 时，内容有效
led_line_num	int	屏幕支持行数(目前最大支持 4 行)
led_proto	int	屏显协议：参考枚举定义：RS485_CTRL_PRO
led_status	int	屏显状态，用户自保存
led_refresh_time	int	屏显内容定时刷新时间[6,15]
line_content	无	屏显每行具体内容,数组大小与 led_line_num 值一致
show_mode	int	屏显显示模式，参考枚举：LED_SCREEN_SHOW_MODE
show_content	string	用户自定义显示（utf-8 的字符串经过 64 位编码之后的内容）
play_content	string	用户自定义播放（utf-8 的字符串经过 64 位编码之后的内容）仅支持 32 省份简称+大写字母+数字+军+警+学+十+百+千+万+年+月+日
use_serial_port	int	使用的串口号（1,2，部分设备支持 3）
state	int	回复状态；200：成功，400：格式错误，500 服务器错误
state_code	int	回复状态；200：成功，400：格式错误，500 服务器错误
car_info	无	用户需要显示车辆信息时，有效
park_time	int	停车时间（分钟）
payment_amount	int	收费金额，元
car_type	int	车辆类型：LED_VOICE_CAR_TYPE
car_plate	string	车牌号：UTF-8 经过 64 位编码

相关枚举定义：

### 1. 屏显协议：RS485\_CTRL\_PRO

```
enum RS485_CTRL_PRO {
    ZS_SERIAL = 0x00,      /* A 臻识科技识别结果输出协议 */
    YB_BX5K1 = 0x01,      /* B 仰邦科技 BX5K1 控制卡协议 */
    XQ_EQ2013 = 0x02,     /* C 谐阔 EQ 系列 2013 控制卡协议 */
    KF_LED = 0x03,        /* D 科发 LED 控制卡协议 */
    FK_LED = 0x04,        /* E 方控智能 LED 控制卡协议 */
};
```

### 2. 屏显模式：LED\_SCREEN\_SHOW\_MODE

```
enum LED_SCREEN_SHOW_MODE {
    LED_SHOW_MODE_NONE = 0x00,      // 不显示
    LED_SHOW_MODE_USER_DEFINE = 0x01, // 显示用户自定义内容
    LED_SHOW_MODE_SYS_TIME = 0x02,   // 显示时间
    LED_SHOW_MODE_FREE_PARKING = 0x04, // 显示空余车位
    LED_SHOW_MODE_CAR_PLATE = 0x08,   // 显示车牌号
    LED_SHOW_MODE_CAR_TYPE = 0x10,    // 显示车类型
    LED_SHOW_MODE_PARK_TIME = 0x20,   // 显示停车时间
    LED_SHOW_MODE_CHARGE_MONEY = 0x40, // 显示收费金额
    LED_SHOW_MODE_MAX,
};
```



**3. 语音播放模式: LED\_VOICE\_PLAY\_MODE**

```
enum LED_VOICE_PLAY_MODE {
LED_VOIC_PLAY_NONE = 0x00,           // 不播放语音
LED_VOICE_PLAY_WELCOM = 0x01,        // 欢迎语
LED_VOICE_PLAY_CAR_TYPE = 0x02,      // 车辆类型
LED_VOICE_PLAY_CAR_PLATE = 0x04,     // 车牌号
LED_VOICE_PLAY_PARK_TIME = 0x08,     // 停车时间
LED_VOICE_PLAY_CHARGE_MONEY = 0x10,  // 收费金额
LED_VOICE_PLAY_TAG = 0x20,           // 结束语
LED_VOICE_PLAY_USER_DEFINE = 0x40,   // 自定义语音
LED_VOICE_PLAY_MAX,
};
```

**4. 播放语音的车辆类型: LED\_VOICE\_CAR\_TYPE**

```
enum LED_VOICE_CAR_TYPE{
CAR_TYPE_MONTH_RENT = 1,             //月租车
CAR_TYPE_TEMP_CAR = 2,               // 临时车
CAR_TYPE_NO_PLATE = 3,               // 无牌车
CAR_TYPE_BLACKLIST = 4,              // 黑名单
CAR_TYPE_MONTH_RENT_EXPIRE = 7,     // 月租车到期
CAR_TYPE_SPECIAL_CAR = 9,           // 特殊车
CAR_TYPE_MAX,
};
```

**5. 播放的欢迎语、结束语**

```
enum LED_VOICE_USER_DEFINE {
USER_DEFINE_VOICE_NONE = 0,          //
USER_DEFINE_VOICE_WELCOM_1,          // 您好
USER_DEFINE_VOICE_WELCOM_2,          // 欢迎光临
USER_DEFINE_VOICE_WELCOM_3,          // 欢迎回家
USER_DEFINE_VOICE_TAG_1,              // 一路顺风
USER_DEFINE_VOICE_TAG_2,              // 一路平安
USER_DEFINE_VOICE_TAG_3,              // 欢迎下次光临
USER_DEFINE_VOICE_MAX,
};
```

**5.12. LCD 配置:****5.12.1. 设置 LCD 广告音量**

声音设置把一天 24h 分为三个时间段,每个时间段都可以独立设置音量,起盈利大小分为 0-5

档：0 静音 1-5 档音量后台对应的值是档数乘以 20

请求命令参数

```
{
  "cmd": "set_ad_voice",
  "id": "123",
  "body": {
    "media_voice_time_ctrl": [
      {
        "time_begin": "00:00:00",
        "time_end": "08:00:00",
        "voice_level": 1
      },
      {
        "time_begin": "08:00:00",
        "time_end": "17:00:00",
        "voice_level": 0
      },
      {
        "time_begin": "17:00:00",
        "time_end": "24:00:00",
        "voice_level": 5
      }
    ]
  }
}
```

结果返回

```
{
  "cmd": "set_ad_voice",
  "error_msg": "Success",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
id	命令 id
media_voice_time_ctrl	obj 对象有
time_begin	一段时间的起始时间 时：分：秒 格式
time_end	一段时间的结束时间 时：分：秒 格式
voice_level	声音大小 0：静音 1 2 3 4 5 对应声音档数

5.12.2. 查询 LCD 广告音量

字段说明参考设置 LCD 广告音量

请求命令

```
{
```

```

"cmd": "get_ad_voice",
"id": "123"
}

```

**响应命令**

```

{
  "cmd": "get_ad_voice",
  "id": "123",
  "body": {
    "media_voice_time_ctrl": [
      {
        "time_begin": "00:00:00",
        "time_end": "08:00:00",
        "voice_level": 1
      },
      {
        "time_begin": "08:00:00",
        "time_end": "17:00:00",
        "voice_level": 0
      },
      {
        "time_begin": "17:00:00",
        "time_end": "24:00:00",
        "voice_level": 5
      }
    ]
  },
  "state_code": 200,
  "error_msg": "Success"
}

```

**5.12.3. 设置 LCD 背光亮度**

背光亮度分为智能调节模式，手动调节模式，关闭屏显模式

智能调节模式：根据环境光来调节屏幕亮度，与时间无关

手动调节模式：根据时间段配置的亮度来设置屏幕亮度

关闭屏显模式：背光全部关闭，黑屏

**请求参数命令**

```

{
  "cmd": "set_lcd_bright",
  "id": "123",
  "body": {
    "lcd_bright_time_ctrl": [
      {
        "time_begin": "00:00:00",
        "time_end": "08:00:00",
        "level": 2
      }
    ],
  },
}

```

```
{
  {
    "time_begin": "08:00:00",
    "time_end": "17:00:00",
    "level": 5
  },
  {
    "time_begin": "17:00:00",
    "time_end": "24:00:00",
    "level": 2
  }
],
"lcd_bright_mode": 0
}
```

命令响应

```
{
  "cmd": "set_lcd_bright",
  "error_msg": "Success",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
id	命令 id
lcd_bright_time_ctrl	obj 对象有
time_begin	一段时间的起始时间 时：分：秒 格式
time_end	一段时间的结束时间 时：分：秒 格式
level	Lcd 背光亮度等级 0：不亮 1 2 3 4 5 对应背光档数
lcd_bright_mode	Lcd 亮度调节模式 0：智能 1：手动-与时间段相关 2 关闭屏显

5.12.4. 查询 LCD 背光参数

参数字段参考设置 LCD 背光亮度参数

请求命令

```
{
  "cmd": "get_lcd_bright",
  "id": "123"
}
```

命令响应

```
{
  "cmd": "get_lcd_bright",
  "id": "123",
  "body": {
    "lcd_bright_time_ctrl": [
      {
```

```

        "time_begin": "00:00:00",
        "time_end": "08:00:00",
        "level": 2
    },
    {
        "time_begin": "08:00:00",
        "time_end": "17:00:00",
        "level": 5
    },
    {
        "time_begin": "17:00:00",
        "time_end": "24:00:00",
        "level": 2
    }
],
"lcd_bright_mode": 0
},
"state_code": 200,
"error_msg": "Success"
}

```

### 5.12.5. 设置 LCD 忙时半屏显示

忙时文字只能显示在下半屏，上半屏显示二维码。场景有全屏，半屏。忙时和闲时。两两组合：全屏忙时(full)，全屏闲时(full\_ad)，半屏忙时(busy\_half)，半屏闲时(free\_half)；后面 4 条都是在广告在页面配置好的条件下进行控制

#### 设置参数命令

```

{
  "body": {
    "scene_name": "YnVzeV9oYWxm",
    "parking_fee": "5pS26LS5MjDIhYM=",
    "parking_time": "5YGc6L2m5pe26Ze0ICAzMOWIhumSnw==",
    "charge_state": "5pS26LS55oiQ5Yqf",
    "qrcode_text": "https://www.vzicar.com",
    "custom": [
      "5Ymp5L2Z6L2m5L2NOiAzMA==",
      "5Ymp5L2Z6L2m5L2NOiAzMA==",
      "5Ymp5L2Z6L2m5L2NOiAzMA=="
    ]
  },
  "cmd": "ad_push_message",
  "id": "132156"
}

```

#### 命令响应

```

{
  "cmd": "ad_push_message",
  "error_msg": "Success",
  "id": "132156",
  "state_code": 200
}

```

}

字段名	说明
id	命令 id
scene_name	要控制的场景名称 base64 编码, YnVzeV9oYWxm(busy_half)
parking_fee	收费金额 5pS26LS5MjDIhYM=(收费 20 元)
parking_time	停车时长 5YGc6L2m5pe26Ze0ICAzMOWIhumSnw==(停车时间 30 分钟)
charge_state	收费状态 5pS26LS55oiQ5Yqf(收费成功)
qrcode_text	动态二维码内容
custom	自定义文字 utf-8 base64,数组跟网页配置一一对应。比如配置三个就显示前三行

### 5.12.6. 设置 LCD 闲时下半屏显示

#### 设置参数

```
{
  "body": {
    "scene_name": "ZnJlZV9oYWxm",
    "parking_space_left": "5Ymp5L2Z6L2m5L2NOiAzMA=="
  },
  "cmd": "ad_push_message",
  "id": "132156"
}
```

#### 响应

```
{
  "cmd": "ad_push_message",
  "error_msg": "Success",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
id	命令 id
scene_name	要控制的场景名称 base64 编码, ZnJlZV9oYWxm(free_half)
parking_space_left	剩余车位 utf-8 base64 编码 5Ymp5L2Z6L2m5L2NOiAzMA==(剩余车位: 30)

### 5.12.7. 设置 LCD 全屏忙时显示

#### 请求命令

```
{
  "body": {
    "scene_name": "ZnVsbA==",
    "parking_space_left": "5Ymp5L2Z6L2m5L2NOiAzMA==",
    "qrcode_text": "https://www.vzicar.com"
  },
  "cmd": "ad_push_message",
}
```

```
{
  "id": "132156"
}
```

**响应命令**

```
{
  "cmd": "ad_push_message",
  "error_msg": "Success",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
id	命令 id
scene_name	要控制的场景名称 base64 编码, ZnVsbA== (full)
parking_space_left	剩余车位 utf-8 base64 编码 5Ymp5L2Z6L2m5L2N0iAzMA==(剩余车位: 30)
qrcode_text	动态二维码内容

**5.12.8. 设置 LCD 全屏广告显示****请求命令**

```
{
  "body": {
    "scene_name": "ZnVsbF9hZA=="
  },
  "cmd": "ad_push_message",
  "id": "132156"
}
```

**响应**

```
{
  "cmd": "ad_push_message",
  "error_msg": "Success",
  "id": "132156",
  "state_code": 200
}
```

字段名	说明
id	命令 id
scene_name	要控制的场景名称 base64 编码, ZnVsbF9hZA== (full_ad)

**5.12.9. 自定义屏显说明**

LCD 屏配置的区域范围为宽 360\*高 640, 不同分辨率的屏会按比例进行换算。

"scene\_info"数组中的"scene\_type"为"ad"的元素, 由于元素中可能有视频资源, 需要单独存放, 不能和图片放在一起。

以下配置包含两个场景, 可以根据场景名称进行切换。具体字段说明如下:

```

{
  "ad_source": "local",          // 必填, 广告源类型: local-本地/none-无/remote-远程
  "ad_mode": 3,                  // 广告模式: 1-全屏/2-半屏/3-自定义模式
  "ad_group": {                  // 广告组配置容器
    "group_id": 0,               // 必填, 广告组唯一标识 (范围 0-255)
    "scenes": [                 // 场景数组 (至少包含 1 个场景)
      {
        "scene_name": "YWQ=",    // Base64 解码为"ad", 场景名称标识,可发 tcp 协议根据名称
        切换场景
        "scene_id": 0,           // 场景唯一 ID (范围 0-255)
        "scene_info": [         // 复合场景分层配置 (当前场景分上下两部分)
          { // ----- 上层信息区 (420px 高度) -----
            "background_type": 1, // 背景类型: 0-颜色/1-图片/2-无
            "background_color": "", // 背景颜色 (background_type=0 时生效)
            "background_image": "1.png", // 背景图片路径 (需确保文件存在)
            "scene_type": "",      // 场景类型: ad/car_in/car_out (当前为空需补全)
            "scene_pos": {        // 区域坐标 (基于 360x640 基准分辨率)
              "x": 0,             // 左上角 X 坐标 (像素)
              "y": 0,             // 左上角 Y 坐标 (像素)
              "width": 360,       // 区域宽度 (像素)
              "height": 420      // 区域高度 (像素)
            },
            "elem_list": [        // 元素列表 (按渲染顺序排列)
              { // 元素 1: 时间显示
                "elem_enable": 1, // 元素启用开关: 1-启用/0-禁用
                "elem_type": "dynamic_text", // 元素类型: dynamic_text
                "elem_sub_type": "datetime", // 子类型: datetime-实时时间
                "elem_pos": {      // 元素坐标 (相对父容器)
                  "x": 70,
                  "y": 10,
                  "width": 290,
                  "height": 20
                },
                "elem_bg_color": "rgba(255,255,255,0)", // 背景透明
                "elem_fg_color": "#FF0000", // 文字颜色: 红色
                "elem_font_family": 0, // 字体 ID: 0-思源黑体
                "elem_font_bold": 0, // 字体加粗: 0-常规
                "elem_font_size": 16, // 字号: 16px
                "elem_align": "center", // 文本对齐方式
                "week_enable": 0 // 星期显示开关: 0-隐藏
              },
              { // 元素 2: LOGO 图片
                "elem_enable": 1,
                "elem_type": "static_pic", // 静态图片元素
                "elem_pos": {

```



```

        "x": 10,
        "y": 10,
        "width": 60,
        "height": 40
    },
    "elem_res_list": [{          // 资源列表 (可多个)
        "path": "logo.png"      // 图片路径 (需校验文件存在)
    }]
},
{ // 元素 3: 车牌号显示
    "elem_enable": 1,
    "elem_bg_color": "rgba(255,255,255,0)", // 透明背景
    "elem_type": "dynamic_text",
    "elem_sub_type": "car_id",          // 车牌号动态数据
    "elem_pos": {
        "x": 0,
        "y": 230,
        "width": 360,
        "height": 24
    },
    "elem_fg_color": "#FFFFFF",        // 白色文字
    "elem_font_family": 0,
    "elem_font_bold": 1,               // 加粗显示
    "elem_font_size": 24,
    "elem_align": "center"
},
{ // 元素 4: 自定义文本 1
    "elem_enable": 1,
    "elem_type": "dynamic_text",
    "elem_sub_type": "custom",         // 需外部传入文本
    "elem_pos": {
        "x": 0,
        "y": 260,
        "width": 360,
        "height": 64
    },
    "elem_bg_color": "rgba(255,255,255,0)",
    "elem_fg_color": "#FFFFFF",
    "elem_font_family": 0,
    "elem_font_bold": 0,
    "elem_font_size": 64,              // 超大字号需校验屏幕支持
    "elem_align": "center"
},
{ // 元素 5: 自定义文本 2
    "elem_enable": 1,

```

```

        "elem_type": "dynamic_text",
        "elem_sub_type": "custom",
        "elem_pos": {
            "x": 0,
            "y": 330,
            "width": 360,
            "height": 64
        },
        "elem_bg_color": "rgba(255,255,255,0)",
        "elem_fg_color": "#FFFFFF",
        "elem_font_family": 0,
        "elem_font_bold": 0,
        "elem_font_size": 64,
        "elem_align": "center"
    }
]
},
{ // ----- 下层广告区 (220px 高度) -----
    "background_type": 2, // 无背景
    "background_color": "",
    "background_image": "",
    "scene_type": "ad", // 广告类型场景
    "scene_pos": {
        "x": 0,
        "y": 420,
        "width": 360,
        "height": 220
    },
    "elem_list": [{
        "elem_enable": 1,
        "elem_name": "ad", // 元素名称 (用于调试)
        "elem_type": "ad", // 广告容器元素
        "elem_pos": {
            "x": 0,
            "y": 420,
            "width": 360,
            "height": 220
        }
    }],
    "elem_res_list": [ // 广告资源池 (轮播)
        {
            "path": "22.mp4", // 视频路径
            "elem_type": "video", // 视频类型
            "video_play_count": 1 // 播放次数: 1 次
        },
        {

```

```

        "path": "1.jpg",          // 图片路径
        "picture_max_duration": 10, // 显示时长 (秒)
        "elem_type": "static_pic" // 图片类型
    }
}
]]
}
]
},
{
    "scene_name": "ZnVsbA==",      // Base64 解码为"full", 场景名称标识
    "scene_id": 1,                  // 场景 ID
    "scene_info": [                 // 分上下两部分
        { // ----- 上层广告区 (240px 高度) -----
            "background_type": 2,    // 无背景
            "background_color": "",
            "background_image": "",
            "scene_type": "ad",      // 广告场景
            "scene_pos": {
                "x": 0,
                "y": 0,
                "width": 360,
                "height": 240
            },
            "elem_list": [{
                "elem_id": 0,        // 元素唯一 ID (范围 0-255)
                "elem_type": "ad",   // 广告容器
                "elem_pos": {
                    "x": 0,
                    "y": 0,
                    "width": 360,
                    "height": 240
                },
                "elem_res_list": [    // 广告资源池
                    {
                        "path": "33.mp4",    // 视频广告
                        "elem_type": "video",
                        "video_play_count": 1
                    },
                    {
                        "path": "green.png",  // 静态图片
                        "picture_max_duration": 10,
                        "elem_type": "static_pic"
                    }
                ]
            }
        ]
    }
}

```

```

    }
  },
  { // ----- 下层信息区 (400px 高度) -----
    "background_type": 1,          // 背景图片
    "background_color": "",
    "background_image": "2.png",   // 背景图路径
    "scene_type": "",              // 需补全场景类型
    "scene_pos": {
      "x": 0,
      "y": 240,
      "width": 360,
      "height": 400
    },
    "elem_list": [
      { // 元素 1: 时间+星期
        "elem_enable": 1,
        "elem_type": "dynamic_text",
        "elem_sub_type": "datetime", // 时间类型
        "elem_pos": {
          "x": 0,
          "y": 10,
          "width": 360,
          "height": 20
        },
        "elem_bg_color": "rgba(255,255,255,0)",
        "elem_fg_color": "#FFFFFF", // 白色文字
        "elem_font_family": 0,
        "elem_font_bold": 0,
        "elem_font_size": 56,        // 特大字号 (需校验)
        "elem_align": "center",
        "week_enable": 1             // 显示星期
      },
      { // 元素 2: 车牌号
        "elem_enable": 1,
        "elem_type": "dynamic_text",
        "elem_sub_type": "car_id",   // 动态车牌
        "elem_pos": {
          "x": 0,
          "y": 160,
          "width": 360,
          "height": 32
        },
        "elem_bg_color": "rgba(255,255,255,0)",
        "elem_fg_color": "#FFFFFF",
        "elem_font_family": 0,

```



```
{
  "cmd": "set_cloud_ctrl",
  "error_msg": "Success",
  "id": "123",
  "state_code": 200
}
```

字段名	说明
id	命令 id
dir	<ul style="list-style-type: none"> <li>● 0:设置超时时间</li> <li>● 1: 保持继续移动</li> <li>● 2: 向上移动</li> <li>● 3: 保持向上移动</li> <li>● 4: 向下移动</li> <li>● 5: 保持向下移动</li> <li>● 8: 停止上下移动</li> <li>● 16: 向左移动</li> <li>● 17: 保持向左移动</li> <li>● 32: 向右移动</li> <li>● 33: 保持向右移动</li> <li>● 64: 停止左右移动</li> </ul>
Value	type=0 的时候, 此字段为设置超时时间, 单位毫秒, 其他情况时这个字段可以不填或者填-1。

## 5.14.设置用户自定义 OSD (视频)

请求命令参数

```
{
  "cmd": "set_osd_para",
  "body": {
    "osd_type": 1,
    "user_osd": {
      "user_osd_param": [
        {
          "id": 0,
          "display": 1,
          "color": 0,
          "front_size": 3,
          "text": "56ys5LiA6KGM"
        },
        {
          "id": 1,
          "display": 1,
          "color": 1,
          "front_size": 3,
          "text": "c2Vjb25klGxpbmU="
        },
        {
          "id": 2,
          "display": 1,
          "color": 2,
          "front_size": 3,
          "text": "dGJkuInooYw="
        },
        {
          "id": 3,
          "display": 1,
          "color": 3,
          "front_size": 3,
          "text": "56ys5Zub6KGMT1NEIFRDUCA="
        }
      ],
      "x_pos": 10,
      "y_pos": 10
    }
  }
}
```

结果返回

```
{
  "cmd": "set_osd_para",
  "error_msg": "Sucess",
  "id": "999999",
  "state_code": 200
}
```

参数解释

字段名	说明
id	自定义 OSD 行数，取值 0 到 3，id 为 0 表示第一行，3 表示第四行
display	是否显示，1：显示文本      0：不显示文本

color	文本颜色，取值0到3    0：白色   1：红色    2：蓝色    3：绿色
front_size	文本字体大小。取值0到3    0：最小字号        3：最大字号
text	想要显示的文本经过b a s e 6 4编码后的内容
x_pos	横坐标百分比，取值范围0到1 0 0    例如取值为0，最左边显示
y_pos	纵坐标百分比，取值范围0到1 0 0    例如取值为0，最上边显示
osd_type	0：动态叠加自定义 osd，叠加内容不存 flash 1：静态叠加自定义 osd，叠加内容存 flash

5.15.图片 osd：仅 E5 支持

5.15.1.        设置图片 osd: AVS\_SET\_PIC\_OSD\_PRM

5.15.1.1.    E5 1.0 协议

请求

```
{
  "module": "AVS_PIC_OSD_MODULE",
  "type": "AVS_SET_PIC_OSD_PRM",
  "cmd": "AVS_SET_PIC_OSD_PRM",
  "is_dynamic": 0,
  "body": {
    "pos": 0,
    "font": {
      "color": 0,
      "size": 0
    },
    "separation": 0,
    "content": [
      {
        "id": 0,
        "enable": 0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson": {
          "type": 0 // 0: YYYY/MM/DD ,1: MM/DD/YYYY, 2: DD/MM/YYYY, 3:
          YYYY-MM-DD,4:MM-DD-YYYY,5:DD-MM-YYYY
        }
      },
      {
        "id": 1,
        "enable": 0,
        "x_pos": 0, //横坐标，仅当叠加位置为图片内有效
        "y_pos": 0,
        "configJson": {
          "type": 0 // 0: YYYY/MM/DD ,1: MM/DD/YYYY, 2: DD/MM/YYYY, 3:
          YYYY-MM-DD,4:MM-DD-YYYY,5:DD-MM-YYYY
        }
      },
      {
        "id": 2,
```



```

        "enable":0,
        "configJson":{
            "type":0, // 0: 12hours, 1: 24hours
            "ms_enable":1
        }
    },
    {
        "id":3,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{
            "text":"T1NEIFRleHQ="
        }
    },
    {
        "id":4,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{

        }
    },
    {
        "id":5,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{

        }
    },
    {
        "id":6,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{

        }
    },
    {
        "id":7,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{

        }
    },
    {
        "id":8,
        "enable":0,

```

```

        "x_pos": 0,
        "y_pos": 0,
        "configJson":{
            "text":"T1NEIFRleHQ="
        }
    },
    {
        "id":9,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{
            "text":"T1NEIFRleHQ="
        }
    },
    {
        "id":10,
        "enable":0,
        "x_pos": 0,
        "y_pos": 0,
        "configJson":{
            "text":"T1NEIFRleHQ="
        }
    }
],
"content_order":[0,1,2,3,4,5,6,7,8,9,10]
}
}

```

层级	参数名称	参数类型	长度	是否必须	描述
1.1	module	string		是	模块名字
1.2	cmd	string		是	控制命令
1.3	type	string		是	消息类型
1.4	is_dynamic	int		否	0: 静态图片 OSD 信息叠加，写入 flash; 1: 动态平台 OSD 信息叠加，不写 flash; 无此字段时默认为静态 OSD 叠加
1.5	body	object		是	消息体
1.5.1	pos	int		是	叠加位置，0: 图片外上，1: 图片外下，2: 图片内，3: 关闭
1.5.2	font	object		是	字体
1.5.2.1	color	int		是	字体颜色，0: 白色，1: 红色，2: 蓝色，3: 绿色
1.5.2.2	size	int		是	字体大小，0: 16，1: 24，2: 32，3: 48
1.5.3	separation	int		是	0: 空格；1: 竖线 (扩张区域显示时生效)

1.5.4	content	array		是	
1.5.4.1	id	int		是	0: 实时结果, 1: 日期, 2: 时间, 3: 地点, 4: 车辆事件 (部分支持), 5: 车辆类型, 6: 车标车款, 7: 车身颜色, 8: 自定义内容 1, 9: 自定义内容 2, 10: 自定义内容 3
1.5.4.2	x_pos	int		是	水平位置, 0~100
1.5.4.3	y_pos	int		是	垂直位置, 0~100
1.5.4.4	configJson	object		是	
1.5.4.5	type	int		否	与 id 值有关, 见示例
1.5.4.6	text	string		否	文本的 base64 编码, 编码前最多 30 个汉字或者 60 个英文
1.5.4.7	ms_enable	int		否	精确到毫秒
1.5.5	content_order	array		是	根据 id 排序 (扩张区域显示时生效)

#### 响应

```
{
  "cmd": "AVS_SET_PIC_OSD_PRM",
  "error_msg": "Sucess",
  "id": "999999",
  "state_code": 200
}
```

### 5.15.1.2. 支持图片 OSD 的 DV500 平台协议

#### 请求

```
{
  "type": "AVS_SET_PIC_OSD_PRM",
  "module": "AVS_PIC_OSD_MODULE",
  "cmd": "AVS_SET_PIC_OSD_PRM",
  "is_dynamic": 1,
  "body": {
    "content": [
      {
        "configJson": {
          "enable": 0
        },
        "configJsons": [],

```

```
"enable": 1,
"id": 0,
"multi_line": 0,
"x_pos": 0,
"y_pos": 0
},
{
  "configJson": {
    "enable": 0,
    "type": 0
  },
  "configJsons": [
    {
      "enable": 1,
      "ms_enable": 1,
      "text": "",
      "type": 0
    },
    {
      "enable": 1,
      "ms_enable": 1,
      "text": "",
      "type": 0
    },
    {
      "enable": 1,
      "ms_enable": 1,
      "text": "",
      "type": 0
    }
  ],
  "enable": 1,
  "id": 1,
  "multi_line": 1,
  "x_pos": 5,
  "y_pos": 5
},
{
  "configJson": {
    "enable": 0,
    "ms_enable": 0,
    "type": 0
  },
  "configJsons": [],
  "enable": 1,
  "id": 2,
  "multi_line": 0,
  "x_pos": 10,
  "y_pos": 10
},
{
  "configJson": {
    "enable": 0,
    "text": "T1NEIFRleHQ="
```

```
    },
    "configJsons": [],
    "enable": 1,
    "id": 3,
    "multi_line": 0,
    "x_pos": 15,
    "y_pos": 15
  },
  {
    "configJson": {
      "enable": 0
    },
    "configJsons": [],
    "enable": 1,
    "id": 4,
    "multi_line": 0,
    "x_pos": 20,
    "y_pos": 20
  },
  {
    "configJson": {
      "enable": 0
    },
    "configJsons": [],
    "enable": 1,
    "id": 5,
    "multi_line": 0,
    "x_pos": 25,
    "y_pos": 25
  },
  {
    "configJson": {
      "enable": 0
    },
    "configJsons": [],
    "enable": 1,
    "id": 6,
    "multi_line": 0,
    "x_pos": 30,
    "y_pos": 30
  },
  {
    "configJson": {
      "enable": 0
    },
    "configJsons": [],
    "enable": 1,
    "id": 7,
    "multi_line": 0,
    "x_pos": 35,
    "y_pos": 35
  },
  {
    "configJson": {
```

```
"enable": 0,
"text": "T1NEIFRleHQ="
},
"configJsons": [
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "MQ==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "Mg==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "Mw==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "NA==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "NQ==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "Ng==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "Nw==",
    "type": 0
  },
  {
    "enable": 1,
    "ms_enable": 0,
    "text": "OA==",
    "type": 0
  }
],
"enable": 1,
```

```

    "id": 8,
    "multi_line": 1,
    "x_pos": 40,
    "y_pos": 40
  },
  {
    "configJson": {
      "enable": 0,
      "text": "T1NEIFRleHQ="
    },
    "configJsons": [],
    "enable": 0,
    "id": 9,
    "multi_line": 0,
    "x_pos": 45,
    "y_pos": 45
  },
  {
    "configJson": {
      "enable": 0,
      "text": "T1NEIFRleHQ="
    },
    "configJsons": [],
    "enable": 0,
    "id": 10,
    "multi_line": 0,
    "x_pos": 50,
    "y_pos": 50
  }
],
"content_order": [
  0,
  1,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10
],
"font": {
  "color": 0,
  "size": 2
},
"pos": 1,
"separation": 0
}
}
```

层级	参数名称	参数类型	长度	是否必须	描述
----	------	------	----	------	----

1.1	module	string		是	模块名字
1.2	cmd	string		是	控制命令
1.3	type	string		是	消息类型
1.4	is_dynamic	int		否	0: 静态图片 OSD 信息叠加, 写入 flash; 1: 动态平台 OSD 信息叠加, 不写 flash; 无此字段时默认为静态 OSD 叠加
1.5	body	object		是	消息体
1.5.1	pos	int		是	叠加位置, 0: 图片外上, 1: 图片外下, 2: 图片内, 3: 关闭
1.5.2	font	object		是	字体
1.5.2.1	color	int		是	字体颜色, 0: 白色, 1: 红色, 2: 蓝色, 3: 绿色
1.5.2.2	size	int		是	字体大小, 0: 16, 1: 24, 2: 32, 3: 48
1.5.3	separation	int		是	0: 空格; 1: 竖线 (扩张区域显示时生效)
1.5.4	content	array		是	
1.5.4.1	id	int		是	0: 实时结果, 1: 日期, 2: 时间, 3: 地点, 4: 车辆事件 (部分支持), 5: 车辆类型, 6: 车标车款, 7: 车身颜色, 8: 自定义内容 1, 9: 自定义内容 2, 10: 自定义内容 3
1.5.4.2	x_pos	int		是	水平位置, 0~100
1.5.4.3	y_pos	int		是	垂直位置, 0~100
1.5.4.4	configJson	object		是	
1.5.4.5	type	int		否	与 id 值有关, 见示例
1.5.4.6	text	string		否	文本的 base64 编码, 编码前最多 30 个汉字或者 60 个英文
1.5.4.7	ms_enable	int		否	精确到毫秒
1.5.4.8	configJsons	array		否	如果 id 为 1, 按顺序, 分别为日期, 时间, 星期设置
1.5.5	content_order	array		是	根据 id 排序 (扩张区域显示时生效)

## 响应

{



```

"cmd" : "AVS_SET_PIC_OSD_PRM",
"error_msg" : "Sucess",
"id" : "999999",
"state" : 200,
"state_code" : 200
}

```

### 5.15.2. 获取图片 osd 配置: AVS\_GET\_PIC\_OSD\_PRM

请求

```

{
  "module": "AVS_PIC_OSD_MODULE",
  "type": "AVS_GET_PIC_OSD_PRM",
  "cmd": "AVS_GET_PIC_OSD_PRM",
  "body": {}
}

```

响应

字段含义参考设置图片 osd 中的说明: (动态图片 OSD 配置不支持获取操作)

```

{
  "body": {
    "content": [
      {
        "configJson": {},
        "enable": 0,
        "id": 0,
        "x_pos": 0,
        "y_pos": 0
      },
      {
        "configJson": {
          "type": 0
        },
        "enable": 0,
        "id": 1,
        "x_pos": 0,
        "y_pos": 0
      },
      {
        "configJson": {
          "ms_enable": 1,
          "type": 0
        },
        "enable": 0,
        "id": 2,
        "x_pos": 0,
        "y_pos": 0
      }
    ],
    "configJson": {

```

```
        "text" : "T1NEIFRleHQ="
    },
    "enable" : 0,
    "id" : 3,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {},
    "enable" : 0,
    "id" : 4,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {},
    "enable" : 0,
    "id" : 5,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {},
    "enable" : 0,
    "id" : 6,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {},
    "enable" : 0,
    "id" : 7,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {
        "text" : "T1NEIFRleHQ="
    },
    "enable" : 0,
    "id" : 8,
    "x_pos" : 0,
    "y_pos" : 0
},
{
    "configJson" : {
        "text" : "T1NEIFRleHQ="
    },
    "enable" : 0,
    "id" : 9,
    "x_pos" : 0,
    "y_pos" : 0
},
{
```

```
        "configJson" : {
            "text" : "T1NEIFRleHQ=",
        },
        "enable" : 0,
        "id" : 10,
        "x_pos" : 0,
        "y_pos" : 0
    }
],
"content_order" : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ],
"font" : {
    "color" : 0,
    "size" : 0
},
"pos" : 0,
"separation" : 0
},
"cmd" : "AVS_GET_PIC_OSD_PRM",
"error_msg" : "Sucess",
"id" : "999999",
"state_code" : 200
}
```

5.16.用户自定义设备信息（R3 不支持）

用户自定义设备信息设置和获取。

5.16.1. 获取自定义信息: get\_custom\_user\_info

请求命令参数

```
{
    "cmd": "get_custom_user_info",
    "id": "123456"
}
```

结果返回

```
{
    "cmd": " get_custom_user_info",
    "body": {
        "manufacturer": "SJ-1",
        "data": { "ver": 1, "is_zoom": 0, "class": 1
    }
    },
    "id": "123456",
    "err_msg": "OK",
    "state_code": 200
}
```

字段名	说明
-----	----

manufacturer	字符串标识的协议适配厂家 如果没有配置或厂家不在支持列表返回 "unknown"
data	自定义信息（不同厂家结构不同，以 SJ-1 为例） <b>ver</b> 版本，整数 0~3，0 表示没有设置。 <b>is_zoom</b> 是否支持变焦，整数 0~2，1 支持，2 不支持，0 表示没有设置。 <b>class</b> 设备等级（或分类），整数 0~15，0 表示没有设置。 注：值为 0（没有设置）的项不会通过广播发送。
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
err_msg	错误信息 命令执行正确返回 "OK"
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 其他: 失败</li> </ul>

设置用户自定义信息: **set\_custom\_user\_info**

请求命令参数

<pre>{   "cmd": "set_custom_user_info",   "id": "132456",   "body": {     "manufacturer": "SJ-1",     "data": {"ver": 1, "is_zoom": 0, "class": 1}   } }</pre>
--

结果返回

<pre>{   "cmd": "set_custom_user_info",   "id": "132456",   "state_code": 200 }</pre>
---

字段名	说明
body	需要设置的信息 manufacturer: 字符串标识的协议适配厂家 data: 自定义信息（不同厂家结构不同，以 SJ-1 为例） <ul style="list-style-type: none"> <li>● <b>ver</b> 版本，整数 0~4，0 表示没有设置。</li> <li>● <b>is_zoom</b> 是否支持变焦，整数 0~2，1 不支持，2 支持，0 表示没有设置。</li> <li>● <b>class</b> 设备等级（或分类），整数 0~16，0 表示没有设置。</li> </ul> 注：值为 0（没有设置）的项不会通过广播发送。
id	序列字符串（唯一），字符串长度 < 30，等于请求时的 id
state_code:	状态码 200: 成功 其他: 失败

### 5.17.防回退版本控制

用户可以设置是否开启防回退版本控制

#### 5.17.1. 获取防回退版本参数: get\_rollback\_cfg

请求命令参数

```
{
  "cmd": "get_rollback_cfg",
  "id": "123456"
}
```

结果返回

```
{
  "body": {
    "prohibit_version_rollback": false,
    "time_seconds": 123
  },
  "type": "ss_get_soft_cfg",
  "err_msg": "success",
  "state_code": 200
}
```

字段名	说明
body:	返回数据: prohibit_version_rollback: 是否开启防回退 time_seconds: 开启防回退有效, 大于等于该时间戳才能升级

设置用户自定义信息: set\_rollback\_cfg

请求命令参数

```
{
  "cmd": "set_rollback_cfg",
  "id": "132456",
  "body": {
    "prohibit_version_rollback": true,
    "time_seconds": 1732982400
  }
}
```

结果返回

```
{
  "cmd": "set_rollback_cfg",
  "error_msg": "Sucess",
  "id": "132456",
  "state_code": 200
}
```

字段名	说明
-----	----

body	需要设置的信息 prohibit_version_rollback: 是否开启防回退控制 time_seconds: 开启防回退后才生效, 设置防回退版本时间戳, 大于该时间戳才允许升级。秒级时间戳:1732982400 备注: 为了统一正式版本和临时版本, 建议设置的时间戳为年月日生成得时间戳, 比如 2024-12-01 00:00:00;防止统一正式版本和临时版本做转换时, 丢失精度, 导致与实际结果不符
, id	序列字符串(唯一), 字符串长度 < 30, 等于请求时的 id
state_code:	状态码 200: 成功 其他: 失败

5.18.用户通过 R3 获取伴侣机 Rg 的相关配置信息

5.18.1. 通过 R3 获取指定伴侣机的配置信息:  
user\_get\_mate\_info

请求命令参数

{ "cmd": "user_get_mate_info", "id": "12345693" }
--

结果返回

{ "body": [ { "dev_name": "SVZT", "device_type": "RG", "ip": "192.168.13.112", "online": 1, "sn": "e7cc75b8-d5774474" } ], "cmd": "user_get_mate_info", "error_msg": "OK", "id": "12345693", "state": 200, "state_code": 200 }
---

字段名	说明
ip	伴侣机 Ip 地址
dev_names	伴侣机设备名
sn	伴侣机序列号

device_type	伴侣机设备类型
online	伴侣机是否在线 0 离线 1 在线
state_code:	状态码 <ul style="list-style-type: none"><li>● 200: 成功</li><li>● 其他: 失败</li></ul>

5.18.2. 通过 R3 获取伴侣机的 Rtsp 代理信息:user\_get\_rtsp\_info

请求命令参数

{ "cmd": "user_get_rtsp_info", "id": "12345694", "body": { "sn": "e7cc75b8-d5774474" } }
--

结果返回

{ "body": { "proxyurl": "rtsp://192.168.13.81:8557/afa5852b-6945f0f3_h264", "proxyurl_sub1": "rtsp://192.168.13.81:8557/afa5852b-6945f0f3_sub_stream_1" }, "cmd": "user_get_rtsp_info", "error_msg": "OK", "id": "12345694", "state": 200, "state_code": 200 }
--

字段名	说明
sn	伴侣机的序列号
proxyurl	转发代理伴侣机主码流地址
proxyurl_sub1	转发代理伴侣机子码流地址
state_code:	状态码 <ul style="list-style-type: none"><li>● 200: 成功</li><li>● 其他: 失败</li></ul>

5.18.3. 通过 R3 获取伴侣机的语音代理信息:user\_request\_talkback

请求命令参数

{
---

```
"cmd": "user_request_talkback",
"id": "1234569",
"body": {
  "sn": "e7cc75b8-d5774474",
  "window_size": 320
}
}
```

结果返回

```
{
  "body": {
    "encode_type": 3,
    "port": 10011,
    "status": 1,
    "sampling_rate": 8000,
    "window_size": 1024,
    "sample_point": 512
  },
  "cmd": "user_request_talkback",
  "error_msg": "OK",
  "id": "1234569",
  "state": 200,
  "state_code": 200
}
```

请求字段	说明
cmd	消息类型
id	消息 ID，可以忽略。
window_size	每次发送数据包的大小，建议设置为 320.
响应字段	说明
cmd	消息类型
Id	消息 ID，可以忽略。
state_code	状态码 <ul style="list-style-type: none"><li>200: 成功</li><li>其他: 失败</li></ul>
port	语音对讲通信 TCP 服务端口
window_size	每次发送数据包的大小，摄像机返回，缓冲的字节数,固定值。
sample_point	采样点，固定值 512。
encode_type	支持的编码类型，目前 RG 设备仅支持 G711A <ul style="list-style-type: none"><li>1. PCM</li><li>2. G711</li><li>3. G711A</li></ul>
sampling_rate	采样率固定值 8000(默认) 16000，表示 8khz 16khz。
status	对讲状态,固定值 0



	0 空闲 1 忙碌
<p>备注:</p> <p>1. 首先通过协议获取音频参数</p> <p>"encode_type":3, "port":2707, "sampling_rate": 8000, "window_size":1024, "sample_point":512</p> <p>encode_type 音频编码格式: 1:PCM; 2:G711;3:G711A port 音频 tcp 服务代理转发端口号 sampling_rate 采样率 window_size 缓存区大小 sample_point 采样点个数</p> <p>2. 建立 tcp 连接到相机代理音频服务端口, 收发音频数据 相机向 tcp 客户端发送的音频数据为: 4 字节数据长度+G711A 音频数据,总长度为 sample_point +4 tcp 客户端向相机发送的音频数据与上一致, 同样为: 4 字节数据长度+G711A 音频数据,总长度为 sample_point +4</p>	

5.18.4. 通过 Tcp 获取日志指定的日志:log\_search

请求命令参数

```
{
  "cmd": "log_search",
  "id": "12345697",
  "body": {
    "start_time": 1627315200,
    "end_time": 1627401599,
    "qtype": 1,
    "max_id": 0,
    "min_id": 0,
    "start_id": 0
  }
}
```

结果返回

```
{
  "body": {
    "logs": [],
    "qnode": {
      "last_id": 658864
      "max_id": 658963,
      "min_id": 647353,
      "start_id": 4294967295
    }
  },
}
```

```

"err_msg": "OK",
"state": 200,
"type": "log_search"
}

```

字段名	说明
start_time	开始时间时间戳
end_time	结束时间时间戳
qtype	日志排序方式，默认 1 表示降序,(0 升序)
max_id	最大的日志 id，第一次获取应答包主动赋值，用于控制搜索次数，除了第一包其他的后续包都需要设置，第一包必须是 0
min_id	最小的日志 id，第一次获取应答包主动赋值，用于控制搜索次数，除了第一包其他的后续包都需要设置,第一包必须是 0
start_id	第一次默认设置为 0，后面是应答包 last_id 的值
last_id	应答包的最后一个日志的 id，用于设置下次发送的 start_id，最后一个包是 0
logs	详细日志信息
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 其他: 失败</li> </ul>

### 5.18.5. 设置 Rg 人车滞留事件上报使能:stay\_event\_enable

请求命令参数

```

{
  "cmd": "stay_event_enable",
  "id": "12345701",
  "body": {
    "enable": true,
    "format": "json",
    "image": true,
    "image_type": 0
  }
}

```

字段名	说明
enable	是否允许推送识别结果，默认值: false <ul style="list-style-type: none"> <li>● true 表示允许推送识别结果</li> <li>● false 表示不推送识别结果</li> </ul>
format	推送识别结果数据格式，默认值: json json 表示识别结果数据格式为 json 数据格式
image	识别结果是否包含图片，默认值: true <ul style="list-style-type: none"> <li>● true 表示识别结果包含图片</li> <li>● false 表示识别结果不包含图片</li> </ul>
image_type	识别的图片类型，默认值: 0 <ul style="list-style-type: none"> <li>● 0 表示返回识别结果全图</li> </ul>

	<ul style="list-style-type: none"> <li>● 1 表示只返回车牌区域小图</li> <li>● 2 表示返回两种图片</li> </ul>
--	---

**结果返回**

<pre>{   "cmd": " stay_event_enable",   "id": "12345701",   "state_code": 200 }</pre>
---

字段名	说明
state_code:	状态码 <ul style="list-style-type: none"> <li>● 200: 成功</li> <li>● 其他: 失败</li> </ul>

当配置完成后（配置为允许推送识别结果的情况 enable: ture），并且在服务器端有 ivsresult 识别结果产生后，tcp server 会主动推送 common\_alarm\_result 识别结果给客户端，推送的识别结果格式如下

**配置完成后，接收到识别结果为配置值：**

正常情况下，一个完整的识别结果（返回 2 种图片的情况）推送消息格式如下：

'V'	'Z'	0	0	长度（数据包的大小）
识别结果字符串+字符串结束符'\0'				
全图的 jpg 格式数据				
小图片（车牌区域图片）的 jpg 格式数据				

例如下面就是一个正常的 JSON 识别推送结果：

<pre>{   "module": "TCP_BUS_REQUEST",   "type": "common_alarm_result",   "cmd": "common_alarm_result",   "device_info": {     "alg_chn": 0,     "chn_id": 0,     "dev_ip": "192.168.19.211",     "dev_port": 80,     "rule_id": 0   },   "full_image": [     {       "fe_crop_info": {         "height": 1140,         "left": 141,         "top": 181,         "width": 2001       },       "full_image_id": 17,       "image_type": 0,       "file_name": "/media/mmcblk0p1/test.jpg"     }   ] }</pre>
---

```
    }
  },
  "small_image": [
    {
      "small_image_id": 18,
      "image_type": 0,
      "file_name": "/media/mmcblk0p1/test.jpg"
    }
  ],
  "results": [
    {
      "event_type": 81,
      "group_id": 0,
      "plate_result": {
        "color_type": "Utf8_Base64",
        "confidence": 0,
        "license": "Utf8_Base64",
        "plate_true_width": 0,
        "type": "Utf8_Base64"
      },
      "lane_ctrl_prop": {
        "car_stay_result": {
          "alarm_state": 1
        },
        "person_stay_result": {
          "alarm_state": 0
        },
        "image_id": 18
      },
      "result_id": 8,
      "time": "2021-11-10 18:28:50",
      "type": "common_alarm_lane_control",
      "trigger_type": 0
    }
  ]
}
```

```
//json 后接图片数据，数据排列格式
|数据头+图片数据|数据头+图片数据|
/* 图片二进制数据头 */
static const char NET_MARK_0 = 'V';
static const char NET_MARK_1 = 'Z';
typedef struct _ImageHead {
uint8 mark[
    2
]; // [0]='V' [1]='Z'
uint16 image_type; // 图片类型; 0=jpg,1=png
uint32 image_id; // 图片 id
uint32 data_size; // 图片长度;
} ImageHead;
```

其中各字段的含义如下表所示：

字段名	说明
-----	----

module	处理模块, 兼容协议, 无用, 不用关心, 恒定值
type	报警事件, 兼容协议, 不用关心, 恒定值
cmd	消息类型, SDK 专用
device_info	设备信息-兼容老设备协议
alg_chn	算法通道恒为 0
chn_id	通道 ID 恒为 0
dev_ip	设备 IP
dev_port	web 端口
rule_id	算法检测框的 ID 0: 虚拟线圈 1: 识别区域 2: 辅线圈 3 布防区域(多线圈时使用)
full_image	背景图信息
fe_crop_info	特征图坐标(只有背景图才有特征图信息)
file_name	图片文件名, 兼容协议, 不用关心
full_image_id	背景图 ID 由相机决定, 方便取相机存储中的图片
image_type	图片格式 0: jpeg
small_image	特征图
small_image_id	特征图 ID 由相机决定, 方便取相机存储中的图片, 和 image_id 一致
image_type	图片格式 0: jpeg
file_name	图片文件名, 兼容协议, 不用关心
plate_result	车牌信息
color_type	车牌颜色 (Utf8_Base64)
confidence	置信度
license	车牌号 (Utf8_Base64)
plate_true_width	车牌宽度
type	车牌类型(Utf8_Base64)
event_type	<p>十六进制[(十进制):事件类型, 实际上报的是十进制, 详细类型说明如下:</p> <ul style="list-style-type: none"> <li>● 0x49(79): 人滞留解除</li> <li>● 0x50(80): 人滞留</li> <li>● 0x51(81): 车滞留</li> <li>● 0x52(82): 车滞留解除</li> <li>● 0x53(83): 机动车辆通行事件</li> <li>● 0x54(84): 机动车辆折返事件</li> <li>● 0x70(112): 非机动车滞留</li> <li>● 0x71(113): 非机动车滞留解除</li> <li>● 0x80(128): 道闸正常</li> <li>● 0x81(129): 道闸异常</li> <li>● 0x82(130): 道闸回落</li> <li>● 0x83(131): 道闸抬升</li> <li>● 0x90(144): 车辆拥堵</li> <li>● 0x91(145): 车辆拥堵解除</li> </ul>

group_id	此字段功能是滞留事件和滞留解除事件、道闸正常和道闸异常事件配对使用,重启后置 0。
image_id	小图 id, 与 small_image_id 这个是一个值。
lane_ctrl_prop	兼容协议, 滞留报警 人 车 非
car_stay_result	车辆滞留报警
alarm_state	车辆滞留报警 0 正常 1 报警
person_stay_result	人员滞留报警 0 正常 1 报警
image_id	小图 ID 兼容老协议, 对接不用关心该字段
错误! 未找到图形项目表。	
time	事件输出时间
type	报警类型, 固定值, 对接可以不关心。
trigger_type	兼容协议, 触发类型 恒定值 0
备注: 1、每个事件类型由 event_type 的值来决定 2、每个事件都有背景图 full_image 和 特征图 small_image 3、只有机动车事件(机动车滞留 通行 折返)与 plate_result 字段有关系, 其他事件则没有该字段 4、lane_ctrl_prop 是车辆滞留报警和人员滞留报警, 兼容老协议用的, 可以不用关心 5、兼容协议的内容, 如果没有对接老 RG 产品可以不用关心这些字段	

## 6. 码表

### 6.1. Gpio 状态码表

Gpio 状态码表 1

状态	描述
0	低电平
1	高电平