车牌识别一体机

TCP 通讯协议说明手册

前言

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。

版本变更记录

版本号	拟制日期	版本描述	存档编号
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	

TCP server 通讯协议文档 第 1 页 共 35 页

目 录

1 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。

1.1 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
D ata								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** JSON，一般的命令和反馈数据都使用 JSON。 ○ **0X01** 心跳包，数据长度为 0， DATA 为空
  ○ **0X02** 二进制数据，用户按情况配置的二进制数据，一般用在接收识别结果中。
- 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 htol 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 = 1MB。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。

1.2 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。协议通用格式如下：

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

```
response:
```

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

字段名	说明
cmd	命令字符串
body	传递的数据，可为空
state	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

2 Tcp 协议命令

2.1 设备维护管理

2.1.1 获取设备的序列号: getsn

请求命令格式

```
{
  "cmd": "getsn"
}
```

结果返回值

如果一切正常则返回如下数据：

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

2.1.2 获取设备当前时间戳: get_device_timestamp

请求命令参数

```
{
  "cmd": "get_device_timestamp"
}
```

结果返回

```
{
  "cmd": "get_device_timestamp",
  "timestamp": 1443213456
}
```

字段名	说明
timestamp	时间（格林威治时间，单位秒）

2.1.3 设置系统时间 请求命令参数

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

字段名	说明
● timestring	● 时间字符串，格式必须是：“XXXX-XX-XX XX:XX:XX”

结果返回

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

字段名	说明
state_code:	状态码 ● 200： 成功 ● 400：客户端请求出错（timestring 错误） ● 500：服务器内部错误（内部设置错误）

2.2 车牌识别

2.2.1 配置推送数据方式: ivsresult 请求命令参数

```
{
  "cmd": "ivsresult",
  "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 ● true 表示允许推送识别结果 ● false 表示不推送识别结果
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 ● 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
image_type	识别的图片类型，默认值：0 ● 0 表示返回识别结果全图 ● 1 表示只返回车牌区域小图 ● 2 表示返回两种图片

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 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

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:

```
{
 "PlateResult" : {
```

```
    "bright" : 0,
    "carBright" : 0,
    "carColor" : 0,
    "colorType" : 1,
    "colorValue" : 0,
    "confidence" : 99,
    "direction" : 0,
    "license" : "青 PTW3Z3",
    "location" : {
      "RECT" : {
        "bottom" : 392,
        "left" : 690,
        "right" : 834,
        "top" : 350
      }
    },
    "timeStamp" : {
      "Timeval" : {
        "sec" : 1458882234,
        "usec" : 921325
      }
    },
    "timeUsed" : 0,
    "triggerType" : 1,
    "type" : 1
  },
  "active_id" : 0,
  "clipImgSize" : 1103,
  "cmd" : "ivs_result",
  "fullImgSize" : 51566,
  "id" : 0,
  "imageformat" : "jpg",
  "timeString" : "2016-03-25 13:03:54"
}
```


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）
colorValue	车牌颜色
colorType	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
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
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	触发结果的类型,见 TH_TRIGGER_TYPE_BIT
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

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 图片数据 **(**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**)**				

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。

2.2.2 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 ● true 需要识别结果图片 ● false 不需要识别结果图片
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 ● json 表示识别结果数据格式为 json 数据格式。

结果返回值

这个命令的结果返回值，与配置推送数据方式里面的结果返回值是一样的,请参考。

2.2.3 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，与配置推送数据方式里面的结果返回值是一样的，请参考。

2.2.4 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"
}
```

结果返回

```
{
  "cmd": "get_max_rec_id",
  "max_id": 1024
}
```

字段名	说明
max_id	识别结果记录中最大的 id 值

2.2.5 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，与配置推送数据方式里面的结果返回值是一样的，请参考。

2.2.6 获取记录图片: get_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
}
```

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。

2.2.7 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

2.3 设备硬件参数配置

2.3.1 控制 IO 输出: ioctl 请求命令参数示例

```
{
  "cmd" : "ioctl",
  "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
value	输出 IO 的状态值 ● 0 断 ● 1 通 ● 2 先通后断（一般做开闸使用）
delay	先通后断的延迟时间

结果返回 无返回

2.3.2 获取 IO 输入状态: get_gpio_value 请求命令参数示例

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

结果返回

```
{
  "cmd" : "get_gpio_value",
  "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 ● 0 低电平 ● 1 高电平

2.3.3 自动聚焦: auto_focus

请求命令参数

```
{
  "cmd": "auto_focus"
}
```

结果返回 无返回,可观察到设备有自动聚焦的动作发生

2.4 设备连接

2.4.1 通知在线消息:response_online

请求命令参数

```
{
  "cmd": "response_online"
}
```

结果返回 无结果返回

2.5 配置透明通道: ttransmission

请求命令参数

```
{
  "cmd": "ttransmission",
  "subcmd": "send",
  "datalen": 6,
  "data": "ABCDEF",
  "comm": "rs485-2"
}

{
  "cmd": "ttransmission",
  "subcmd": "init",
  "data": "rs485-2"
}

{
  "cmd": "ttransmission",
```

```
  "subcmd": "uninit"
}
```

字段名	说明
subcmd	子命令 ● init 初始化 ● uninit 取消初始化 ● send 发送数据
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 ● rs485-1 ● rs485-2

结果返回 如果初始化错误，则返回:

```
{
  "cmd": "ttransmission",
  "subcmd": "init",
  "response": "failed"
}
```

如果初始化成功，则返回:

```
{
  "cmd": "ttransmission",
  "subcmd": "init",
  "response": "ok"
}
```

如果取消初始化失败，则返回

```
{
  "cmd": "ttransmission",
  "subcmd": "uninit",
  "response": "failed"
}
```

如果取消初始化成功，则返回:

```
{
  "cmd": "ttransmission",
  "subcmd": "uninit",
  "response": "ok"
}
```

发送数据没有任何返回值。

字段名	说明
subcmd	子命令 ● init 初始化 ● uninit 取消初始化 ● send 发送数据
response	返回结果

2.6 白名单

2.6.1 脱机检查

2.6.1.1 注册脱机功能: offline

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",
    "interval" : 2
}
```

结果返回

```
{
  "cmd": "offline",
  "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

2.6.1.2 取消注册脱机功能　请求命令参数示例

```
{
  "cmd" : "reg_offline_check",
   "sucmd" : "cancel"
}
```

无返回结果

字段名	说明
sucmd	当 subcmd 字段为“cancel”时，取消注册脱机功能，没有此字段，或此字段不为 cancel,此命令为注册脱机功能

2.6.2 操作白名单：white_list_operator

2.6.2.1 增加或者更新白名单

```
{
  "cmd": "white_list_operator",
  "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": 1,
    "seg_time": "2016-10-20 12:30:40",
    "need_alarm": 0,
    "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 ● update_or_add 增加或更新白名单 ● delete 删除白名单 ● 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
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 32 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

增加或者更新返回值:

```
{
  "cmd": "white_list_operator",
  "operator_type": "update_or_add",
  "state": "succeed"
}
```

字段名	说明
operator_type	子命令 ● update_or_add 增加或更新白名单 ● delete 删除白名单 ● select 白名单查询
state	返回结果

2.6.2.2 以车牌号查询白名单

```
{
  "cmd" : "white_list_operator",
  "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 ● update_or_add 增加或更新白名单 ● delete 删除白名单 ● Select 白名单查询
sub_type	子子命令 ● plate 根据车牌号进行查询
plate	车牌号

以车牌号查询白名单返回结果：

```
{
  "cmd" : "white_list_operator",
  "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" : "2015-1-20 11:00:00",
      "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 ● update_or_add 增加或更新白名单 ● delete 删除白名单 ● select 白名单查询
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
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
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 32 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

2.6.2.3 白名单删除

```
{
    "cmd":"white_list_operator",
    "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",
    "operator_type":"delete",
    "state":"succeed"
}
```

字段名	说明
operator_type	子命令 ● update_or_add 增加或更新白名单 ● delete 删除白名单 ● select 白名单查询
state	返回结果状态
plate	车牌号

2.7 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（encrpty_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

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

2.7.1 获取加密方式 get_ems 请求命令参数

```
{
  "cmd" : "get_ems"
}
```

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"
    }
  ],
  "signature" : "BNjnUHLkgPaLH3sI",
  "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 ● **m_id** 加密方式的索引 ● **name** 加密方式的名字
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 ● 200：成功

2.7.2 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",
  "new_encrypt_key" : "j6NrhO8gWFKU0O8mk8+A/g==",
  "prime_key" : "Vg0MgpeBCOzzCxbsQ68V2NHd0Zk="
}
```

字段名	说明
prime_key	主密钥字串（加密之后的密码） ● 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
new_encrypt_key	新设置的用户密码（加密之后的密码） ● 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

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",
  "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 ● 200： 成功 ● 401：用户没有权限 ● 400：用户参数设置错误（有可能密码太长或者太短） ● 500：服务器内部错误（内部设置错误）

2.7.3 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**

```
{
 "cmd" : "change_encrypt_key",
 "encrypt_key" : "OB46x4SOg0ZTXa9BDFGh/i5LAfQ=",
 "new_encrypt_key" : "sZuBTdl+Fy0HjgFJ/yDmSA=="
}
```

字段名	说明
encrypt_key	旧的用户密码（加密后的） ● encrypt _key 的加密方法：由 HMAC-SHA 算法再加上原本的 encrypt _key（以前的用户密码），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
new_encrypt_key	新设置的用户密码（加密之后的密码） ● 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

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

```
{
 "cmd" : "change_encrypt_key",
 "signature" : "8GZiU0f8u2slTCsp",
 "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● 200： 成功 ● 401：用户没有权限 ● 400：用户参数设置错误（有可能密码太长或者太短） ● 500：服务器内部错误（内部设置错误）

2.7.4 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" : "9k1t5PhnUXlfgJKgHmcswU3PIp8=",
  "m_id" : 1
}
```

字段名	说明
encrypt_key	用户密码（加密后的） ● encrypt _key 的加密方法：由 HMAC-SHA 算法再加上原本的 encrypt _key（用户密码），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
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",
  "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 ● 200：成功 ● 401：用户没有权限 ● 400：用户参数设置错误（m_id 错误） ● 500：服务器内部错误（内部设置错误）

2.8 设备组网

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

2.8.1 得到当前设备 vzid 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 ● get_cdvzid 获取当前设备 vzid

结果返回

```
{
  "body": {
    "state": 200,
    "type": "get_cdvzid",
    "vzid": {
      "enable_group": true,
      "ip_addr": "192.168.4.157",
      "name": "4.157Enter",
      "sn": "0ba76bc6-e26d52eb",
      "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 ● true 开启 ● false 关闭
ip_addr	设备 ip
name	设备名字
sn	设备序列号
type	设备类型 ● input 入口设备 ● output 出口设备

2.8.2 得到所有已连接设备信息 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 ● get_ovzid 得到所有已连接的设备信息

结果返回

```
{
  "body": {
    "state": 200,
    "type": "get_ovzid",
    "vzid": [
      {
        "enable_group": false,
        "ip_addr": "",
        "name": "",
        "sn": "",
        "type": "unkown"
      },
      {
                                                "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",
  "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_ovzid 得到所有已连接的设备信息
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

2.8.3 得到入口设备记录 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

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_irs",
    "pos_begin": 0,
    "pos_end": 32
  }
}
```

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 ● get_irs 获取入口设备记录
pos_begin	获取记录的开始位置
pos_end	获取记录的结束位置

结果返回


```
{
  "body": {
    "irs_data": [
      {
        "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": 94,
            "direction": 0,
            "fragment_path": "/tmp/app/html/snapshot/lpr/patch_tri_snap_2.jpg",
            "id": 0,
            "image_path": "/tmp/app/html/snapshot/lpr/tri_snap_2.jpg",
"image_sd_path":
"/media/mmcblk0p1/VzIPCCap/2015_11_11/1533026308_鲁 NQ061C.jpg",
            "location": {
              "bottom": 545,
              "left": 918,
              "right": 1011,
              "top": 528
            },
            "n_time": 0,
            "plate": "鲁 NQ061C",
            "plate_color": 3,
            "timeval": {
              "tv_sec": 1447255982,
              "tv_usec": 632568
```

```
                },
                "trig_type": 8
            },
            "state": 1
        }        },
    {
        "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": 98,
                "direction": 0,
                "fragment_path": "/tmp/app/html/snapshot/lpr/patch_tri_snap_3.jpg",                "id":
0,
                "image_path": "/tmp/app/html/snapshot/lpr/tri_snap_3.jpg",
"image_sd_path":
"/media/mmcblk0p1/VzIPCCap/2015_11_11/1533173601_豫 NQ061C.jpg",
                "location": {
                    "bottom": 566,
                    "left": 810,
                    "right": 925,
                    "top": 545
                },
                "n_time": 0,
                "plate": "豫 NQ061C",
                "plate_color": 3,
                "timeval": {
                    "tv_sec": 1447255997,
                    "tv_usec": 360076
                },
                "trig_type": 1
            },
            "state": 1
        }
    }
  ],
  "irs_size": 400,
  "pos_begin": 0,
  "pos_end": 2,
  "type": "get_irs"
 },
 "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_irs 获取入口设备记录
state	状态码
irs_data	入口设备记录的列表（由多条入口记录组成）
irs_size	入口设备记录总条数
pos_begin	获取记录的开始位置
pos_end	获取记录的结束位置

（irs_data 内的字段）入口设备记录字段说明：

字段名	说明
input_record	一条入口设备记录，记录由设备 vzid+ivs_result+车牌状态组成；设备 vzid 和 ivs_result 的说明上面已经提到过了，请自行查找。车牌状态： ● 0 车牌已使用 ● 1 车牌未使用 ● 2 车牌失效 ● 3 车牌信息不全（出口设备未找到此车的进入信息）

2.8.4 得到出口设备记录 请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
  "body": {
    "type": "get_ors",
    "pos_begin": 0,
    "pos_end": 20
  }
}
```

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 ● get_ors 获取出口设备记录
pos_begin	获取记录的开始位置
pos_end	获取记录的结束位置

结果返回


```
{
  "body": {
    "ors_data": [
      {
        "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_10.jpg",          "id":
0,
            "image_path":
"/tmp/app/html/snapshot/lpr/tri_snap_10.jpg",
"image_sd_path":
"/media/mmcblk0p1/VzIPCCap/2015_11_11/2329507201_桂 JJ330Y.jpg",
            "location": {
              "bottom": 512,
              "left": 941,
              "right": 1021,
              "top": 498
            },
            "n_time": 0,
            "plate": "桂 JJ330Y",
            "plate_color": 3,
            "timeval": {
              "tv_sec": 1447284590,
```

```
                "tv_usec": 725158
              },
              "trig_type": 1
            },
            "state": 2
          },
          "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": 99,
              "direction": 4,
              "fragment_path":
"/tmp/app/html/snapshot/lpr/patch_tri_snap_6.jpg",          "id":
0,
              "image_path":
"/tmp/app/html/snapshot/lpr/tri_snap_6.jpg",
"image_sd_path":
"/media/mmcblk0p1/VzIPCCap/2015_11_12/1517592301_桂 JJ330Y.jpg",
"location": {
                "bottom": 627,
                "left": 941,
                "right": 1069,
```

```
                "top": 600
              },
              "n_time": 0,
              "plate": "桂 JJ330Y",
              "plate_color": 3,
              "timeval": {
                "tv_sec": 1447341479,
                "tv_usec": 235076
              },
              "trig_type": 1
            },
            "state": 0
          }
        }
    ],
    "ors_size": 45,
    "pos_begin": 0,
    "pos_end": 2,
    "type": "get_ors"
  },
  "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_ors 获取出口设备记录
state	状态码
ors_data	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）
ors_size	出口设备记录总条数
pos_begin	获取记录的开始位置

pos_end	获取记录的结束位置

（ors_data 内的字段）出口设备记录字段说明：

字段名	说明
input_record	入口记录信息，上面已经提到过了，请自行查找
output_record	出口记录信息，output_record 和 input_record 的定义一样

2.8.5 得到出口设备缓存记录 请求命令参数

```
{
  "cmd": "dg_json_request",
  "id": "132156",
   "body": {
     "type": "get_obrs",
     "pos_begin": 0,
     "pos_end": 20
   }
}
```

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 ● get_obrs 获取出口设备缓存记录
pos_begin	获取记录的开始位置
pos_end	获取记录的结束位置

结果返回 与得到入口设备记录一样，有分页的机制

2.8.6 得到当前组网内所有设备信息 请求命令参数

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

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 ● get_agdi 得到当前组网内所有设备信息

结果返回

```
{
  "body": {
    "all_gds": [
      {
        "is_connected": false,
        "vzid": {
          "enable_group": false,
          "ip_addr": "192.168.1.104",          "name":
"ABCED",
          "sn": "4b01fc33-40cdb008",
          "type": "unkown"          }
      }
    ],
    "type": "get_agdi"
  },
  "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_agdi 得到当前组网内所有设备信息
● all_gds	设备信息列表（由多条设备信息组成），一条设备信息由连接状态和 vzid 组成，vzid 上面已经提到过了，请自行查找，连接状态 is_connected ● false 未连接 ● true 已连接

2.8.7 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 ● true 允许接收组网识别消息 ● 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,
          "timeval": {
            "tv_sec": 1447342483,
            "tv_usec": 351562
          },
          "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,
          "timeval": {
            "tv_sec": 1447342483,
            "tv_usec": 351562
          },
          "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 ● 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
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,
          "timeval": {
            "tv_sec": 1447342410,
            "tv_usec": 91949
          },
          "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