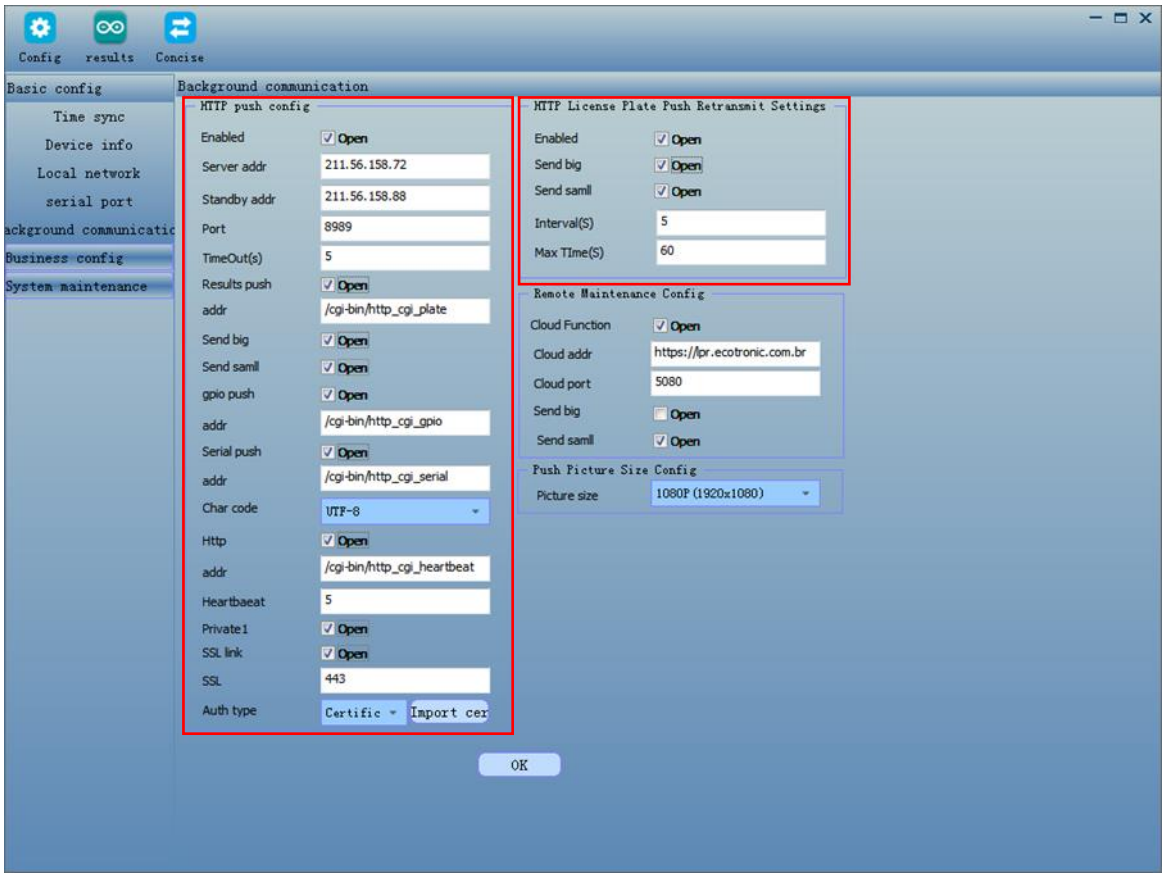
Parking Cam LPR camera HTTP API V1.2

Contents

1 Overview

The LPR camera API is based on HTTP V1.1, this requires the user to set up an HTTP server and configure the address of this HTTP server inside of LPR camera. When the LPR camera has a recognition result or other content that needs to be pushed, it will send HTTP commands to the server address. The data interaction content is in JSON format, which is sensitive with uppercase and lowercase letters.

2 Camera configuration instructions



2.1 Description of HTTP push fields

Enable	Enable HTTP / HTTPS push function on camera
Server address	HTTP server network IP address or domain name
Server alternate address	When the HTTP server address is unreachable, the camera will connect to the alternate address
Port	HTTP server port number
Overtime time	A HTTP (request , response) timeout, if this time is exceeded , the camera will close socket Note : When the camera push heartbeat is enable, and heartbeat interval is shorter than the timeout , socket will not be closed , keep the long connection
Result push	License plate recognition result push enable
address	Server path for license plate recognition result push
send pictures	License plate recognition results include large picture (Enable/disable)
Send small picture	License plate recognition results include small picture (enable/disable)
GPIO push	I/O input trigger information push enable/diable
address	Server path for IO input to trigger information push
Push serial data	485 input data push enable switch
address	485 Input data push path addressConfiguration
Character Encoding	Character coding (GB2312, UTF-8) option
HTTP heartbeat	Heartbeat push enable/disable
address	Server path for heartbeat push
Heartbeat interval	Interval for heartbeat push , in seconds
Private protocol	Customized http connection protocol, default is disable. (Don't not change this configuration)
SSL connection	HTTPS transmission enable switch
SSL port	HTTPS server port number
Type of authentication	Anonymous : Do not verify the certificate CA certificate : Verify the server certificate , must import the CA certificate

2.2 HTTP LPR result push retransmission.

Enable/Disable	License plate recognition push retransmission function is enabled Note :

	1) Camera keeps retransmitting until server responds stops retransmitting, or open gate command. 2) The maximum quantity of Plate Number queues for camera retransmission is 10. If it exceeds, the oldest one will be dismissed, according first in first out ruler. 3) After the camera restarts , the retransmission queue will be lost 4) The default retransmission interval is the configured timeout time (maximum 20S)
Send big picture	When retransmitting license plate recognition, including large picture
Send thumbnail	When retransmitting license plate recognition, including small picture
Unattended	When the function is enabled 1) The camera is working in “online mode” when it connected to server, server controls access permission. 2) The camera is working “offline mode” when lost connection with server, camera working in “standalone”, controls access permission, according pre-defined rulers.

3 Data type definition

3.1 License plate color macro definition E_PlateColor

```
typedef enum
{
        UNKNOWN_PLATE = 0,
        BLUE_PLATE,
        YELLOW_PLATE,
        WHITE_PLATE,
        BLACK_PLATE,
        GREEN_PLATE,
        YELLOW_GREEN_PLATE,
        BLACK_PLATE_OTHER
}E_PlateColor ;
```

3.2 License plate type macro definition ITS_PlateType

```
typedef enum
{
```

```
    PLATE_TYPE_NULL = 0, // Unknown
    PLATE_TYPE_BLUE, // Not used in brazil
    PLATE_TYPE_BLACK, // Not used in brazil
    PLATE_TYPE_YELL, // Not used in brazil
    PLATE_TYPE_YELL2, // Not used in brazil
    PLATE_TYPE_POL, // Not used in brazil
    PLATE_TYPE_APOL, // Not used in brazil
    PLATE_TYPE_APOL2, // Not used in brazil
    PLATE_TYPE_ARM, // Not used in brazil
    PLATE_TYPE_ARM2, // Not used in brazil
    PLATE_TYPE_INDI, // Personal license plate
    PLATE_TYPE_NEWN, // New energy license plate
    PLATE_TYPE_NEWN1, // New energy license plate
    PLATE_TYPE_EMB, // Embassy license plate
    PLATE_TYPE_CON, // Consulate license plate
    PLATE_TYPE_MIN, // Civil Aviation License Plate
} ITS_PlateType ;
```

3.3 Model macro definition E_VehiclSieze

```
typedef enum
{
        UNKNOWN_SIZE,
        LARGE_VEHICLE, / * large car * /
        MIDDLE_VEHICLE, / * medium car * /
        SMALL_VEHICLE, / * Small car * /
} E_VehiclSieze ;
```

3.4 Trigger mode defined E_SnapMode

```
typedef enum
{
        SNAP_MODE_UNKNOW = 0,
        SNAP_MODE_MANUAL, / * Manual * /
        SNAP_MODE_VIDEO, / * Video detection* /
        SNAP_MODE_LOOP, / * Induction coil detection* /
        SNAP_MODE_MAX,
} E_SnapMode ;
```

4 Communication between camera and server.

4.1 Camera push data to server:


```
{
    "AlarmInfoPlate ": {
    "channel ": 0,
    "deviceName ":" default ",
    " ipaddr ": "192.168.0.100",
    "result ": {
    "PlateResult ": {
        " bright ": 0,
        "carBright ": 0,
        "carColor ": 0,
        "colorType ": 0,
        "colorValue ": 0,
        "confidence ": 0,
        "direction ": 0,
        "license ":" AAA8888",
        "location ": {
            "RECT ": {
                " left ": 0,
                " top ": 0,
                "right ": 0,
                " bottom ": 0
            }
        },
    " timeStamp ": {
        "Timeval ": {
            "sec ": 1441815171,
            "usec ": 0
        }
    },
    "timeUsed ": 0,
    "triggerType ": 1,
    "type ": 0
    }
    },
    "serialno": "e10b2d6c8c07b422361457935b518642"
    }
}
```

The meaning of each field is as follows:

Field name	Field Type	description
AlarmInfoPlate		Push result is the license plate recognition result
channel	Int	Channel number (reserved)
deviceName	String	Device name (can be configured on camera configuration > device information- > project name)
ipaddr	String	Camera IP address
serialno	String	Camera serial number , camera unique identification (can be viewed on camera configuration > device information > camera serial number)
result		Recognition result data
PlateResult	String	License plate related information
bright	Int	Reserve
carBright	Int	Vehicle body brightness (reserved)
carColor	Int	Vehicle body color (reserved)
colorType	Int	License plate color reference E_PlateColor
colorValue	Int	Reserve
confidence	Int	Reliability of recognition results (0-100)
direction	Int	Moving Direction (0: Unknown 1: coming 2: Going)
license	String	License plate number
location		License plate position in picture
RECT		The position is a rectangular area (indicated by the coordinates of the upper left corner and the lower right corner)
left	Int	Upper left corner _X coordinate
top	Int	Top left corner _Y coordinate
right	Int	Bottom right corner_X coordinate
bottom	Int	Bottom right corner_Y coordinate
timeStamp		The current recognition , the camera's current time , and the corresponding time stamp
Timeval		Timestamp structure type
sec	Int	From 1970 Nian 1 Yue 1 the number of seconds to identify the time of day
usec	Int	0
timeUsed	Int	Reserve
triggerType	Int	The trigger type of the current recognition result , see E_SnapMode
type	Int	License plate type , see ITS_PlateType
imageFile	String	Large image , base64- encoded image data enables large image sending
imageFileLen	Int	Large image base64 encoded image data length
imageFragmentFile	String	Small image , base64- encoded image data enables small image sending
imageFragmentFileLen	Int	Small image base64 encoded image data length

4.2 HTTP server response data content


```
{
    "Response_AlarmInfoPlate":
{
    "info":"ok",
    "content":"retransfer_stop",
    "is_pay":"true",
```

```
        "serialData":[
            {
                "serialChannel":0,
                "data":"MTEyMzQ1Njc4OQ==",
                "dataLen":10
            },
            {
                "serialChannel":1,
                "data":"MTEyMzQ1Njc4OQ==",
                "dataLen":10
            }
        ]
    }
}
```


Field name	Field Type	have to	description
Response_AlarmInfoPlate		Y	Response is the response pushed for the recognition result
info	String	Y	"Ok" means access authorized, open gate, any other characters such as "no" means not open Note: When server reply "ok", retransmission of result will stop.
content	String	Y	"Retransfer_stop" command to stop the retransmission of current OCR result.
Is_pay	String	Y	Reserve
serialData		Y	Serial port transparent data transfer to other peripherals, for example, a LED display. Note: This serial data is optional, could with or without.
serialChannel	Int	N	485 channel number 0 : Transparent transmission to A1, B1 1 : Transparent transmission to A2, B2
data	String	N	485 transparent transmission data , BASE64 encoded data
dataLen	Int	N	485 transparent transmission data length , as BASE64 before encoding

5 IO input trigger interaction data content description

5.1 Camera request data content


```
{
"AlarmGioIn" : {
    "deviceName" : "default",
```

```
        "ipaddr" : "192.168.0.100",
        "result" : {
            "TriggerResult" :
            {
                "source" : 0,
                "value" : 0
            }
        },
        "serialno" : "e10b2d6c8c07b422361457935b518642"
        }
    }
```


Field name	Field Type	description
AlarmGioIn		Push message is triggered by IO input
deviceName	String	Device name (can be configured on WEB > device information- > project name)
ipaddr	String	Camera IP address
serialno	String	Camera serial number , camera unique identification (can be viewed on the WEB > device information- > camera serial number)
result		IO input information structure
TriggerResult		IO input information
source	Int	Input serial number 0: indicates input 1 1: indicates input 2
value	Int	Status of the input when triggered

5.2 HTTP server response data content

Server don't reply anything when receive a I/O push.

6 Description of serial port input interactive data content

6.1 Camera request data content


```
{
```

```
    "SerialData":
    {
        "channel":0,
        "serialno":"e10b2d6c8c07b422361457935b518642",
        "ipaddr":"192.168.0.100",
        "serialChannel":0,
        "data":"MTEyMzQ1Njc4OQ==",
        "dataLen":10
    }
}
```


Field name	Field Type	description
SerialData		Push message is RS485 data input
channel	Int	Channel number (reserved)
serialno	String	Camera serial number , camera unique identification (can be viewed on the WEB > device information- > camera serial number)
ipaddr	String	Camera IP address
serialChannel	Int	RS485 channel number 0 : Data input from A1, B1 1 : Data input from A2, B2
data	String	RS485 input data , BASE64 encoded data
dataLen	Int	RS485 input data , data length before BASE64 encoding

6.2 HTTP server response data content


```
{
 "Response_SerialData":
 {
    "info":"",
    "serialData":[
        {
            "serialChannel":0,
            "data":"MTEyMzQ1Njc4OQ==",
            "dataLen":10
        },
        {
        "serialChannel":1,
        "data":"MTEyMzQ1Njc4OQ==",
        "dataLen":10
```

```
            }
        ]
    }
}
```

Field name	Field Type	have to	description
Response_SerialData		Y	The response is a RS485 input data push response
info	String	Y	Reserve
serialData		Y	Serial port transparent data array Note: This part is optional, depending on the actual situation
serialChannel	Int	N	RS485 channel number 0 : Data input from A1, B1 1 : Data input from A2, B2
data	String	N	RS485 input data , BASE64 encoded data
dataLen	String	N	RS485 input data length before BASE64 encoding

7 Heartbeat interactive data Description

7.1 Camera request data content

```
{
    "Heartbeat":{
        "countid":1,
        "timeStamp" :
        {
            "Timeval" : {
                "sec" : 1441815171,
                "usec" : 0
            }
        },
        "serialno" : "e10b2d6c8c07b422361457935b518642"
    }
}
```

Field name	Field Type	description
Heartbeat		Push message for heartbeat
countid	Int	Heartbeat count

serialno	String	Camera serial number , camera unique identification (can be viewed on the WEB client- > device information- > camera serial number)
timeStamp		This heartbeat , camera current time , corresponding timestamp
Timeval		Timestamp structure type
sec	Int	Total seconds counting from 0:00 o'clock,1st of Jan, 1970 to current time.
usec	Int	0

7.2 HTTP server response data content

No replay data, Or reply below


```
{
    "Response_Heartbeat":{
        "info":"ok",
        "serialData":[
        {
            "serialChannel":0,
            "data":"MTEyMzQ1Njc4OQ==",
            "dataLen":10
        },
        {
            "serialChannel":1,
            "data":"MTEyMzQ1Njc4OQ==",
            "dataLen":10
        }
        ],
        "shutoff":"ok",
        "snapnow":"yes"
    }
}
```


Field name	Field Type	have to	description
Response_Heartbeat		Y	The response is for heartbeat
info	String	Y	"Ok" means Open gate
serialData		Y	Serial port transparent data array Note: This is optional , depending on the actual situation

serialChannel	Int	N	485 channel number 0 : A1, B1 have data input 1 : Data input for A2 and B2
data	String	N	485 input data , BASE64 encoded data
dataLen	Int	N	485 input data , data length before BASE64 encoding
shutoff	String	N	" The ok " represents the closing any other characters such as " NO " represents No action
snapnow	String	N	" Yes " means to capture any other characters such as " no " means no action

8 White list operations:

8.1 Server make a whitelist query:

Reservation: The HTTP interaction process is based on a request and a response. When the HTTP server need to do whitelist query, this query only can be sent when server receive a Heartbeat from camera, server send whitelist query as heartbeat answer content. .

8.1.1 HTTP server responds to camera heartbeat with a whitelist query data content

```
{
        "whiteList": {
                "queryNumList": 1 000
                "listPosition": 0
        }
}
```

Field name	Field Type	have to	description
whiteList		Y	This response requires the camera to return whitelisted data
queryNumList	Int	Y	Specifies the number of return , up to 1000 bar If it exceeds 1000 , it can be divided into multiple times
listPosition	Int	Y	Specify the starting position , to facilitate multiple queries

8.1.2 Camera push the white list data content to server:


```
{
        "Response_whiteList": {
                "totalList": 1000,
                "queryNumList": 1000,
                "listPosition": 0
                "data": [
                        {
                                "carnum": " ABC8888",
                                "startime": "20181029165012",
                                "endtime": "20181105165012"
                        },
                        ...
                ]
        }
        "serialno": "e10b2d6c8c07b422361457935b518642"
}
```


Field name	Field Type	description
Response_whiteList		Push results are whitelisted data
serialno	String	Camera serial number , camera unique identification (can be viewed on the web client- > device information- > camera serial number)
totalList	Int	Total number of current whitelists on the camera
queryNumList	Int	Number of whitelists pushed this time
listPosition	Int	Consistent with requested location
data		Note : When the total returned is 0 , there are no elements under the array
carnum	String	License plate number (String UTF-8)
startime	String	Whitelist start time
endtime	endtime	Whitelist deadline

8.2 Whitelist added operation, interactive data content description

Reservation: The HTTP interaction process is based on a request and a response. When the HTTP server need to do whitelist add operation, this operation only can be sent when server receive a Heartbeat from camera, server send whitelist add as heartbeat answer content..

8.2.1 Whitelist add operation

```
{
        "addWhiteList":
        {
                "add_data": [
                        {
                                "carnum": "ABC8888",
                                "startime": "20181029165012",
                                "endtime": "20181105165012"
                        },
                        ...
                ]
        }
}
```

Field name	Field Type	have to	description
addWhiteList		Y	The response requires the camera to add whitelist data
add_data		Y	Whitelist data array Note : added up to 1000 each operation, the total number if you want to add more than 1000 bar can be divided into multiple
carnum	String	Y	License plate number (UTF8 coding)
startime	String	Y	Whitelist start time
endtime	String	Y	Whitelist deadline

8.2.2 Camera answer to server whitelist add operation

```
{
        "Response_AddWhiteList": {
                "reponse": "ok"
                "serialno": "e10b2d6c8c07b422361457935b518642"
        }
}
```

Field name	Field Type	description
Response_AddWhiteList		Push result is whitelist result status
reponse	String	" Ok " means success, " no " means failure
serialno	String	Camera serial number , camera unique identification (can be viewed on the client- > device information- > camera serial number)

8.3 Delete all whitelists :

Explanation : The HTTP interaction process is based on a request and a response. When the HTTP server wants to initiate a request to delete all whitelists, it can wait until the camera heartbeat request is received, and the response is to delete all whitelists .

8.3.1 HTTP server response deletes all whitelist data content


```
{
        "deleteWhiteListAll": 1
}
```


Field name	Field Type	have to	description
deleteWhiteListAll	Int	Y	This response requires the camera to delete all whitelisted data

8.3.2 The camera pushes the result for deleting all whitelists


```
{
        "Response_DelWhiteListAll": {
                "reponse": "ok",
                "serialno": "e10b2d6c8c07b422361457935b518642"
        }
}
```


Field name	Field Type	description
Response_DelWhiteListAll		The result of the push is the status of deleting all whitelist results
reponse	String	" Ok " means success, " no " means failure
serialno	String	Camera serial number , camera unique identification (can be viewed on the client- > device information- > camera serial number)

8.4 Delete the specified whitelist record.

Explanation : The HTTP interaction process is based on a request and a response. When the HTTP server wants to initiate a request to delete the specified record from whitelist, it can wait until it receives a heartbeat request from the camera, and the response is to delete the specified whitelist.

8.4.1 HTTP server send require to camera delete the specified record from whitelist:


```
{
        "deleteWhiteList": {
                "del_data": [
                        {"carnum": " 粤 B88888"},
                ]
        }
}
```


Field name	Field Type	have to	description
deleteWhiteList		Y	The response requires the camera to delete the whitelist data for the specified license plate number
del_data		Y	record of license plate numbers to delete Note: The maximum number of record is 10 , more than 10 , divided multiple times
carnum	String	Y	License plate number (Chinese character code GB2312)

8.4.2 The camera pushes the result status data content for deleting the specified record of whitelist


```
{
        " Response_DeleteWhiteList ": {
                "reponse": "ok",
                "serialno": "e10b2d6c8c07b422361457935b518642"
```

```
            }
}
```

Field name	Field Type	description
Response_DeleteWhiteList		Push result is delete specified whitelist result status
reponse	String	" Ok " means success, " no " means failure
serialno	String	Camera serial number , camera unique identification (can be viewed on the client- > device information- > camera serial number)