# Network IP Camera Application Programming Interface (NIPCA)

Ver. 1.7

## Document History

| Version | Date | Comment |
|---|---|---|
| 0.99a | 2007-11-09 | Focus on the configuration settings. |
| 0.99b | 2007-11-20 | Add the Valid values. Add RS-485 commands.<br>Refine all document. |
| 0.99c | 2007-11-21 | Modify HTTP status codes, basic info, datetime DST, motion detection. |
| 0.99d | 2007-12-14 | 2.3: Modify HTTP status codes description.<br>Add descriptions and examples of parameters and values.<br>3.3.4 3.3.5: Modify flicker, autoexposure for sensor_info.cgi sensor.cgi.<br>3.4.8: Modify upnpav, upnpcp for upnp.cgi.<br>5.1.3: Remove redundant penable, tenable, zenable for ptz_step.cgi.<br>5.1.5: Add p,t,z position for ptz_preset_list.cgi.<br>6.1.2: Add keep_alive for notify_stream.cgi. |
| 0.99e | 2007-12-26 | Add the 3.5 event handling. |
| 1.0 | 2008-01-14 | 3.1.3 Modify the request to /users/verify.cgi<br>3.1.6 3.1.7 Modify method and offset.<br>Add the ID of the table dynamic DNS service providers. |
| 1.1 | 2008-01-31 | 2.1 Fix the POST Content-Type to application/x-www-form-urlencoded.<br>Add the ACS Stream Header. |
| 1.2 | 2008-02-21 | 2.1 Fix the POST parameters. |
| 1.3 | 2008-05-26 | 3.5.1 change the definition of macro block size.<br>3.3.4 add hue, autoexposure, autogainctrl<br>3.3.5 add hue, autogainctrl<br>3.6.8 add delaytime |
| 1.4 | 2008-05-16 | 3.1.1 get basic information: add ‘nipca’ item<br>3.1.1 add ‘videoout’ item<br>3.3.4 add ‘inputsize’ and ‘videooutformat’ item.<br>3.3.5 add ‘color’ item.<br>3.5.1 add more actions like cifs_rec, cifs_shot…<br>3.5.4 add more actions and the field prerecord and postrecord.<br>3.5.15, 3.5.16 add keep space item. |

| | | |
|---|---|---|
| | | 4.1.7 add get video stream of associated profile<br>6.1.1, 6.1.2 add mdv# item<br>7.1 change url of rtsp:<br>mpeg4 -> mp4<br>mjpeg -> jpeg<br>add /live# url |
| 1.5 | 2008-10-17 | 3.3.1, 3.3.2 add vprofileformat for video stream cgi 4.1.7<br>3.5.3, 3.5.4 add sw_input event<br>3.5.3, 3.5.4 change ‘actions’ and ‘action’ keyword to be ‘handlers’ and ‘handler’<br>5.3 add an software event trigger function<br>4.1.5, 4.1.6 revise the part of format description<br>5.2.1 unify the speed range to 1-10<br>4.1.8 add put audio upstream (two-way audio talk)<br>4.1.9 add H.264 streaming cgi url<br>Add 7.1.1-7.1.3 to support customized url entry of RTSP live stream.<br>3.4.1 add httpexternalport, rtspport and rtspexternalport. |
| 1.6 | 2009-02-17 | 5.3.1 add ‘trigger’ item to indicate that client want to turn this event on or off.<br>3.3.1, 3.3.2, 4.1.7 Refine the definition of vprofileformat<br>5.1.1 add item “customizedhome” to indicate whether camera support 5.1.9 function or not.<br>5.1.9 add PTZ home manifest command.<br>3.1.13 add ‘reset sensor to default configuration’ function<br>3.3.5, 3.3.6 add videoinformat for some video server models<br>3.3.14, 3.3.15 add IR LED setting functions<br>3.3.16, 3.3.17 add ICR setting functions |
| 1.7 | 2009-06-10 | 3.3.18, 3.3.19 add authentication control for live video and snapshot<br>3.1.1 add field ‘product’, ‘brand’ to basic information<br>5.1.10, 5.1.11 add Auto Patrol/Auto Pan for PTZ control function<br>5.1.12 Configure auot patrol preset sequence order.<br>3.6.5 add example html to submit firmware to ip camera.<br>4.1.2 add ‘profileid’ optional parameter<br>4.1.3 change this section to MPEG-4 elementary stream CGI.<br>4.1.5, 4.1.6, 4.1.9 refine the description of parameter ‘profileid’<br>4.1.7 modify the format of stream with profile M-JPEG (replaced by ACVS wrapped stream)<br>4.1.11 add audio profile stream CGI<br>8.2 Add two more frame type for ACVS header.<br>3.3.4, 3.3.5, 3.3.6 add sharpness<br>3.4.14, 3.5.15 add wireless strength function and wireless site survey<br>3.7 SD card operations added |

# Contents

# 1. Overview

Network IP Camera Access Application Programming Interface (NIPCA-API) is a HTTP-based API for the networks IP camera products. Users can write program easily by calling this API to access all functionalities provided by our IP cameras including: configuration, multimedia streaming data and the control facilities.

Except Streaming, the other groups of API use the same format in transporting HTTP-based message. We will describe the command HTTP request format in the next chapter.

As for the Streaming API, the output format of streaming API depends on different IP camera model. Here we only provide a general entry point to let the IP camera outputting streaming via a permanent HTTP connection.

We also provide the RTSP interface for our IP camera.

## 1.1. API versions

Though we provide a common API for all IP camera models, it may not apply to some old models which were produced before the first version of this API being published. We may also publish the further version of this API in the future. So there may be some difference between different versions of API. However, all our products shall provide the API version information with every firmware version of each model.

## 1.2. Valid values

The following valid values are used in this document:

| Values | Description |
|---|---|
| An integer | Any number between -2147483647 (-$2^{31}$-1) and 2147483647 ($2^{31}$-1). |
| m ... n | Any number between number m and number n. |
| # | A number equals or greater than 0 |
| A string | Any string encoded by UTF-8 |
| An IP address | A string limited to contain an IP address of the format xxx.xxx.xxx.xxx, where xxx is a number between 0 to 255. Example: 192.168.0.90 |
| A MAC Address | A string limited to contain a MAC address of the format xx:xx:xx:xx:xx:xx, where xx is a hexadecimal value. Example: 00:40:8c:cd:00:00 |
| A time | A string limited to contain a time of the format hh:mm:ss. Example: 23:01:14 |
| A date | A string limited to contain a date of the format yyyy-mm-dd. Example: 2004-02-16 |
| <value 1>, <value 2>, <value 3>, ... | Enumeration, only the given values are valid. |
| *<italic string>* | Every italic strings inside brackets including the brackets should be replaced by proper values. |

# 2. HTTP Interface

An HTTP-based protocol always includes two kinds of message, request and response. IP camera prepares a service to wait and accept TCP connection request with a specified port and to process the requests message from a user defined application. In this chapter, we will describe the common format of comprising all the different request and response messages. Although our camera also can support HTTP/1.0, we recommend that a request compliant with HTTP/1.1 may encounter fewer problems. You may also refer to the RFC 2616 HTTP/1.1.

## 2.1. Request messages

To query information of IP camera, use the syntax

```
GET http://<camera name>/<CGI-URL>?<parameter>=<value> HTTP/1.1<CRLF>
Authorization: Basic <basic-cookie><CRLF>
Host: <camera ip-adress><CRLF>
<CRLF>
```

where,

*`<CGI-URL>`* is a URL of a CGI. For example, get basic information is “/common/info.cgi”.

Authorization is optional for some CGIs.

*`<basic-cookie>`* is the base64 encoding of userid:password.

`<CRLF>` is Carriage Return and Line Feed `(\r\n)`.

To set values in the IP camera, you may use HTTP GET method, the syntax is

```
GET http://<camera name>/<CGI-URL>
?<parameter>=<value>[&<parameter>=<value>...] HTTP/1.1<CRLF>
Authorization: Basic <basic-cookie><CRLF>
Host: <camera ip-adress><CRLF>
<CRLF>
```

or HTTP method POST, the syntax is

```
POST http://<camera name>/<CGI-URL> HTTP/1.1<CRLF>
Authorization: Basic <basic-cookie><CRLF>
Host: <camera ip-adress><CRLF>
Content-Type: application/x-www-form-urlencoded<CRLF>
Content-Length: <body length><CRLF>
<CRLF>
<parameter>=<value>[&<parameter>=<value>]
```

where,

*`<body length>`* is the length of the entity body.

**<***`parameter`***>** will be described in the following chapters. Valid characters only include alphabets([A-Za-z]), digits([0-9]) and underline(_). There is no such restriction for *`<value>`*. The content part of the post message should be encoded with “url-encoding” function.

## 2.2. Response messages

While IP Camera receives request message from user, it will do the related action then output result as response message:

```
HTTP/1.1 <HTTP code> <HTTP text><CRLF>

Content-Type: text/plain<CRLF>

Content-Length: <body length><CRLF>

<CRLF>

<parameter>=<values><CRLF>

...
```

## 2.3. Response status codes

The API status codes are defined here.

*Table 1: HTTP status codes*

| HTTP code | HTTP text | Description |
|---|---|---|
| 200 | OK | The request has succeeded, but an application error may occur, please refer to each CGI response. |
| 400 | Bad Request | You used invalid or unsupported parameters or values for this IP camera. |
| 401 | Unauthorized | The request requires user authentication or the authorization was refused. |
| 404 | Not Found | This API is not supported for this IP camera. |
| 500 | Internal Error | The IP camera encountered an internal error or the API can not get the correct status. |
| 503 | Service Unavailable | The IP camera is unable to handle the request due to temporary overload. |

# 3. Configuration API

The CGIs under /config can only be accessed by administrators.

## 3.1. device information

### 3.1.1. get basic information

request:

GET /common/info.cgi

No authentication required.

response:

| Name | Value | Description |
|---|---|---|
| model | A string | model name |
| product | A string | product name of camera |
| brand | A string | brand name |
| version | A string | version number of firmware |
| build | A string | firmware build number |
| nipca | A string | version number of NIPCA supported (e.g. 1.2, 1.4) |
| name | A string | camera name |
| location | A string | camera location |
| macaddr | A MAC address | the MAC address |
| ipaddr | An IP address | IP address |
| netmask | An IP address | Subnet mask |
| gateway | An IP address | Default router/gateway used for connecting devices attached to different networks and networks segment. |
| wireless | yes, no | Only displayed if has wireless |
| ptz | P, T, Z | Only show supported Pan or Tilt or Zoom.<br>For example, ptz=P,T |
| inputs | # | The number of inputs |
| outputs | # | The number of outputs |
| speaker | yes, no | Only displayed if the IP camera has speaker. |
| videoout | yes, no | Only displayed if the IP camera has video out connector. |

### 3.1.2. get extended information

To identify each different camera model.

request:

GET /config/info_ex.cgi

response:

| Name | Value | Description |
|---|---|---|
| exid | A string | extended RDID |
| oemid | A string | OEMID |
| boardid | A string | Board ID |

### 3.1.3. quickly verify user

request:

GET /users/verify.cgi

response:

| Name | Value | Description |
|---|---|---|
| group | A string | the group name of the specified user in the HTTP Authorization header field. |

When the authorization fails, it will return HTTP/1.0 401 Unauthorized

### 3.1.4. get camera info

request:
GET /config/camera_info.cgi

response:

| Name | Value | Description |
|---|---|---|
| name | A string | camera name |
| location | A string | camera location |

### 3.1.5. set camera info

request:
GET/POST /config/camera_info.cgi

parameters:

see the above table.

response:
see the above table.

### 3.1.6. get system date and time

request:
GET /config/datetime.cgi

response:

| Name | Value | Description |
|---|---|---|
| method | 0, 1 | 0: disable ntpd<br>1: enable ntpd |
| timeserver | A host or IP address | NTP time server host name or IP address. |
| timezone | # | time zone ID, see Table Time zone |
| date | A date | yyyy-mm-dd |
| time | A time | hh:mm:ss |
| dstenable | no, yes | disable or enable the DST (Daylight Saving Time) |
| dstauto | no, yes | set DST automatically |
| offset | A time | The amount of time the clock should be turned back/forward (hh:mm), due to DST. |
| starttime | | The time when DST should be enabled in the format m.w.d/hh:mm:ss day d (0 ... 6) of week w (1 ... 5) of month m (1 ... 12).<br>d=0 is a Sunday. |
| stoptime | | Stop time when DST should be disabled in the same format as above. |

### 3.1.7. set system date and time

request:
GET/POST /config/datetime.cgi

parameters:

| Name | Value | Description |
|---|---|---|

| | | |
|---|---|---|
| method | 0, 1, 2 | 0: disable ntpd<br>1: enable ntpd<br>2: manual setting, requires date and time. |
| timeserver | A host or IP address | NTP time server host name or IP address. |
| timezone | 1 ... 83 | time zone ID, see Table Time zone |
| date | A date | yyyy-mm-dd |
| time | A time | hh:mm:ss |
| dstenable | no, yes | disable or enable the DST (Daylight Saving Time) |
| dstauto | no, yes | set DST automatically |
| offset | A time | The amount of time the clock should be turned back/forward (hh:mm), due to DST. |
| starttime | | The time when DST should be enabled in the format m.w.d/hh:mm:ss day d (0 ... 6) of week w (1 ... 5) of month m (1 ... 12).<br>d=0 is a Sunday. |
| stoptime | | Stop time when DST should be disabled in the same format as above. |

response:
see the 3.1.6 table.

## 3.2. users and groups

### 3.2.1. get users

request:
GET /config/user_list.cgi

parameters:

none or

name=<username>

response:
if no request parameter

| Name | Value | Description |
|---|---|---|
| users | # | The total number of users. |
| <username><br>... | <group name><br>... | For example, admin=admingrp<br>It will display all user names line by line. |

if request parameter is name

| Name | Value | Description |
|---|---|---|
| password | A string | base64 encoded password |
| group | A string | the group which this user belongs to. |

### 3.2.2. add or modify a user

request:
GET/POST /config/user_mod.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| name | A string | user name |
| password | A string | base64 encoded password |
| group | A string | the group which this user belongs to. |

response:
see the above table.

### 3.2.3. delete users

request:
GET/POST /config/user_del.cgi

parameters:

name =<username1>,<username2>, ...

You can delete many users at once.

response:
name=<username1>,<username2>, ...

### 3.2.4. get groups

request:
GET /config/group_list.cgi

parameters:

none or

name=<groupname>

response:
if no request parameter

| Name | Value | Description |
|---|---|---|
| groups | | The total number of groups. |
| <groupname><br>... | <user1>, ...<br>... | for example, admingrp=admin,root<br>It will display all group names and users line by line. |

if request parameter is name

| Name | Value | Description |
|---|---|---|
| user | <user1>, ... | the user names |
| privilege | ptz, outputs, speaker, mic, video, notify | the permissions list which this group has. |

### 3.2.5. add or modify group

request:
GET/POST /config/group_mod.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| name | A string | the group name |
| user | <user1>, ... | the user names |
| privilege | ptz, outputs, speaker, mic, video, notify | the permissions list which this group has. |

response:
see the above table.

### 3.2.6. delete a group

You can only delete a group which has no users belong to it.

request:
GET/POST /config/group_del.cgi

parameters:

name=<groupname>

response:
name=<groupname>

### 3.2.7. query supported privileges

request:
GET/POST /config/privilege_info.cgi

response:

| Name | Value | Description |
|---|---|---|
| privilege | ptz, outputs, speaker, mic, video, notify | available permissions. |

## 3.3. video, sensor, audio

### 3.3.1. query stream information

You can get supported parameter values for your IP camera. Some parameters are optional and not displayed if not supported in your IP camera.

request:

GET /config/stream_info.cgi

response:

| Name | Value | Description |
|---|---|---|
| videos | MPEG4, MJPEG, H264 | available video codecs list. for example, videos=MPEG4,MJPEG |
| audios | PCM, ADPCM, AMR, AAC | available audio codecs list. for example, audios=PCM |
| resolutions | <width>x<height>,... | available video resolutions list. for example, resolutions=640x480,320x240 |
| vbitrates | b1, b2, b3, ... | available bitrates (kbps) list for example, vbitrates=600,800,1000 |
| goplengths | | (optional) available GOP lengths list |
| framerates | | available frame rates list |
| qualities | | available MJPEG qualities list, only used if has MJPEG. |
| asamplerates | | audio sample rates (kHz) list |
| abitrates | | audio bitrates (kbps) list |
| Micvol | | avaiable mic volume range from v1 to v2. for example, micvol=0...100 |
| speakervol | | speaker volume range |
| vprofileformat | <Ver#> | The current version is '1.5': This value indicates whether camera support /video/video.cgi or not. Please also refer to 4.1.7 |
| vprofilenum | # | The total number of avaiable video streams. |
| vprofile# | <codec name> | video profile # (# is a number from 1 to the count of profiles) |
| vprofileurl# | | the URL for video profile # |
| vprofileres# | <width>x<height> | The resolution for video profile # |
| aprofilenum | # | The total number of avaiable audio streams. |
| aprofile# | <codec name> | audio profile # (# is a number from 1 to the count of profiles) |
| aprofileurl# | | the URL for audio profile # |

### 3.3.2. get video config

request:

GET /config/video.cgi

parameters:

profileid=<video profile number>

response:

| Name | Value | Description |
|---|---|---|
| vprofileformat | Ver# | The current version is '1.5': This value indicates whether camera support /video/video.cgi or not. Please also refer to 4.1.7 |
| profileid | # | profile number (# is a number from 1 to the count of profiles) |
| resolution | <width>x<height> | resolution |

| | | |
|---|---|---|
| bitrate | An integer | in kbit/s |
| codec | MPEG4, MJPEG | a video codec |
| goplength | An integer | the MPEG GOP length. |
| framerate | 1 ... 30 | a frame rate in fps |
| quality | # (0-100) | only used when the codec is MJPEG |

### 3.3.3. set video config

request:
GET/POST /config/video.cgi

parameters:

see the above table.

response:
see the above table.

### 3.3.4. sensors information

request:
GET /config/sensor_info.cgi

response: (only supported parameters are displayed)

| **Name** | **Value** | **Description** |
|---|---|---|
| brightness | b1 ... b2 | available brightness range |
| contrast | c1 ... c2 | available contrast range |
| saturation | s1 ... s2 | available saturation range |
| hue | h1 … h2 | available hue range |
| whitebalance | auto, fixed_indoor, fixed_outdoor, fixed_fluor, disabled | available white balances list |
| maxexposuretime | $m_1$ ... $m_2$ | a range of the maxexposuretime from $1/m_1$ to $1/m_2$ second. |
| backlightcomp | yes, no | has backlight compensation |
| noisereduction | off, low, high | a list of noise reduction level. |
| autoexposure | yes, no | Indicate if camera support auto exposure function |
| autogainctrl | yes, no | Indicate if camera support auto gain control. |
| inputsize | <width>x<height> | Dimension of sensor size. |
| videooutformat | auto: auto detect.<br>ntsc: NTSC<br>pal: PAL<br>pal-m:PAL M<br>pal-n: PAN-N | For the cameras which has an analog video output connector, this field indicates the format of the video signal |
| sharpness | s1 ... s2 | available sharpness range |

### 3.3.5. get sensors config

request:
GET /config/sensor.cgi

response: (only supported parameters are displayed.)

| **Name** | **Value** | **Description** |
|---|---|---|
| brightness | An integer | the brightness |
| contrast | An integer | the contrast |
| saturation | An integer | the saturation |

| | | |
|---|---|---|
| hue | An integer | the hue |
| whitebalance | auto, fixed_indoor, fixed_outdoor, fixed_fluor, disabled | the white balance |
| flicker | auto, 50, 60 | anti flicker. auto or 50 or 60 Hz. |
| autoexposure | yes, no | enable or disable the auto exposure |
| maxexposuretime | # | The divisor of maximum exposure time (1/# second). E.g. if the maximum exposure time is 1/10, then the value of this field is 10. |
| backlightcomp | yes, no | backlight compensation. Will make darker objects in the foreground appear clearer if the background is very bright. |
| noisereduction | off, low, high | noise reduction level. |
| mirror | off, on | disable/enable image flip horizontally |
| flip | off, on | disable/enable image flip vertically |
| autogainctrl | yes, no | enable or disable auto gain control function |
| color | yes, no | Select color mode or B/W mode |
| videoinformat | auto: auto detect.<br>ntsc: NTSC<br>pal: PAL<br>pal-m:PAL M<br>pal-n: PAN-N | For video server, the input analog video signal could be one of many video formats such as NTSC or PAL. To let video server recognize the format of video input signal, the sensor module should be configured to match the format. |
| sharpness | An integer | the sharpness |

## 3.3.6. set sensors config

request:

GET/POST /config/sensor.cgi

parameters:

see the above 2 tables to set the valid values.

response:

see the above table.

## 3.3.7. get audio config

request:

GET /config/audio.cgi

parameters:

profileid=<audio profile number>

response:

| Name | Value | Description |
|---|---|---|
| profileid | # | audio profile number (# is a number from 1 to the count of profiles) |
| codec | PCM, ADPCM, AMR, AAC | the audio codec |
| samplerate | An integer | The clock rate for the audio sampling. (in kHz) |
| channel | 1, 2 | the audio channel number. |
| bitrate | An integer | The output bitrate. (in kbit/s) |

## 3.3.8. set audio config

request:

GET/POST /config/audio.cgi

parameters:

see the above table.

response:
  see the above table.

### 3.3.9. get microphone

request:
  GET /config/mic.cgi

response:

| Name | Value | Description |
|---|---|---|
| enable | no, yes | microphone disable/enable |
| volume | | microphone volume. please refer 3.3.1 micvol |

### 3.3.10. set microphone

request:
  see the above table.

response:
  see the above table.

### 3.3.11. get speaker

request:
  GET /config/speaker.cgi

response:

| Name | Value | Description |
|---|---|---|
| enable | no, yes | speaker disable/enable |
| volume | | speaker volume. please refer 3.3.1 speakervol |

### 3.3.12. set speaker

request:
  GET/POST /config/speaker.cgi

  parameters:

  see the above table.

response:
  see the above table.

### 3.3.13. reset sensor to default configuration

request:
  GET/POST /config/sensor_reset.cgi

  parameters:

  reset=go

response:

| Name | Value | Description |
|---|---|---|
| reset | yes, fail | the result of sensor reset |

### 3.3.14. get IR LED setting

request:
GET /config/irled.cgi

response:

| Name | Value | Description |
|---|---|---|
| mode | on,<br>off,<br>auto | indicate whether IR LED is on, off, or auto. |

### 3.3.15. set IR LED setting

request:
see the above table.

response:
see the above table.

### 3.3.16. get ICR(Infrared Cut filter Removal) setting

request:
GET /config/icr.cgi

response:

| Name | Value | Description |
|---|---|---|
| mode | on(day mode),<br>off(night mode),<br>auto,<br>schedule | indicate whether icr is on or off |
| starttime | A time | start time of schedule (in 24hr format "hh:mm", only when mode=schedule)<br>for example 07:30 means 7:30 am.<br>for example 19:30 means 7:30 pm. |
| endtime | A time | end time of schedule (in 24hr format "hh:mm", only when mode=schedule)<br>for example 07:30 means 7:30 am.<br>for example 19:30 means 7:30 pm. |

### 3.3.17. set ICR setting

request:
see the above table.

response:
see the above table.

### 3.3.18. get stream authentication setting

request:
GET /config/stream_auth.cgi

response:

| Name | Value | Description |
|---|---|---|
| livevideo | on | indicate whether it needs authentication to get live video stream. |

| | | |
|---|---|---|
| | off | |
| snapshoturl | on<br>off | indicate whether it needs authentication to get a snapshot. |

### 3.3.19. set stream authentication setting

request:
see the above table.

response:
see the above table.

Note: If the value of 'livevideo' is off, then the authentication for snapshot url will be turned off automatically.

## 3.4. network

### 3.4.1. get network config

request:
GET /config/network.cgi

response:

| Name | Value | Description |
|---|---|---|
| dhcp | off, on | disable/Enable dynamic IP address assignment |
| ip | An IP address | IP address |
| netmask | An IP address | subnet mask |
| gateway | An IP address | default gateway |
| dns1 | An IP address | primary DNS server |
| dns2 | An IP address | secondary DNS server |
| pppoe | off, on | use PPPoE |
| pppoeuser | A string | PPPoE user name |
| pppoepass | A string | PPPoE password |
| ddns | off, on | disable/enable dynamic ddns service |
| ddnsprovider | | ID of the provider, see Table dynamic DNS service providers |
| ddnshost | A string | ddns host name |
| ddnsuser | A string | ddns user name |
| ddnspass | A string | ddns password |
| upnp | off, on | disable/enable UPnP |
| httpport | 1 ... 65535 | TCP port number for HTTP |
| httpexternalport | 1 ... 65535 | The external port number for upnp NAT router to map the HTTP service port of camera |
| rtspport | 1 ... 65535 | The port number of RTSP service |
| rtspexternalport | 1 ... 65535 | The external port number for upnp NAT router to map the RTSP service port of camera |

### 3.4.2. set network config

request:
GET/POST /config/network.cgi

parameters:

see the above table.

response:
see the above table.

### 3.4.3. get PPPoE

request:
GET /config/pppoe.cgi

response:

| Name | Value | Description |
|---|---|---|
| pppoe | off, on | disable/enable PPPoE |
| user | | PPPoE user name |
| pass | | PPPoE password |

### 3.4.4. set PPPoE

request:
GET/POST /config/pppoe.cgi

parameters:

see the above table.

response:
see the above table.

### 3.4.5. get DDNS providers

request:
GET /config/ddnsproviders.cgi

response:

| Name | Value | Description |
|---|---|---|
| num | # | The number of dynamic dns service providers. |
| providers | | available providers ID list |

### 3.4.6. get DDNS settings

request:
GET /config/ddns.cgi

response:

| Name | Value | Description |
|---|---|---|
| ddns | off, on | disable/enable dynamic DNS service |
| provider | | ID of the provider, see Table dynamic DNS service providers |
| host | | DDNS host name |
| user | | DDNS user name |
| pass | | DDNS password |

### 3.4.7. set DDNS

request:
GET/POST /config/ddns.cgi

parameters:

see the above table.

response:

see the above table.

### 3.4.8. get upnp information

request:

GET /config/upnp.cgi

response:

| Name | Value | Description |
|---|---|---|
| upnpav | off, on | disable/enable UPnP AV server. |
| upnpcp | off, on | disable/enable UPnP CP port forward |

### 3.4.9. set upnp information

request:

GET /config/upnp.cgi

parameters:

see the above table.

response:

see the above table.

### 3.4.10. get TCP port number for HTTP

request:

GET /config/httpport.cgi

response:

| Name | Value | Description |
|---|---|---|
| httpport | 1 ... 65535 | TCP port number for HTTP |

### 3.4.11. set TCP port number for HTTP

request:

GET/POST /config/httpport.cgi

see the above table.

response:

see the above table.

### 3.4.12. get system wireless

request:

GET /config/wireless.cgi

response:

| Name | Value | Description |
|---|---|---|
| enable | off, on | disable/enable wireless |
| mode | managed,<br>ad-hoc | The type of wireless network to associate with, managed (using an access point) or ad-hoc (not using an access point). |
| essid | A string | ESSID |
| chpatterns | A string | The pattern of available wireless channels. read-only. 1111000011110000 means channel 1,2,3,4,9,10,11,12 are available. |
| channel | 1 ... 16 | wireless channel |

| | | |
|---|---|---|
| auth | open,<br>shared,<br>WPA-PSK, WPA2-PSK | Authentication method. open system, shared key , WPA-PSK or WPA2-PSK |
| encryption | none,<br>WEP,<br>TKIP, AES | when auth is open: none, WEP.<br>when auth is shared: WEP<br>when auth is WPA-PSK or WPA2-PSK: TKIP, AES |
| format | hex,<br>ASCII | only used for WEP |
| keylength | 64,<br>128 | WEP key length (bits) |
| activekey | 1 ... 4 | Which WEP key to use when transmitting. |
| key1 | | The keys must match the keys in the wireless access point. They could either be in hex format or in ASCII.<br>Hex: the string must be exactly 10 hex characters for 64-bit WEP and 26 hex characters for 128-bit WEP. (Hex chars are 0123456789ABCDEF)<br>ASCII: The string must be exactly 5 characters for 64-bit WEP and 13 characters for 128-bit WEP. |
| key2 | | |
| key3 | | |
| key4 | | |
| passphrase | | WPA passphrase |

## 3.4.13. set system wireless

request:
GET/POST /config/wireless.cgi

parameters:

see the above table.

response:
see the above table.

## 3.4.14. get current wireless connection condition

request:
GET /config/wlansignal.cgi

response:

| Name | Value | Description |
|---|---|---|
| signal | 0…100 | Current wireless channel signal strength |

## 3.4.15. do wireless site survey

request:
GET /config/wlansurvey.cgi

response: (1 site)

| Name | Value | Description |
|---|---|---|
| ssid | A string | SSID |
| signal | 0…100 | The signal strength indicator of wireless AP. |
| mode | Ad-hoc<br>infrastructure | Wireless mode |
| channel | 1 ... 16 | wireless channel |
| auth | open,<br>shared,<br>WPA-PSK, WPA2-PSK | Authentication method. open system, shared key , WPA-PSK or WPA2-PSK |
| encryption | none, | when auth is open: none, WEP. |

| | | |
|---|---|---|
| | WEP,<br>TKIP, AES | when auth is shared: WEP<br>when auth is WPA-PSK or WPA2-PSK: TKIP, AES |

note:
Each wireless AP (access point) found has several attributes such as the above table. 'ssid' is the first attribute of any one wireless AP. The camera output these attributes of all found wireless AP in sequence.

## 3.5. event handling

### 3.5.1. get motion detection

There are 2 possible types of motion dection dependent on your IP camera hardware.

request:
GET /config/motion.cgi

response:
macro block type:

| Name | Value | Description |
|---|---|---|
| enable | no, yes | disable/enable motion detection |
| mbmask | A hex string | The macro block mask hex string of the native screen resolution which is calculated linearly from left to right then top to bottom. The size of one macro block depends on the video resolution. However, no matter the resolution of video is, the number of macro block is always 40x30. That is there is 40 block in extension of the width of video and 30 block in height. |
| sensitivity | 0 ... 100 | sensitivity |

window type:

| Name | Value | Description |
|---|---|---|
| totalnum | # | total motion detection window numbers. read-only. |
| sensitivity | 0 ... 100 | sensitivity |
| enable# | no, yes | disable/enable motion detection window # |
| mdw# | A string | motion detection window # in the format x,y,w,h<br>x,y is the coordinate. the 0,0 means the left top position.<br>w,h is the width and height of the window. |

### 3.5.2. set motion detection

request:
GET/POST /config/motion.cgi

parameters:

see the above table.

response:
see the above table.

### 3.5.3. query event handlers information

request:
GET /config/event_info.cgi

response:

| Name | Value | Description |
|---|---|---|
| num | # | Total number of event handlers |
| name | <event handler1>, ... | The names of available event handlers |
| events | schedule, md#, input#, storagefail, storagefull, sw_input | available event types list.<br>Note: sw_input is a special event type. It is a event triggered by remote client software via a specific cgi function. Please also refer to 5.3. |
| handlers | cifs_rec, cifs_shot, ftp_rec, ftp_shot, sd#_rec, sd#_shot, mail_rec, mail_shot, output# | available actions list.<br>Note: sd#_rec, sd#_shot are recording and snapshot to local disk where # is the serial number of local disk. |

## 3.5.4. get an event handler

request:

GET /config/event.cgi

parameters:

name=<event handler name>

response:

| Name | Value | Description |
|---|---|---|
| name | A string | The name of the event handler. |
| event | schedule,<br>md#,<br>input#,<br>storagefail,<br>storagefull<br>sw_input | schedule<br>md# is motion detection 1, 2, 3, 4<br>input# is digital input 1, 2, 3, 4<br>storage fail<br>storage full<br>software triggered event. |
| handler | cifs_rec,<br>cifs_shot,<br>ftp_rec,<br>ftp_shot,<br>sd#_rec,<br>sd#_shot,<br>mail_rec,<br>mail_shot,<br>output# | Record film to a CIFS/SMB server<br>Snapshot to a CIFS/SMB server<br>Record film to a FTP server<br>Snapshot to a FTP server<br>Record film to local disk<br>Snapshot to local disk<br>Record media clip to a SMTP server<br>Snapshot to a SMTP server<br>output# is digital output 1, 2, 3, 4 |
| profileid | # | Only for recording and snapshot actions. Assign which profile of stream should be recorded or snapshot. |
| prerecord | <time> | Time period before event trigger. The unit of this value is second. For example, if prerecord=10, then the designated action will start before the event being triggered.<br>Note: this field is only applied with recording actions like: cifs_rec, ftp_rec, sd#_rec, mail_rec, and so on… |
| postrecord | <time> | Time period after event trigger. The unit of this value is second. For example, if postrecord=10, then the designated action will be stopped after the end of triggered event.<br>Note: this field is only applied with recording actions like: cifs_rec, ftp_rec, sd#_rec, mail_rec, and so on… |
| schedule | always,<br>never,<br><schedule profile id> | the schedule of the event handler |
| minitriggerinterval | A time string | Time interval between triggers in format “hh:mm:ss”, any triggers that occur during the interval are ignored. Max interval is 23:59:59. 00:00:00 means the interval is disabled. |

## 3.5.5. set event handler

request:

GET/POST /config/event.cgi

parameters:

see the above table.

response:
see the above table.

### 3.5.6. delete an event handler

request:
GET/POST /config/event_del.cgi

parameters:

name=<event handler name>

response:

name=<event handler name>

### 3.5.7. list schedule profiles

request:
GET /config/schedule_list.cgi

response:

| Name | Value | Description |
|---|---|---|
| num | # | Total number of schedule profiles. |
| id | <schedule profile id>,... | the schedule profile id |

### 3.5.8. get schedule profile

request:
GET /config/schedule.cgi

id=<schedule profile id>

response:

| Name | Value | Description |
|---|---|---|
| id | A string | schedule profile id |
| weekdays | A string | Pattern of weekdays.<br>for example, 0111110 means Mon Tue Wed Thu Fri.<br>for example, 1000000 means Sunday. |
| starttime | A time | start time in 24hr format "hh:mm".<br>for example 07:30 means 7:30 am.<br>for example 19:30 means 7:30 pm. |
| duration | A time | How long the schedule in format "hours:minute". Max 168:00 hours. |

### 3.5.9. set or add schedule profile

request:
GET/POST /config/schedule.cgi

parameters:

see the above table.

response:

see the above table.

## 3.5.10. delete a schedule profile

request:
GET/POST /config/schedule_del.cgi

parameters:

id=<schedule profile id>

response:
id=<schedule profile id>

## 3.5.11. get FTP action

request:
GET /config/action_ftp.cgi

response:

| Name | Value | Description |
|---|---|---|
| host | | FTP server host name or IP address |
| port | 1 ... 65535 | FTP server port number |
| path | | FTP initial path |
| user | | FTP login user name |
| pass | | FTP login password |
| passive | no, yes | active mode or passive mode |

## 3.5.12. set FTP action

request:
GET/POST /config/action_ftp.cgi

parameters:

see the above table.

response:
see the above table.

## 3.5.13. get mail action

request:
GET /config/action_mail.cgi

response:

| Name | Value | Description |
|---|---|---|
| sender | An email address | the sender |
| to | An email address | email addresses separated by comma (,) |
| host | | SMTP server host name or IP address |
| port | | TCP port number |
| user | | user name |
| pass | | password |

## 3.5.14. set mail action

request:
GET/POST /config/action_mail.cgi

parameters:

see the above table.

response:
see the above table.

## 3.5.15. get CIFS/SMB action

request:
GET /config/action_cifs.cgi

response:

| Name | Value | Description |
|---|---|---|
| user | <user name>, NULL | user name. NULL means no user authentication. |
| pass | <password>, NULL | password. NULL means no user authentication. |
| path | | CIFS/SMB mount point.<br>for example, //192.168.1.5/public |
| split | size,<br>time | split file based on file size<br>or split file based on record time |
| maxsize | | maximum file size in KB(=1000Bytes) to be recorded for one file |
| maxtime | | maximum time in “hours:minutes” to be recorded for one file |
| full | stop,<br>del | stop recording when disk full.<br>delete oldest file when disk full. |
| keepspace | | the size of kept space from being recorded |

## 3.5.16. set CIFS/SMB action

request:
GET/POST /config/action_cifs.cgi

parameters:

see the above table.

response:
see the above table.

## 3.6. system tools

### 3.6.1. get digital input/output

request:
GET /config/io.cgi

response: (only supported inputs and outputs are displayed)

| Name | Value | Description |
|---|---|---|
| in1 | off, on | Digital input set 1 |
| in2 | off, on | Digital input set 2 |
| in3 | off, on | Digital input set 3 |
| in4 | off, on | Digital input set 4 |
| out1 | off, on | Digital output set 1 |
| out2 | off, on | Digital output set 2 |
| out3 | off, on | Digital output set 3 |
| out4 | off, on | Digital output set 4 |

### 3.6.2. set digital output

You can only set the available digital outputs, inputs are read-only.

request:
GET/POST /config/io.cgi

parameters:

| | | |
|---|---|---|
| out1 | off, on | Digital output set 1 |
| out2 | off, on | Digital output set 2 |
| out3 | off, on | Digital output set 3 |
| out4 | off, on | Digital output set 4 |

response:
see the above table.

### 3.6.3. get LED

request:
GET /config/led.cgi

response:

| Name | Value | Description |
|---|---|---|
| led | on, off | enable or disable the special purpose LED. |

### 3.6.4. set LED

request:
GET/POST /config/led.cgi

parameters:

see the above table.

response:
see the above table.

### 3.6.5. firmware upgrade

request:

POST /config/firmwareupgrade.cgi

The file content is provided in the HTTP body according to the format given in RFC 1867. The body is created automatically by the browser if using HTML form with input type "file".

```
Example:
POST /config/firmwareupgrade.cgi HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AsCg5y\r\n
Content-Length: <content length>\r\n
\r\n
--AsCg5y\r\n
Content-Disposition: form-data; name="update.bin"; filename="update.bin"\r\n
Content-Type: application/octet-stream\r\n
\r\n
<firmware file content>
\r\n
--AsCg5y--\r\n
```

response:

| Name | Value | Description |
|---|---|---|
| upgrade | ok, fail | the upgrade was successful or fail |

Note:
You can use web browser (e.g. Microsoft Internet Explorer or FireFox) to transport firmware to IP camera. To do this, you should write a HTML file with a form architecture to post firmware file to camera. For example:

```
<!-- saved from url=(0022)http://internet.e-mail -->
<html>

<head>
<script language="JavaScript" type="text/javascript">
function sendUpdate()
{
   var updateForm = document.updateForm;
   document.updateForm.action = "http://" + camip.value + "/config/firmwareupgrade.cgi";
   updateForm.submit();
}
</script>
</head>

<body>
Input camera ip (ex. 192.168.1.1): <input name="camip" type="text" id="camip" value=""/>
<form enctype="multipart/form-data" method="post" action="" name="updateForm">
   Choose firmware file: <input name="upload" type="file" id="upload" value=""/>
   and click
<input name="submit6" value="commit" type="button" onclick="sendUpdate()"/>
</form>
</body>

</html>
```

### 3.6.6. reboot the camera

request:

GET/POST /config/system_reboot.cgi

parameters:

reboot=go

response:

| Name | Value | Description |
|---|---|---|
| reboot | yes, fail | the reboot was successful or fail |

## 3.6.7. reset all configurations to the factory default

request:

GET/POST /config/system_reset.cgi

parameters:

reset=go

response:

| Name | Value | Description |
|---|---|---|
| reset | yes, fail | the reset was successful or fail |

## 3.6.8. get RS-485 settings

request:
GET /config/rs485.cgi

response:

| Name | Value | description |
|---|---|---|
| enable | no, yes | disable/enable RS-485 |
| proto | custom, Dyna, Lilin, Lilin2, PelcoD, PelcoP | protocol type |
| devid | | device ID of the RS-485 slave device.<br>Dyna: 1 ... 223<br>Lilin: 1 ... 64<br>Lilin2: 0 … 255<br>PelcoD: 1 ... 255<br>PelcoP: 1 ... 32<br>custom: not applicable |
| baudrate | 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 | custom baud rate |
| databits | 7, 8 | custom data bits |
| parity | None, Even, Odd | custom parity |
| stopbits | 1, 2 | custom stop bits |
| home | | custom home command |
| up | | custom up command |
| down | | custom down command |
| left | | custom left command |
| right | | custom right command |
| stop | | custom stop command |
| stoppattern | A string | whether to use the custom stop command for custom command 1, 2, 3, 4.<br>0101 means custom command 2 and 4 with stop command. |
| cmdname1 | | custom command1 name |
| cmdname2 | | custom command2 name |
| cmdname3 | | custom command3 name |
| cmdname4 | | custom command4 name |
| cmdstr1 | | custom command1 string |
| cmdstr2 | | custom command2 string |
| cmdstr3 | | custom command3 string |
| cmdstr4 | | custom command4 string |
| delaytime | | the interval of time between two consecutive command string being executed(in millisecond, for example 300ms). |

## 3.6.9. set RS-485 settings

request:
GET/POST /config/rs485.cgi

see the above table.

response:
see the above table.

## 3.7. SD card operation

### 3.7.1. get information of SD card

request:
GET /config/sdcard.cgi

response: (only supported inputs and outputs are displayed)

| Name | Value | Description |
|---|---|---|
| status | available, unavailable | SD Card is available? |
| total | integer | Total size of SD Card in KBytes |
| used | integer | Used size of SD Card in KBytes |
| free | integer | Free size of SD Card in KBytes |
| picture | integer | How many snapshot files (.jpg) |
| video | integer | How many recorded files (.avi) |

### 3.7.2. get detail information of SD card

request:
GET /config/sdcard.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| type | picture, video | See detailed information of indicated type |

response:

| Name | Value | Description |
|---|---|---|
| status | ready, invalid, write_protected | SD Card is available? |
| num | integer | How many folder |
| <folder name 1>,<folder name 2>,…<folder name n> | string | How many files in this folder |

### 3.7.3. format SD card

request:
GET /config/sdcard_format.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| format | go | Do format or query SD status.<br>Note:<br>If field 'format' isn't given or SD card is being formatted, the ip camera reports the status of formatting |

response:

| Name | Value | Description |
|---|---|---|
| status | none,<br>formatting,<br>ready,<br>unformatted | Status of SD formatting |

### 3.7.4. list files of SD card

request:
GET /config/sdcard_list.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| type | picture, video | Which type of files would like to see |
| date | (see description) | Indicate which files of date you want to get. Format: "yyyymmdd" |
| page | integer | This command would list files of a page. You can indicate which list of page you would like to see. |
| pagesize | integer | How many files in a page |

response:

| Name | Value | Description |
|---|---|---|
| status | ready,<br>invalid,<br>write_protected | Status of SD Card, if SD Card is not available, the value would be 'unavailable', or it would<br>show 'available' |
| type | picture, video | Which type of files would like to see |
| date | (see description) | Indicate which files of date you want to get. Format: "yyyymmdd" |
| page | integer | This command would list files of a page. You can indicate which list of page you would<br>like to see.<br>For example:<br>If I have 40 files in the 'date' folder and would like to list the files in page 1 where I<br>assume the number of files in each page is 20. The request url may be:<br>GET<br>/config/sdcard_list.cgi?type=picture&date=20080704&page=1&pagesize=20<br>Where:<br>type=picture: list the files in jpg format<br>date=20080229: list the files which snapshoot at 2008/2/29<br>page=1: list the files in page 1<br>pagesize=1: indicate that there are 20 files in each page<br>And the server would response the fist 20 files which snapshoot at 2008/2/29 |
| pagesize | integer | How many files in a page. See more details in previous parameter. |
| total_file | integer | Total number of these files |
| total_page | integer | Total page of these files |
| num | integer | Number of files in indicated page |
| file_<filename> | String | Attributes of this file, format is <timestamp>,<recording type>,<size><br>where:<br>timestamp: yyyy-mm-dd HH:MM:SS<br>recording type: d (digital input) or m (motion)<br>size: size of file |

## 3.7.5. download files of SD card

request:

GET /config/sdcard_download.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| type | picture, video | Which type of files would like to see |
| file | string | The file name could be got in command /config/sdcard_list.cgi. |

response (when file is available):
```
HTTP/1.0 200 OK<CRLF>
Content-Type: application/octet-stream<CRLF>
Content-Length: <size of file><CRLF>
<CRLF>
<Binary data of file>
```

response (when file is not available):

| Name | Value | Description |
|---|---|---|
| num | integer | Number of files which in deleting list |
| file_<filename> | string | The status of deleting action of indicated file, <filename>.<br>0: File is successfully deleted<br>1: File is not exists<br>2: File is not deleted |

## 3.7.6. delete files of SD card

request:

GET /config/sdcard_delete.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| type | picture, video | Which type of files would like to see |
| file | <file 1>[,<file 2>, …] | File list which would be deleted. The file name could be get in command /config/sdcard_list.cgi. |

response:

| Name | Value | Description |
|---|---|---|
| num | integer | Number of files which in deleting list |
| file_<filename> | string | The status of deleting action of indicated file, <filename>.<br>0: File is successfully deleted<br>1: File is not exists<br>2: File is not deleted |

# 4. Streaming

## 4.1. Live streaming URL

### 4.1.1. get a JPEG image

Returns a JPEG image with the default resolution and compression as defined in the configuration.

request:

GET /image/jpeg.cgi

response:

```
HTTP/1.0 200 OK\r\n
Content-Type: image/jpeg\r\n
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
```

### 4.1.2. get motion JPEG video stream

Returns a multipart image stream with the default resolution and compression as defined in the configuration. The content type is “multipart/x-mixed-replace” and each image ends with a boundary string <boundary>.

request:

GET /video/mjpg.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # | optional. If omitted, the url will output one of stream profile that match the assign format (motion JPEG). |

response:

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary>\r\n
\r\n
--<boundary>\r\n
Content-Type: image/jpeg\r\n
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
--<boundary>\r\n
Content-Type: image/jpeg\r\n
Content-Length: <image size>\r\n
\r\n
<JPEG image data>\r\n
--<boundary>\r\n
```

### 4.1.3. get MPEG-4 elementary video stream

Returns a MPEG-4 elementary stream with assigned profile id defined in the configuration. The content type is “video/MP4V-ES” please refer to INAN MIME Media Types.

request:

GET /video/MP4V-ES.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # | optional. If omitted, the url will output one of stream profile that match the assign format (MPEG4 elementary stream). |

response:
```
HTTP/1.0 200 OK\r\n
Content-Type: video/MP4V-ES\r\n
\r\n
<MPEG-4 ISO/IEC 14496-2 elementary stream>
```

### 4.1.4. get the video or audio stream session ID (Obsolete)

request:

GET /video/sessionid.cgi

response:

sessionid=<a session id>

### 4.1.5. get MPEG-4 video stream

Return the MPEG-4 video stream. The video data header please refer to the ACS Stream Header.

request:

GET /video/ACVS.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # | optional. If omitted, the url will output one of stream profile that match the assign format (ACVS/MPEG4 stream). |
| sessionid (obsolete) | <a session id> | optional. product dependent. |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: video/ACVS\r\n
\r\n
<ACAS Video Stream Data>

Where <ACAS Video Stream Data> is defined as below:
<ACS_VideoHeader>
<MPEG4 Raw Data>
<ACS_AudioHeader>
<MPEG4 Raw Data>
...
```

<ACS_VideoHeader> is defined in **8.2 Advanced ip-Camera Stream(ACS) Header**.
<MPEG4 Raw Data> is raw data of MPEG4 video stream.

### 4.1.6. get audio stream

The audio data header please refer to the ACS Stream Header.

request:

GET /audio/ACAS.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # | optional. If omitted, the url will output one of stream profile that match the assign format (ACAS/PCM stream). |
| sessionid (obsolete) | <a session id> | optional. product dependent. |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: audio/ACAS\r\n
\r\n
<ACAS Audio Stream Data>
```

Where <ACAS Audio Stream Data> is defined as below:
<ACS_AudioHeader>
<Audio Raw Data>
<ACS_AudioHeader>
<Audio Raw Data>
...

<ACS_AudioHeader> is defined in **8.2 Advanced ip-Camera Stream(ACS) Header**.
<Audio Raw Data> is raw data of audio stream. The format of this data depends on <ACS_AudioHeader>.

### 4.1.7. get profile video stream

Return the video stream associated with a specific profile. The video stream format depends on the compression type of video in that profile. Please read note below.

request:
GET /video/video.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # (1 to the count of video profile) | If omitted, the url will output the stream of default profile (profile id = 1). |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: video/ACVS\r\n
\r\n
<video streaming data>
```

note:
`<video streaming data>`:
If the compression type of the designated profile is motion-JPEG(MJPEG), the stream format is different with as multipart format stream. The format of video stream with profile motion-JPEG is wrapped by ACVS header.
On the other hand, if the compression type is MPEG4 (or H.264 or other advanced compression methods), the output format follows the standard of ACVS format (See Appendix 8.2).
This url is only available while the value of item ‘vprofileformat’ in 3.3.1 or 3.3.2 is equal to or greater than 1.0

### 4.1.8. put audio upstream (two-way audio talk)

There are two requests to use this service. One of the requests is “verification request”, the other is called “uploading request”.

While uploading audio data from client to camera server, the client may run into some situations instead of successfully keeping uploading audio data. For example, if another client has been uploading audio data, server will disconnect the connection after client starting uploading audio. On the other hand, if client send command with wrong authentication information, the server will also reject the request from client. So a client should use verification request to test if it has the correct authentication information before uploading audio stream. In other word, if a user has past the verification request but it still got disconnecting after uploading request because of the other connect existing.

#### Verification request:

Request object:
```
HEAD /dev/speaker.cgi?client=<MAC address of client side>
```

Request header:
```
Authorization: Basic <base64 encode(username:password)>\r\n
Content-Type: audio/ACAS\r\n
Content-length: 0\r\n
\r\n
```

Response of verification request:
If the authorization is verified, the camera should return 200 OK to indicate client side can keep uploading request:

```
HTTP/1.0 200 OK\r\n
```

If the authorization is failed, the camera would return HTTP error code to indicate client side should stop the uploading request, for example:

```
HTTP/1.0 401 Unauthorized\r\n
```

**Uploading request:**

Request object:

```
POST /dev/speaker.cgi?client=<MAC address of client side>
```

Request header:

```
Authorization: Basic <base64 encode(username:password)>\r\n
Content-Type: audio/ACAS\r\n
Content-length: 4\r\n
Connection: Keep-Alive\r\n
\r\n
```

Request body:

```
<Random 4CC>
<AAH>
<1K audio data>
<AAH>
<1K audio data>
<AAH>
<1K audio data>
...
```

**Where:**

```
<Random 4CC>: 4 byte random code.

<AAH>: the header of AAH defined as follow:
typedef struct _ACS_AudioHeader
{
unsigned long ulHdrID;                //Header ID  = 0xF6010000
unsigned long ulHdrLength;          // sizeof(ACS_AudioHeader)
unsigned long ulDataLength;         // audio data length
unsigned long ulSequenceNumber; // sequence numger
unsigned long ulTimeSec;              //sample time stamp
unsigned long ulTimeUSec;           // sample time stamp
unsigned long ulDataCheckSum;    // not used…
unsigned short usFormat;              //  0x00000010  S16 LE
unsigned short usChannels;           // 1 channel
unsigned short usSampleRate;       // 8000   hz
unsigned short usSampleBits;        // 16   bits
unsigned long ulReserved;            //
}ACS_AudioHeader, *PACS_AudioHeader;

<1K audio data>: audio data acquired by client side in the format specified by <AAH>
header
```

Response of uploading request:

There are no response for this request.

## 4.1.9. get H.264 video stream

Return the H.264 video stream. The video data header please refer to the ACS Stream Header.

request:

GET /video/ACVS-H264.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # | optional. If omitted, the url will output one of stream profile that |

| | | |
|---|---|---|
| | | match the assign format (ACVS/H.264 stream). |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: video/ACVS\r\n
\r\n
<ACAS Video Stream Data>

Where <ACAS Video Stream Data> is defined as below:
<ACS_VideoHeader>
<H.264 Raw Data>
<ACS_AudioHeader>
<H.264 Raw Data>
...

<ACS_VideoHeader> is defined in 8.2 Advanced ip-Camera Stream(ACS) Header.
<MPEG4 Raw Data> is raw data of MPEG4 video stream.
```

### 4.1.10. get audio WAVE stream

Return the audio stream in .WAV format.

request:

GET /audio/x-wav.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| sec | # | Duration of audio streaming.<br>0: (default) indicate maximum duration<br>1-120000: duration in second |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: audio/x-wav\r\n
\r\n
<wave format data>

Where <wave format data> is a standard Microsoft wave file format. Please refer to MIME:
audio/x-wav.
```

### 4.1.11. get profile audio stream

Return the audio stream associated with a specific profile. The audio stream format depends on the compression type of audio in that profile. Please read note below.

request:

GET /audio/audio.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| profileid | # (1 to the count of audio profile) | If omitted, the url will output the stream of default profile (profile id = 1). |

response:
```
HTTP/1.0 200 OK\r\n
Content-type: audio/ACAS\r\n
\r\n
<audio streaming data>
```

note:

`<audio streaming data>`:

The audio stream is wrapped by ACAS header.

# 5. Camera Control API

## 5.1. Remote control

### 5.1.1. query PTZ information

request:

GET /config/ptz_info.cgi

response: (only supported parameters are displayed.)

| Name | Value | Description |
|---|---|---|
| pmax | An integer | maximum position of pan |
| pmin | An integer | minimum position of pan |
| tmax | An integer | maximum position of tilt |
| tmin | An integer | minimum position of tilt |
| zmax | An integer | maximum position of zoom |
| zmin | An integer | minimum position of zoom |
| customizedhome | no, yes | to indicate whether camera can use “customized home” function. Please refer to section 5.1.9 |

### 5.1.2. get the current PTZ position

request:

GET /config/ptz_pos.cgi

response: (only supported parameters are displayed.)

| Name | Value | Description |
|---|---|---|
| p | An integer | the pan position |
| t | An integer | the tilt position |
| z | An integer | the zoom position |

### 5.1.3. get the PTZ movement size in a step

request:

GET /config/ptz_step.cgi

response: (only supported parameters are displayed.)

| Name | Value | Description |
|---|---|---|
| pstep | An integer | pan movement size in a step |
| tstep | An integer | tilt movement size in a step |
| zstep | An integer | zoom movement size in a step |

### 5.1.4. set the PTZ movement size in a step

You can specify any of the parameters you want to set.

request:

GET/POST /config/ptz_step.cgi

parameters:

see the above table.

response:
    see the above table.

### 5.1.5. list all PTZ presets

request:
    GET /config/ptz_preset_list.cgi

response:

| Name | Value | Description |
|---|---|---|
| presets | <preset name1>, ... | all presets |
| <preset name1><br>... | <p>,<t>,<z><br>... | the position of the preset name line by line.<br>for example,<br>door1=100,0<br>gate1=-20,-100 |

### 5.1.6. add, delete or goto a PTZ preset

request:
    GET/POST /config/ptz_preset.cgi

    parameters:

| Name | Value | Description |
|---|---|---|
| Name |  | preset name |
| Act | add<br>del<br>go | add the current position to the preset<br>delete the preset<br>go to the preset |

response:
    see the above table.

### 5.1.7. move PTZ absolutely

request:
    GET/POST /config/ptz_move.cgi

    parameters:

| Name | Value | Description |
|---|---|---|
| P | An integer | Pans the device relative to the home (0,0,0) position. |
| T | An integer | Tilts the device relative to the home (0,0,0) position. |
| Z | An integer | Zooms the device relative to the home (0,0,0) position. |

response:
    see the above table. If the movement is out of boundary, you will get the actual absolute position.

### 5.1.8. move PTZ relatively

request:
    GET/POST /config/ptz_move_rel.cgi

    parameters:

| Name | Value | Description |
|---|---|---|
| P | -32 ... 32 | Pans the device some steps relative to the current position. |
| T | -32 ... 32 | Tilts the device some steps relative to the current position. |
| Z | -32 ... 32 | Zooms the device some steps relative to the current position. |

response:
see the above table. If the movement is out of boundary, you will get the actual relative p, t, z values it moves.

### 5.1.9. get, set, goto, reset PTZ customized home position

request:
GET/POST /config/ptz_home.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| Act | get<br>set<br>go<br>reset | get current home position. This is the default value.<br>set current home position<br>go to home position<br>reset home position to factory setting |
| P | An integer | (only for act=set)Pans the device relative to the default (0,0,0) position. |
| T | An integer | (only for act=set)Tilts the device relative to the default (0,0,0) position. |
| Z | An integer | (only for act=set)Zooms the device relative to the default (0,0,0) position. |

response:
Return current home position.

| Name | Value | Description |
|---|---|---|
| P | An integer | the pan position |
| T | An integer | the tilt position |
| Z | An integer | the zoom position |

note:
If no any parameter is given, the effect will equivalent giving ‘act=get’. If any of parameters p,t,z is given and the value of parameter ‘act’ is not ‘set’, it will be ignored by camera.

### 5.1.10. Auto Patrol

request:
GET/POST /config/auto_patrol.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| Act | go<br>[continue]<br>[stop] | Run PTZ’s auto patrol.<br>act=go means run auto patrol function once<br>act=[continue] or act=[stop] means begin continuous patrol mode and stop patrol |

response:
see the above table.

note:
The item enclosed by [ ] means optional value. That optional value can be used only in some special models.

### 5.1.11. Auto Pan

request:
GET/POST /config/auto_pan.cgi

parameters:

| Name | Value | Description |
|---|---|---|

| | | |
|---|---|---|
| Act | go<br>[continue]<br>[stop] | Run PTZ's auto pan.<br>act=go means run auto pan function once<br>act=[continue] or act=[stop] means begin continuous pan mode and stop pan |

response:
see the above table.

note:
The item enclosed by [ ] means optional value. That optional value can be used only in some special models.

### 5.1.12. Configure sequence order of presets for Auto Patrol

request:
GET /config/config_auto_patrol.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| presets | <preset name1>, <preset name2>,... | A sequence of presets. The CGI auto_patrol function move camera PTZ preset by this sequence. Maximum count of preset in this sequence is 20.<br>Note: if this parameter is not given, the camera will list current sequence. |

response:
see the above table. If the count of given preset point is greater than 20, only first 20 preset in the sequence will kept by camera.

## 5.2. via RS-485

### 5.2.1. do RS-485 commands

request:
GET/POST /config/rs485_do.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| direction | 0-13 | 0: wide (zoom out) with stop.<br>1: up with stop<br>2: tele (zoom in) with stop<br>3: left with stop<br>4: home (only for custom command)<br>5: right with stop<br>6: focus far with stop<br>7: down with stop<br>8: focus near with stop<br>9:<br>10: custom command 1<br>11: custom command 2<br>12: custom command 3<br>13: custom command 4 |
| speed | An integer | speed control for up, down, left, right. (1 – 10)<br>(includes: Dyna, Lilin, Lilin2, PelcoD, PelcoP) |

## 5.3. Software input control

### 5.3.1. Trigger software input event

request:

GET/POST /config/sw_input.cgi

parameters:

| Name | Value | Description |
|---|---|---|
| trigger | on, off | Means software input being turn on or off |
| <param1> | <value1> | (option)Parameter1. This value is dependent with camera |
| <param2> | <value2> | (option)Parameter2. |
| … | … | … |

To set an event handler, please refer to 3.5.3 and 3.5.4.

# 6. Notification API

## 6.1. Camera status notification

### 6.1.1. get the notification status

request:

GET /config/notify.cgi

response:

| Name | Value | Description |
|---|---|---|
| md# | on, off | event motion detection # is triggered or not. |
| mdv# | <degree of motion> (0-100) | Percentage of motion detected by camera. |
| input# | on, off | event input # is triggered or not. |
| storagefull | on, off | event storage full |
| storagefail | on, off | event storage fail |
| recording | on, off | status is recording |
| snapshooting | on, off | status is snapshooting |
| mdetecting | on, off | status is motion detecting |
| output# | on, off | status of output # is on or off |
| vsignal | on, off | status of video signal is on or lost |
| speaker | on, off | status of speaker is on or off |
| mic | on, off | status of microphone is on or off |

### 6.1.2. get the notification stream

request:

GET /config/notify_stream.cgi

response:

The client side should keep receiving notification information from camera. It includes all available events or status as follow table. The notification information is only generated on while event or status changed. If there is no any changed event or status being reported within 30 second, a special tag: "`keep_alive`" will be send to the client side.

| Name | Value | Description |
|---|---|---|
| md# | on, off | event motion detection # is triggered or not. |
| mdv# | <degree of motion> (0-100) | Percentage of motion detected by camera. |
| input# | on, off | event input # is triggered or not. |
| storagefull | on, off | event storage full |
| storagefail | on, off | event storage fail |
| recording | on, off | status is recording |
| snapshooting | on, off | status is snapshooting |
| mdetecting | on, off | status is motion detecting |
| output# | on, off | status of output # is on or off |
| vsignal | on, off | status of video signal is on or lost |
| speaker | on, off | status of speaker is on or off |
| mic | on, off | status of microphone is on or off |
| cameraname | <camera name> | status of camera name |

# 7. RTSP API

The Real-Time Streaming Protocol (RTSP) is a protocol to get audio and video streaming data provided by a media server. IP camera can act as a media server and stream the real time audio and video . By RTSP request, a client application can get streaming data from IP camera. The detail of RTSP protocol please refer to RFC 2326.

## 7.1. Live streaming

### 7.1.1. get URL entry of specified profile

request:

GET /config/rtspurl.cgi

parameters:

profileid=<video profile number>

response:

| Name | Value | Description |
|---|---|---|
| profileid | # | profile number (# is a number from 1 to the count of profiles) |
| urlentry | <entry of video profile> | URL entry of associated video stream profile |

### 7.1.2. set video config

request:

GET/POST /config/rtspurl.cgi

parameters:

see the above table.

response:

see the above table.

### 7.1.3. Get live video

The requested URI of an IP camera stream data can be described by following:

rtsp://<server ip>/<urlentry>

Get video and audio stream for use on PC.

Where <urlentry> is the url entry associated with one of the video profile. The value can be got by calling **/config/rtspurl.cgi** (see 7.1.1)

**NOTE:**

Since our camera now can let user to change the url entry of each video profile, the following rtsp urls are obsolete, user should use 7.1.3 to get rtsp stream.

rtsp://<server ip>/mp4

Get video and audio stream with MPEG-4 video format for use on PC.

rtsp://<server ip>/jpeg

Get video (and audio) stream with M-JPEG video format for use on PC.

rtsp://<server ip>/3gpp

Get video (and audio) stream with MPEG-4 video format for use on 3gpp compliant device.

rtsp://<server ip>/live#

where # is the number from 1 to the count of video profile. For example, use rtsp://192.168.1.1/live1 to get the stream of video profile number 1.

## 7.2 RTSP Methods:

A. OPTIONS: Report the methods supported by the IP camera.

Please use "OPTIONS" method to get the other methods supported by the IP camera.

# 8. Appendix

## 8.1. Table used in NIPCA

*Table 1: Time zone*

| ID | Time zone |
|---|---|
| 1 | (GMT-12:00) International Date Line West |
| 2 | (GMT-11:00) Midway Island, Samoa |
| 3 | (GMT-10:00) Hawaii |
| 4 | (GMT-09:00) Alaska |
| 5 | (GMT-08:00) Pacific Time (US & Canada) |
| 6 | (GMT-08:00) Tijuana, Baja California |
| 7 | (GMT-07:00) Chihuahua, La Paz, Mazatlan |
| 8 | (GMT-07:00) Mountain Time (US & Canada) |
| 9 | (GMT-07:00) Arizona |
| 10 | (GMT-06:00) Central America |
| 11 | (GMT-06:00) Guadalajara, Mexico City, Monterrey |
| 12 | (GMT-06:00) Saskatchewan |
| 13 | (GMT-06:00) Central Time (US & Canada) |
| 14 | (GMT-05:00) Bogota, Lima, Quito, Rio Branco |
| 15 | (GMT-05:00) Eastern Time (US & Canada) |
| 16 | (GMT-05:00) Indiana (East) |
| 17 | (GMT-04:00) Caracas, La Paz |
| 18 | (GMT-04:00) Atlantic Time (Canada) |
| 19 | (GMT-04:00) Santiago |
| 20 | (GMT-04:00) Manaus |
| 21 | (GMT-03:30) Newfoundland |
| 22 | (GMT-03:00) Buenos Aires, Georgetown |
| 23 | (GMT-03:00) Brasilia |
| 24 | (GMT-03:00) Greenland |
| 25 | (GMT-03:00) Montevideo |
| 26 | (GMT-02:00) Mid-Atlantic |
| 27 | (GMT-01:00) Azores |
| 28 | (GMT-01:00) Cape Verde Is. |
| 29 | (GMT) Greenwich Mean Time : Dublin, Edinburgh, Lisbon, London |
| 30 | (GMT) Casablanca, Monrovia, Reykjavik |
| 31 | (GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague |
| 32 | (GMT+01:00) West Central Africa |
| 33 | (GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb |
| 34 | (GMT+01:00) Brussels, Copenhagen, Madrid, Paris |
| 35 | (GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna |
| 36 | (GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius |
| 37 | (GMT+02:00) Athens, Bucharest, Istanbul |
| 38 | (GMT+02:00) Beirut |
| 39 | (GMT+02:00) Harare, Pretoria |
| 40 | (GMT+02:00) Cairo |
| 41 | (GMT+02:00) Minsk |
| 42 | (GMT+02:00) Amman |

| ID | Time zone |
|---|---|
| 43 | (GMT+02:00) Windhoek |
| 44 | (GMT+02:00) Jerusalem |
| 45 | (GMT+03:00) Baghdad |
| 46 | (GMT+03:00) Moscow, St. Petersburg, Volgograd |
| 47 | (GMT+03:00) Tbilisi |
| 48 | (GMT+03:00) Nairobi |
| 49 | (GMT+03:00) Kuwait, Riyadh |
| 50 | (GMT+03:30) Tehran |
| 51 | (GMT+04:00) Baku |
| 52 | (GMT+04:00) Abu Dhabi, Muscat |
| 53 | (GMT+04:00) Yerevan |
| 54 | (GMT+04:30) Kabul |
| 55 | (GMT+05:00) Ekaterinburg |
| 56 | (GMT+05:00) Islamabad, Karachi, Tashkent |
| 57 | (GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi |
| 58 | (GMT+05:30) Sri Jayawardenepura |
| 59 | (GMT+05:45) Kathmandu |
| 60 | (GMT+06:00) Astana, Dhaka |
| 61 | (GMT+06:00) Almaty, Novosibirsk |
| 62 | (GMT+06:30) Yangon (Rangoon) |
| 63 | (GMT+07:00) Krasnoyarsk |
| 64 | (GMT+07:00) Bangkok, Hanoi, Jakarta |
| 65 | (GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi |
| 66 | (GMT+08:00) Taipei |
| 67 | (GMT+08:00) Irkutsk, Ulaan Bataar |
| 68 | (GMT+08:00) Perth |
| 69 | (GMT+08:00) Kuala Lumpur, Singapore |
| 70 | (GMT+09:00) Yakutsk |
| 71 | (GMT+09:00) Osaka, Sapporo, Tokyo |
| 72 | (GMT+09:00) Seoul |
| 73 | (GMT+09:30) Adelaide |
| 74 | (GMT+09:30) Darwin |
| 75 | (GMT+10:00) Hobart |
| 76 | (GMT+10:00) Brisbane |
| 77 | (GMT+10:00) Vladivostok |
| 78 | (GMT+10:00) Canberra, Melbourne, Sydney |
| 79 | (GMT+10:00) Guam, Port Moresby |
| 80 | (GMT+11:00) Magadan, Solomon Is., New Caledonia |
| 81 | (GMT+12:00) Fiji, Kamchatka, Marshall Is. |
| 82 | (GMT+12:00) Auckland, Wellington |
| 83 | (GMT+13:00) Nuku'alofa |

*Table 2: dynamic DNS service providers*

| ID | provider URIs |
|---|---|
|  | www.ez-ip.net |
|  | www.penguinpowered.com |
|  | members.dhs.org |
| dyndns | members.dyndns.org |
|  | www.3322.org |
|  | update.ods.org |
|  | cgi.tzo.com |

| | |
|---|---|
| | members.easydns.com |
| | api.easydns.com |
| | www.justlinux.com |
| | www.dyns.cx |
| | dup.hn.org |
| | www.zoneedit.com |
| | ipv6tb.he.net |

## 8.2. Advanced IP-Camera Stream (ACS) Header

*Multimedia header:*

| ACS Audio header | ACS Video header |
|---|---|
| typedef struct _ACS_AudioHeader<br>{<br>unsigned long ulHdrID; //Header ID<br>unsigned long ulHdrLength;<br>unsigned long ulDataLength;<br>unsigned long ulSequenceNumber;<br>unsigned long ulTimeSec;<br>unsigned long ulTimeUSec;<br>unsigned long ulDataCheckSum;<br>unsigned short usFormat;<br>unsigned short usChannels;<br>unsigned short usSampleRate;<br>unsigned short usSampleBits;<br>unsigned long ulReserved;<br>}ACS_AudioHeader, *PACS_AudioHeader; | typedef struct _ACS_VideoHeader<br>{<br>unsigned long ulHdrID; //Header ID<br>unsigned long ulHdrLength;<br>unsigned long ulDataLength;<br>unsigned long ulSequenceNumber;<br>unsigned long ulTimeSec;<br>unsigned long ulTimeUSec;<br>unsigned long ulDataCheckSum;<br>unsigned short usCodingType;<br>unsigned short usFrameRate;<br>unsigned short usWidth;<br>unsigned short usHeight;<br>unsigned char ucMDBitmap;<br>unsigned char ucMDPowers[3];<br>}ACS_VideoHeader, *PACS_VideoHeader |

Description:

The byte order of this header is little-endian.

Common part:

`ulHdrID`: Special id for identifying ACS header. For audio: the value of this id is 0xF6010000 (since our header is in little-endian so the byte array of this id is ’00 00 01 F6’). For video the value is 0xF5010000.

`ulHdrLength`: Header length. (32 bytes in current version)

`ulDataLength`: Payload data length.

`ulSequenceNumber`: Sequence number.

`ulTimeSec`: Time stamp in sec since 1970/01/01 00:00.

`ulTimeUSec`: Microsecond part of time stamp

`ulDataCheckSum`: Store last 4 bytes of payload data.

Audio part:

`usFormat`: Audio data format. The possible value:

```
AFMT_MS_ADPCM: 0

AFMT_MU_LAW: 1
AFMT_A_LAW: 2
AFMT_IMA_ADPCM: 4
AFMT_U8: 8
AFMT_S16_LE: 0x10 /* Little endian signed 16*/
AFMT_S16_BE: 0x20 /* Big endian signed 16 */
AFMT_S8: 0x40
```

```
AFMT_U16_LE: 0x80 /* Little endian U16 */
AFMT_U16_BE: 0x100 /* Big endian U16 */
AFMT_MPEG: 0x200 /* MPEG (2) audio */
AFMT_AC3: 0x400

AFMT_AMR: 0x800
```

`usChannels`: Audio channels number: mono(1) or stereo(2).

`usSampleRate`: Sample rate.

`usSampleBits`: Bits count per sample.

`ulReseverd`: Reserved.

Video part:

`usCodingType`: Encoding type of frame. The possible values are:

VFCT_IVOP (MPEG4): 0

VFCT_PVOP (MPEG4): 1

VFCT_JPEG: 5

VFCT_H264_IFRM: 10

VFCT_H264_PFRM: 11

`usFrameRate`: Frames per second.

`usWidth`: The width of frame dimension

`usHeight`: The height of frame dimension

`ucMDBitmap`: The height of frame dimension

`ucMDPowsers[3]`: The height of frame dimension

## *Extension header:*

We propose add extensive header for dealing with other information attaching with the multimedia stream. Instead of appending this kind of information to multimedia stream, it can save more bandwidth utilization.

Table: Alphanetworks IP-Camera streaming (ACS) extension header:

| **ACS extension header** |
|---|
| typedef struct _ACS_ExtentHeader<br>{<br>unsigned long ulHdrID; // ’00 00 01 FE’<br>unsigned long ulHdrLength;<br>unsigned char pbyReserved[96];<br>} ACS_ExtentHeader, *PACS_ExtentHeader |

Description:

The extension header is interleaved within the video stream or audio stream when the information is required by client.

`ulHdrID`: Special id for identifying ACS header. 0xFE010000.

`ulHdrLength`: Header length. (32 bytes in current version)

`pbyReserved[96]`: To be defined.