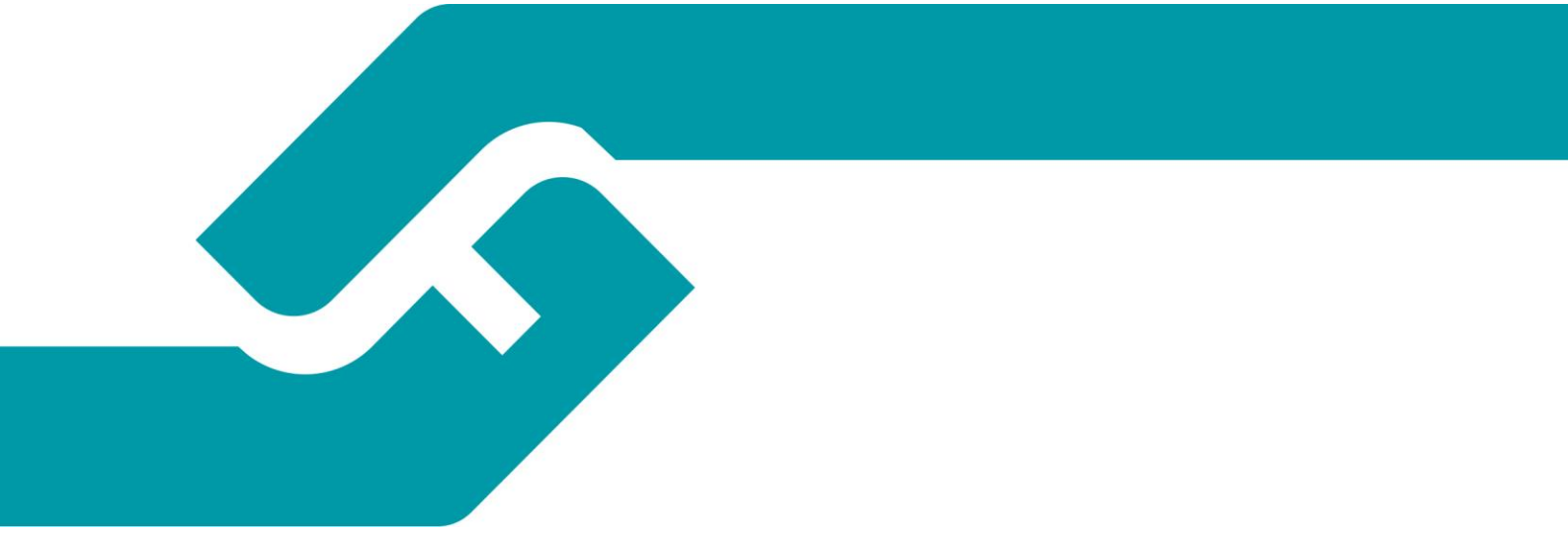
# NVR SDK

User Guide |
2019/8/23

# Table of Contents

# 1. NVR Device Discover

```
Send following structure to PORT 31501 via UDP broadcast.
  struct
  {
    unsigned char NetMagic[4];       // fixed code 0x26982334
    unsigned char Cmd[2];            // 0x0401 for inquiring device information
    unsigned char SenderMAC[6];
    unsigned char RecverMAC[6];
    unsigned char IntentDocFormat; // fill 1 intend DeviceInfo XML response
    unsigned char Reserved;
    unsigned char PadloadLen[2];   // fill zero
  };  // total 22 bytes
Response XML string
  <DeviceInfo version='1.0'>
    <HttpPort>80</HttpPort>
    <DeviceName>AiO NVR</DeviceName>
    <ModelName>undefined</ModelName>
    <FwVer>1.0.25-201911271625</FwVer>
    <Category>NVR</Category>
    <Ipv4>
      <Addr>192.168.33.167</Addr>
      <Mask>255.255.255.0</Mask>
    </Ipv4>
    <MacAddr>00:04:4b:e6:32:f7</MacAddr>
    <IPChannels>8</IPChannels>
    <AnalogChannels>0</AnalogChannels>
  </DeviceInfo>
```

---

# 2. Device manager

---

## 2.1 Camera probe

```
Url: http://ip:port/device/probe
Method: GET
Parameter: None
Return: JSON format array
  [{
    "DeviceName":   Device name
    "IPAddress":    IPV4 address  xxx.xxx.xxx.xxx
    "Port":         Port number
    "Manufacturer": Manufacturer of the device
    "Model":        Model name of the device
    "MacAddress":   mac address of the device
  }]
```

Description:
Command NVR system to probe devices in the network. Returns array of found devices.
```

---

## 2.2 Add camera

```
Url: http://ip:port/device/addCamera
Method: POST
Parameter: JSON object
  {
    "DeviceName": Device name
    "IPAddress": IPV4 address xxx.xxx.xxx.xxx
    "Port": Port number
    "UserName": user name to connect to the device
    "Password": password to connect to the device
    "Manufacturer": Manufacturer of the device (optional)
    "Model": Model name of the device (optional)
    "MacAddress": mac address of the device (optional)
    "protocol": protocol of the device
    "SubnetMask": SubnetMask xxx.xxx.xxx.xxx
    "Enable": boolean for the device is enabled or not
    "ODSPosition":
  }
Return: JSON string
  {
    "Result": "OK" or "Error" if there is an Error.
    "Description":
      If the “Result” is “OK”, the description is "ChannelID=theChannelID".
      If the “Result” is “Error”, the description is about the Error.
  }
Description:
Command NVR system to add a device to a channel. NVR system will use default profile and find a channel to add it.
The return object will show the result of this action, errors will be described in the description field if there is any.
```

---

## 2.3 Get camera list

```
Url: http://ip:port/device/cameraList
Method: GET
Parameter: None
Return: JSON format array
  [{
    "ChannelID": Channel ID of the device
    "Enable": boolean for the device is enabled or not
```
```

```
    "DeviceName":   Device name
    "IPAddress":    IPV4 address xxx.xxx.xxx.xxx
    "Port":         Port number
    "Manufacturer": Manufacturer of the device
    "FirmwareVersion": FirmwareVersion of the device
    "Model":        Model name of the device
    "MacAddress":   mac address of the device
    "protocol":     protocol of the device
    "SubnetMask":    SubnetMask xxx.xxx.xxx.xxx
    "ODSPosition":
  }]
Description:
  Retrieve the list of devices currently added to the NVR system.
```

---

## 2.4 Get camera setting

```
Url: http://ip:port/device/cameraDetail?ChannelID={ChannelID}
Parameters:
  ChannelID:  device channel ID
Method: GET
Return: JSON format object
  {
    "ChannelID":    Channel ID of the device
    "Enable":       (data type: boolean)  If the device is enabled.
    "DeviceName":   Device name
    "IPAddress":    IPV4 address xxx.xxx.xxx.xxx
    "SubnetMask":    SubnetMask xxx.xxx.xxx.xxx
    "Port":         Port number
    "Manufacturer": Manufacturer of the device
    "FirmwareVersion": FirmwareVersion of the device
    "Model":        Model name of the devicec
    "MacAddress":   mac address of the device
    "protocol":                  protocol of the device
    "UserName":     user name to connect to the device
    "Password":     password to connect to the device
    "Schedule":     Schedule record setting,
    "TimeSync":     Synchronize time with NVR periodically
    "ODSPosition":
  }
Description:
  Retrieve device settings of a single channel.
```

---

## 2.5 Delete camera

```
Url: http://ip:port/device/deleteCamera?ChannelID={ChannelID}
Parameters:
  ChannelID:  device channel ID
Method: GET
Return: JSON format object
{
  "Result": “OK” or “Error”
  "Description": If the "Result" is “OK”, the description would be the deleted channel ID.
    If the "Result" is Error, the description is about the Error.
}
Description:
  Delete device from the NVR system by channel ID.
```

---

## 2.6 Update camera setting

```
Url: http://ip:port/device/updateCamera?ChannelID={ChannelID}
Parameters:
  ChannelID:  device channel ID
Method: POST
Parameter: JSON format object
  {
    "ChannelID":  The Channel ID after replacing.
    "Enable":  boolean for the device is to be enabled or not.
    "DeviceName":  Device name to be replaced.
    "IPAddress":  IP address(IPV4) is to be replaced.
    "Port":  Port number is to be replaced.
    "UserName":  User name to be replaced.
    "Password":  Password to be replaced.
    "protocol":  protocol of the device
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, the description would be the target Channel ID before after replacing.
        If the "Result" is “Error”, the description is about the error.
  }
Description:
  Update the settings of a device, the item of the parameter can be one or more than one, not all fields are required.
  Only the updating ones are enough.
```

---

## 2.7 Set camera

```
Url: http://ip:port/device/setCamera?ChannelID={ChannelID}
Parameters:
   ChannelID:   device channel ID
Method: POST
Parameter: JSON format object
   {
      "ChannelID":    The specified Channel ID to add a camera.
      "Enable":       boolean for the device is to be enabled or not.
      "DeviceName":   Device name to be replaced.
      "IPAddress":    IP address(IPV4) is to be replaced.
      "Port":         Port number is to be replaced.
      "UserName":     User name to be replaced.
      "Password":     Password to be replaced.
      "Manufacturer": Manufacturer of the device (optional)
      "FirmwareVersion": FirmwareVersion of the device (optional)
      "Model":        Model name of the device (optional)
      "MacAddress":   mac address of the device (optional)
      "protocol":     protocol of the device
      "SubnetMask":    SubnetMask xxx.xxx.xxx.xxx
      "ODSPosition":
   }
Return: JSON format object
   {
      "Result":  "OK", or “Error”
      "Description":
            If the "Result" is “OK”, the description would be the specified Channel ID to add device.
            If the "Result" is “Error”, the description is about the error.
   }
Description:
   Add a device to a specified channel.
   NVR system will use default profile.
   The return object will show the result of this action, errors will be described in the description field if there is any.


```

---

## 2.8 Get camera status

HTTP and Websocket are supported to get the status of camera

```
HTTP
Url: http://ip:port/device/cameraStatus
Method: GET
Return: JSON format array
```

[{
"ChannelID": Channel ID of the status.
"Online": boolean for the channel is on line or not.
"Recording": boolean for the channel is recording or not.
"Event": boolean for the channel is triggered by an event or not.
"Manual": boolean for the channel is manual recording or not.
}]
Description:
Get the status of each channels in NVR system.

Websocket
Url: ws://ip:port/device/cameraStatus
Message: JSON format object array
[{
"ChannelID": Channel ID of the status.
"Online": boolean for the channel is on line or not.
"Recording": boolean for the channel is recording or not.
"Event": boolean for the channel is triggered by an event or not.
}]
Description:
Status of each channels in NVR system will be sent per second.
```
---

## 2.9 PTZ Control

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Parameters:
ChannelID: device channel ID
Method: POST
Parameter: JSON format object
{
"command": "tilt_up"|"tilt_down"|"pan_left"|"pan_right"|"zoom_in"|"zoom_out"|"stop"
"speed": move speed for the move command
"number": the id number for preset or goto
}
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}
```
---

## 2.10 Set preset

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}

Method: POST
Parameter: JSON object
```
{
  "command": "set_preset"
  "parameters" :
  {
    "preset_name": String, Required. Name of the preset.
    "preset_token": String, Optional. token of the preset.
  }
}
```
Return: JSON string
```
{
  "PresetToken": String, Optional. token of the preset.
}
```
Description:
Set preset with name, or update preset with preset token.
If set preset without preset token, it is to create a new preset and return the new preset token.
If set preset with preset token, it is to update the preset identified by the preset token. And return the same preset token.

```
---

## 2.11 Remove preset

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON object
{
  "command": "remove_preset"
  "parameters" :
  {
    "preset_token": String, required. token of the preset.
  }
}
Return: JSON string
{
  "Result": "OK", or “Error”
  "Description":
      If the "Result" is “OK”, removed the preset successfully.
      If the "Result" is “Error”, fail to remove the preset.
}
```

Description:
Remove the preset identified by the preset token.

```
---

## 2.12 Goto preset

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON  object
  {
    "command":  "goto_preset"
    "parameters"   :
    {
      "preset_token":     String, required.  token of the preset.
      "speed":     Floating point number, optional, default is 0.1.  Speed to goto the preset. Max is 1.
    }
  }
Return: JSON string
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, goto the preset successfully.
        If the "Result" is “Error”, fail to goto the preset.
  }

Description:
  Goto the preset identified by the preset token.
```

---

## 2.13 Get presets

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON  object
  {
    "command":  "get_presets"
  }
Return: JSON string
  {
    "presets":
     [
       {
         name:         String. name of the preset.
         preset_token:   String. preset token.
       }
     ]
  }
Description:
  Get all presets set in the device.
```

```
---
```

## 2.14 Start tour

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON  object
  {
    "command":  "start_tour"
    "parameters"   :
    {
      "speed":     Floating point number, optional, default is 0.1.  Speed to goto the preset. Max is 1.
      "picked_presets":
      [
        {
          name:         String. Required. name of the preset.
          preset_token:  String. Required. preset token.
          time:         Number. Required. timeout seconds to goto the next preset.
        }
      ]
    }
  }
Return: JSON string
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, Start tour successfully.
        If the "Result" is “Error”, fail to start tour.
  }

Description:
  Start tour with a set of presets.
  If there is no picked presets, the presets of the tour setting will be cleared. And the running tour will be stopped.

```
---
```

## 2.15 Stop tour

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON  object
  {
    "command":  "stop_tour"
  }
Return: JSON string
```
```

```
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “OK”, stop tour successfully.
          If the "Result" is “Error”, fail to stop tour.
  }

Description:
  Stop tour.


```
---

## 2.16 Get tour presets

```
Url: http://ip:port/device/ptzControl?ChannelID={ChannelID}
Method: POST
Parameter: JSON  object
  {
    "command":   "get_tour_presets"
  }
Return: JSON string
  {
    "presets":
     [
       {
          name:          String. name of the preset.
          preset_token:   String. preset token.
       }
     [
  }
Description:
  Get all presets of the tour .

```
---

## 2.17 Get network interface information

```
Url: http://ip:port/device/getNetworkInterfaces?ChannelID={ChannelID}
Method: POST
Parameter: N/A

Return: JSON string
{
  "Result":
  {
```

```
"NetworkInterfaces":
{
  "$":
  {
    "token":   String | onvif token of canera interface
  },
  "Enabled":     String | interface enabled / disabled
  "Info":
  {
    "Name":        String | interface name
    "HwAddress":   String | mac address
    "MTU":         String | MTU value
  },
  "Link":
  {
    "AdminSettings":
    {
      "AutoNegotiation":
      "Speed":
      "Duplex":
    },
    "OperSettings":
    {
      "AutoNegotiation":
      "Speed":
      "Duplex":
    },
    "InterfaceType":
  },
  "IPv4":    json object | IPv4 settings
  {
    "Enabled":  String | ipv4 enabled / disabled
    "Config":   json object | IPv4 settings
    {
      "Manual":
      {
        "Address":         String | ipv4 address
        "PrefixLength":    String | ipv4 netmask
      },
      "DHCP":    String | "true" or "false"
    }
  }
},
"NTPInformation":
{
  "FromDHCP":        String | "true" or "false"
  "NTPManual":
  {
    "Type":        String | for example "DNS",
    "IPv4Address":  String | for example "time.windows.com",
    "DNSname":     String | for example "time.windows.com"
```

```
        }
      },
      "HostnameInformation":
      {
        "FromDHCP":       String | "true" or "false"
        "Name":           String | for example "EBN1240-A"
      },
      "NetworkGateway":
      {
        "IPv4Address":
      },
      "DNSInformation":
      {
        "FromDHCP":       String | "true" or "false"
        "DNSManual":      String array | DNS address list
        [
          {
            "Type":       String | for example "IPv4"
            "IPv4Address": String | for example "168.95.1.1"
          },
          {
            "Type":       String | for example "IPv4",
            "IPv4Address": String | for example "0.0.0.0"
          }
        ]
      },
      "NetworkProtocols":    json object | supported protocols
      [
        {
          "Name": "HTTP",
          "Enabled": "false",
          "Port": "80"
        },
        {
          "Name": "HTTPS",
          "Enabled": "false",
          "Port": "0"
        },
        {
          "Name": "RTSP",
          "Enabled": "true",
          "Port": "554"
        }
      ]
    }
}
```

Description:

Get camera network interface information.

```
```

---

## 2.18 Set camera network interface

```
Url: http://ip:port/device/setNetworkInterfaces?ChannelID={ChannelID}
Method: POST
Parameter: JSON object ( for manual setting example )
{
  "NetworkInterfaces": {
    "$":
    {
      "token": "eth0"
    },
    "Enabled": "true",
    "MTU": "1500",
    "IPv4":
    {
      "Enabled": "true",
      "Config":
      {
        "Manual":
        {
          "Address": "192.168.33.249",
          "PrefixLength": "24"
        },
        "DHCP": "false"
      }
    },
    "NetworkGateway":
    [
      { "IPv4Address": "192.168.33.254" },
      { "IPv4Address": "192.168.33.253" }
    ]
  }
}

Parameter: JSON object ( for dhcp setting example )
{
  "NetworkInterfaces":
  {
    "$":
    {
      "token": "eth0"
    },
    "Enabled": "true",
    "MTU": "1500",
    "IPv4":
    {
      "Enabled": "true",
```

```
        "Config":
        {
           "DHCP": "true"
        }
      },
      "NetworkGateway":
      [
        { "IPv4Address": "192.168.33.254" },
        { "IPv4Address": "192.168.33.253" }
      ]
   }
}
```

Return: JSON string

```
   {
      "Result":  "OK", or “Error”
      "Description":  onvif error message.
   }
```

Description:

```
```

---

## 2.19 Set camera DNS

```
```

Url: http://ip:port/device/setDNS?ChannelID={ChannelID}
Method: POST
Parameter: JSON object ( for manual setting example )

```
{
   "DNSInformation":
   {
      "FromDHCP": "false",
      "DNSManual":
      [
        {
           "Type": "IPv4",
           "IPv4Address": "192.168.33.254"
        },
        {
           "Type": "IPv4",
           "IPv4Address": "8.8.8.8"
        }
      ]
   }
}
```

Parameter: JSON object ( for dhcp setting example )

```
{
```

```
  "DNSInformation":
  {
    "FromDHCP": "true"
  }
}
```

Return: JSON string

```
  {
    "Result":  "OK", or “Error”
    "Description":  onvif error message.
  }
```

Description:

```
---

## 2.20 Set camera NTP server

```
Url: http://ip:port/device/setNTP?ChannelID={ChannelID}
Method: POST
Parameter: JSON object ( for manual setting example )
{
  "NTPInformation":
  {
    "FromDHCP": "false",
    "NTPManual":
    {
      "Type": "DNS",
      "DNSname": "time.windows.com"
    }
  }
}
--- or ---
{
  "NTPInformation":
  {
    "FromDHCP": "false",
    "NTPManual":
    {
      "Type": "IPv4",
      "DNSname": "192.168.1.1"
    }
  }
}
--- or ---
{
  "NTPInformation":
  {
```
```

```
    "FromDHCP": "false",
    "NTPManual":
    {
      "Type": "IPv6",
      "DNSname": "xxxxxxxxxxxxxxxxx"
    }
  }
}
```

Parameter: JSON object ( for dhcp setting example )

```
{
  "NTPInformation":
  {
    "FromDHCP": "true"
  }
}
```

Return: JSON string

```
  {
    "Result":  "OK", or “Error”
    "Description":  onvif error message.
  }
```

Description:

```
```

---

## 2.21 Get camera encoder configurations

```
```

Url: http://ip:port/device/GetVideoEncoderConfigurations
Method: POST
Parameter: Array of JSON object

```
[
  {
    "ch":1,
    "mainsub":0
  },
  {
    "ch":2,
    "mainsub":0
  },
  {
    "ch":3,
    "mainsub":0
  },
  {
    "ch":4,
    "mainsub":0
```

```
  },
]

Return: JSON object
{
  "Result":     json object | array of camera encoder configurations
  [
    {
      "ch": 1,      Integer | the channel number
      "mainsub": 0,   Integer | 0 = main stream / 1 = sub stream
      "Configuration":
      {
        "$":
        {
          "token": "VideoEncoderToken_1"     String | the video encoder token
        },
        "Name": "VideoEncoder_1",            String | the video encoder name
        "UseCount": "1",                  String | current user count
        "Encoding": "H264",                 String | current video encoder
        "Resolution":                     Json Object | current video resolution
        {
          "Width": "1920",
          "Height": "1080"
        },
        "Quality": "5",                   String | current video quality
        "RateControl":
        {
          "FrameRateLimit": "30",          String | current FPS
          "EncodingInterval": "1",
          "BitrateLimit": "2048"           String | current bitrate
        },
        "H264":
        {
          "GovLength": "60",              String | current I-frame interval
          "H264Profile": "Main"
        },
        "Multicast":
        {
          "Address":
          {
            "Type": "IPv4",
            "IPv4Address": "0.0.0.0"
          },
          "Port": "8860",
          "TTL": "128",
          "AutoStart": "false"
        },
        "SessionTimeout": "PT00H01M00S",
        "AudioEnabled": false,             Bool | enable / disable audio
        "AudioSupport": true,              Bool | audio supported
      },
```

```
"Options":          Json Object | supported configurations
{
  "QualityRange":    Json Object | supported quality range
  {
    "Min": "0",
    "Max": "5"
  },
  "codecs":         Json Object Array | supported codec list
  [
    {
      "name": "H264",
      "params":
      {
        "ResolutionsAvailable":    Json Object Array | supported resolution list
        [
          {
            "Width": "1280",
            "Height": "720"
          },
          {
            "Width": "1280",
            "Height": "960"
          },
          {
            "Width": "1920",
            "Height": "1080"
          },
          {
            "Width": "2048",
            "Height": "1520"
          },
          {
            "Width": "2304",
            "Height": "1296"
          },
          {
            "Width": "2592",
            "Height": "1520"
          },
          {
            "Width": "2592",
            "Height": "1944"
          }
        ],
        "GovLengthRange":      Json Object | supported I-frame interval range
        {
          "Min": "0",
          "Max": "100"
        },
        "FrameRateRange":      Json Object | supported frame rate range
        {
```

```
                "Min": "1",
                "Max": "30"
              },
              "EncodingIntervalRange":
              {
                "Min": "1",
                "Max": "1"
              },
              "H264ProfilesSupported":
              [
                "Baseline",
                "Main",
                "High"
              ],
              "BitrateRange":       Json Object | supported bitrate range
              {
                "Min": "256",
                "Max": "8192"
              }
            }
          }
        ]
      }
    }
  ],
  "Description": ""
}
```

Description:

```
---
```

## 2.22 Set camera encoder configurations

```
Url: http://ip:port/device/SetVideoEncoderConfiguration
Method: POST
Parameter: Array of JSON object
[
  {
    "ch": 1,
    "mainsub": 0,
    "Configuration":
    {
      "$":
      {
        "token": "VideoEncoderToken_1"
      },
      "Name": "VideoEncoder_1",
```

```
        "UseCount": "1",
        "Encoding": "H264",
        "Resolution":
        {
           "Width": "1920",
           "Height": "1080"
        },
        "Quality": "5",
        "RateControl":
        {
           "FrameRateLimit": "30",
           "EncodingInterval": "1",
           "BitrateLimit": "4096"
        },
        "H264":
        {
           "GovLength": "60",
           "H264Profile": "Main"
        },
        "Multicast":
        {
           "Address":
           {
              "Type": "IPv4",
              "IPv4Address": "0.0.0.0"
           },
           "Port": "8860",
           "TTL": "128",
           "AutoStart": "false"
        },
        "AudioEnabled": true,
        "SessionTimeout": "PT00H01M00S"
     }
  }
]
```


Return: JSON object

```
  {
     "Result":  "OK", or “Error”
     "Description":  onvif error message.
  }
```

Description:

```
```

---

# 3. Streaming

---

## 3.1 Live

Both RTSP and Websocket are supported for live stream
```
RTSP
rtsp://ip:port/live/{ChannelID}/{StreamID}
Parameter:
   ChannelID :    The requested channel ID for live stream.
   StreamID:      0 means main stream, 1 means second stream.

Websocket
ws://ip:port/live/{ChannelID}/{StreamID}
Parameter:
   ChannelID :    The requested channel ID for live stream.
   StreamID:      0 means main stream, 1 means second stream.

Websocket stream frame format:
   [32 bytes frame header] + [frame data]
   frame header (frame information):
   {
      u32 magic;         Magic number is 0x89639615.
      u32 datalen;       data length of the frame
      u16 frameType;     frame type
      u16 videoWidth;    Width of the Video
      u16 videoHeight;   Height of the Video
      u32 sec;           Unix time in sec since 1970/1/1.
      u16 msec;          msec is millisecond ( 0-999)
      u8  reverse[12];   reserved
   };

   frameType:
      FRAME_TYPE_H264I = 6,
      FRAME_TYPE_H264P = 7,
      FRAME_TYPE_H265I = 8,
      FRAME_TYPE_H265P = 9,
      FRAME_TYPE_G711 = 0x10
```
---

## 3.2 Live Audio backchannel

```
Websocket
ws://ip:port/audioBackchannel?ChannelID={ChannelID}
Parameter:
   ChannelID:  The channel ID to talk to, if None, to NVR server
```

Websocket stream frame format:
[32 bytes frame header] + [frame data]
frame header (frame information):
{
u32 magic; Magic number is 0x89639615.
u32 datalen; data length of the frame
u16 frameType; frame type
u16 videoWidth; Width of the Video
u16 videoHeight; Height of the Video
u32 sec; Unix time in sec since 1970/1/1.
u16 msec; msec is millisecond ( 0-999)
u8 reverse[12]; reserved
};

frameType:
FRAME_TYPE_G711 = 0x10
Description:
Send Audio back to NVR. Audio format is G711 ulaw.
```

---

## 3.3 Playback

Both RTSP and Websocket are supported for playback stream
```
RTSP
rtsp://ip:port/playback/{ChannelID}/{StreamID}?time={StartTime}&session={SessionID}

Websocket
ws://ip:port/playback/{ChannelID}/{StreamID}?time={StartTime}&session={SessionID}
Parameter:
ChannelID: The requested channel ID for playback.
StreamID: 0 means main stream, 1 means second stream.
StartTime: The time to start playback. Represented by UTC in Unix Time format.
SessionID: A unique string which generated by client.

Websocket playback stream frame format is the same as websocket Live stream frame format.
```

---

## 3.4 Playback control

Send command to stream server to control the playback behavior.
```
Parameter: JSON format object
{
"command": "stop"|"play"|"pause"|"forward"|"backward”|"fast_forward"|"fast_backward”|
"slow_forward"|"slow_backward"|"step_forward"|"step_backward"|"seek"

"time": UnixTime seconds since 1970/1/1, used by command: "seek". The seek command will seek to the Video frame by “time”, then the playback will be paused, not playing froward or backward.

"speed": 1|2|4|8|16 play speed is used by command: "fast_forward", "fast_backward”, "slow_forward", "slow_backward". minimum is 1, maximum is 16
};

Playback control for RTSP stream:
Send control command with "SET PARAMETER" RTSP command via rtsp connection.

Playback control for Websocket stream:
Send control command with text message via Websocket connection.
```
---

## 3.5 Playback control over http

```
Url: http://ip:port/device/playbackControl?session={SessionID}
Method: POST

Parameter:
SessionID: The SessionID to control playback.
JSON format object for post data
{
"command": "stop"|"play"|"pause"|"forward"|"backward”|"fast_forward"|"fast_backward”|"slow_forward"|"slow_backward"|"step_forward"|"step_backward"|"seek"

"time": UnixTime seconds since 1970/1/1, used by command: "seek". The seek command will seek to the Video frame by “time”, then the playback will be paused, not playing froward or backward.

"speed": 1|2|4|8|16 play speed is used by command: "fast_forward", "fast_backward”, "slow_forward", "slow_backward". minimum is 1, maximum is 16
};
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Send command to stream server to control the playback behavior.

```
```

---

## 3.6 GetSnapshot

```
getSnapshot

Url: http://ip:port/device/getSnapshot?ChannelID={ChannelID}
Method: GET

Return: JSON string
  {
      "snapshotPath": String | required | snapshot file path
  }

Description:
  Ask server to get snapshot
```
---

## 3.7 SearchSnapshotPath

```
Url: http://ip:port/device/searchSnapshotPath
Method: POST
Parameter: JSON  object
  {
    channels:    Object | array of channels to search for Snapshot
      [
        Number | required | channel number
      ]
    types:      array of Strings   [ "Manual" | "Normal" | "Motion" | "InputAlarm" ]    types to search for
Snapshot
    startTime:    Number | required | UTC time for start time
    endTime:     Number | required | UTC time for end time
  };

Return: JSON string
  [
    {
      "channel":    number | required | channel
      "timeStamp":   number | required | UTC time
      "type":      string | required | "Manual" or "Normal" or "Motion" or "InputAlarm"
      "snapshotPath": String | required | snapshot file path
    }
  ]
```

Description:
Search snapshot with channel, start time, and end time.

```
---
```

## 3.8 DeleteSnapshot

```

Url: http://ip:port/device/deleteSnapshot
Method: POST
Parameter: JSON object
{
"snapshotPath": String | required | snapshot file path
};

Return: JSON string
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “OK”, delete Snapshot successfully.
If the "Result" is “Error”, fail to delete Snapshot.
}

Description:
Delete snapshot on the disk by snapshot path.

```
---
```

## 3.9 Download snapshot

```

Url: http://ip:port/device/download/?snapshot={snapshotFilePath}
Method: GET

Return:
jpg binary data

or
when error

return JSON format object
{
"Result": “Error”
"Description":

If the "Result" is “Error”, the description is about the error.
}

Description:
Download snapshot with the Snapshot path retrieved from searchSnapshotPath.
```

---

# 4. Event and meta data

---

## 4.1 Throw an event

Both HTTP and Websocket are supported to throw an evant to NVR system
```
HTTP
URL: http://ip:port/meta/addEvent
Method: POST
Parameter: JSON format object
{
"ChannelID": The Channel ID
"Type": A string for event type .
"State": {active|inactive|once}
}
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “OK”, the event throw to NVR system
If the "Result" is “Error”, the description is about the error.
}
Description:
Throw an event to NVR system.

Websocket
URL: ws://ip:port/meta/addEvent
Parameter: JSON format object
{
"ChannelID": The Channel ID
"Type": A string for event type .
"State": {active|inactive|once}
}
Description:
Throw an event to NVR system via websocket connection.
```

---
```

## 4.2 Listen on events

Listen on the event hook can receive the all events throw by modules.
```
Websocket
URL: ws://ip:port/meta/eventHook
Message:
  {
    "ChannelID":    The Channel ID
    "Type":       A string for event type .
    "State":      {active|inactive|once}
  }
```
---

## 4.3 Throw a meta data

Both HTTP and Websocket are supported to throw a meta data to NVR system
```
HTTP
URL: http://ip:port/meta/addMeta
Method: POST
Parameter: JSON format object
  {
    "ChannelID":    The Channel ID
    "Type":       A string for event type .
    "Data":       A string for meta data
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, the meta data throw to NVR system
        If the "Result" is “Error”, the description is about the error.
  }
Description:
  Throw a meta data to NVR system.

Websocket
URL: ws://ip:port/meta/addMeta
Parameter: JSON format object
  {
    "ChannelID":    The Channel ID
    "Type":       A string for event type .
    "Data":       A string for meta data
  }
Description:
  Throw a meta data to NVR system via websocket connection.
```
---

## 4.4 Listen on meta datas

Listen on the meta data hook can receive the all meta datas throw by modules.

```
Websocket
URL: ws://ip:port/meta/metaHook
Message:
  {
    "ChannelID":    The Channel ID
    "Type":       A string for event type .
    "Data":       A string for meta data
  }
```

---

## 4.5 Event search

```
URL: http://ip:port/record/eventSearch
Method:
  POST
Parameter: JSON format object
  {
   "ChannelID": [1,2,3,4],
   "StartTime": 1584892800,
   "EndTime": 1584979199,
   "Type": ["Motion","InputAlarm","ManualRecord","Tag"],
   "Stream": "Main"  ( or "Sub" )
  }

Return: JSON format object array
 [
  {
    "ChannelID": 1,
    "ev_start": 1584949182,    | the event start time
    "ev_end": 1584949183,    | the event end time
    "pre_rec": 1584949172,     | the pre-rec time of this event (for archive)
    "post_rec": 1584949213,    | the post-rec time of this event (for archive)
    "video_size": 14707618,    | the video stream size of this event
    "Type": "Motion",         | the event type
    "comment": "Motion-1"      | the event comment
  },
 ]
Description:
  To search the events, for example Motion, InputAlarm...
```

---

### 4.6 Meta data search

```
URL: http://ip:port/record/metaSearch
Method:
  POST
Parameter: JSON format object
  {
    ChannelID:  The channel ID to search
    Type:      The meta data types to search
    StartTime:  Start time of the search, UTC in UNIX Time format
    EndTime:    End time of the search, UTC in UNIX Time format
  }
Return: JSON format object array
  [
    {
      ChannelID:    Channel ID of this event
      Type:      meta data type string, for example "diEventStart", "diEventStop", "manualStart",
"manualStop", "manualAlarmStart", "manualAlarmStop"
      Data:      meta data, for example active, inactive, once.
      Timestamp:  meta data time, UTC in UNIX Time format, in milisecond.
    }
  ]
Description:
  To search the meta datas.
```

---

## 5. Record map

---

### 5.1 Get map by day

```
URL: http://ip:port/record/getMapByDay
Method:
  POST
Parameter: JSON format object
  {
    ChannelIDs:    Channels array, 4 channels at most
    StartTime:     Start time of the query, UTC in UNIX Time format
    EndTime:       End time of the query, UTC in UNIX Time format
    mainsub:       1 is main, 2 is sub
  }
Return: JSON format object
  {
    StartTime:     Start time of the query, UTC in UNIX Time format
    EndTime:       End time of the query, UTC in UNIX Time format
    CH{ChannelID}: Map data of the channel.
  }
```

Note:
1. StartTime must be at day boundry.
2. The lonest duration is 35 days, duration longer than 35 days will be truncate to 35 days.
3. Map data is a string, the character in string present the record status in a day.
"+" means have record data.
"-" means have no record data.
4. For example:
"+++--+++" means
the 1st, 2nd, 3rd, 6th, 7th and 8th day have records, and the 4th and 5th day have no record.

```
---

## 5.2 Get map by minute

```
URL: http://ip:port/record/getMapByMinute
Method:
  POST
Parameter: JSON format object
  {
    ChannelIDs:   Channels array, 4 channels at most
    StartTime:    Start time of the query, UTC in UNIX Time format
    EndTime:      End time of the query, UTC in UNIX Time format
    mainsub:      1 is main, 2 is sub
  }
Return: JSON format object
  {
    StartTime:    Start time of the query, UTC in UNIX Time format
    EndTime:      End time of the query, UTC in UNIX Time format
    CH{ChannelID}: Map data array of the channel.
  }
Note:
  1. StartTime must be at hour boundry.
  2. The lonest duration is 24 hours, duration longer than 24 hours will be truncate to 24 hours.
  3. Map data is a string array, a string present an hour status.
    the character in string present the record status in a minute.
    "+" means have record data.
    "-" means have no record data.
  4 For example:
   ["++++++++++++++++++++++++++++++----------------------------",
    "----------------------------++++++++++++++++++++++++++++++",
   means
   the 1st~30th minute of the first hour has record and the 31st~60th have not,
   the 1st~30th minute of the second hour has no record and the 31st~60th have.
```
---
```

### 5.3 Get map by second

```
URL: http://ip:port/record/getMapBySecond
Method:
  POST
Parameter: JSON format object
  {
    ChannelIDs:   Channels array, 4 channels at most
    StartTime:    Start time of the query, UTC in UNIX Time format
    EndTime:      End time of the query, UTC in UNIX Time format
    mainsub:      1 is main, 2 is sub
  }
Return: JSON format object
  {
    StartTime:    Start time of the query, UTC in UNIX Time format
    EndTime:      End time of the query, UTC in UNIX Time format
    CH{ChannelID}: Map data array of the channel.
  }
Note:
  1. The lonest duration is 60 minutes, duration longer than 60 minutes will be truncate to 60 minutes.
  2. Map data is a string array, a string present a minute status.
     the character in string present the record status in a second.
    "+" means have record data.
    "-" means have no record data.
  4 For example:
   ["++++++++++++++++++++++++++++++----------",
    "----------++++++++++++++++++++++++++++++",
   means
   the 1st~30th second of the first minute has record and the 31st~60th have not,
   the 1st~30th second of the second minute has no record and the 31st~60th have.
```

---

# 6. Schedule Record

---

### 6.1 Set Schedule by group

```
Url: http://ip:port/schedule/setScheduleByGroup?ChannelID={ChannelID}&GroupID={GroupID}
Method: POST
Parameter:
    ChannelID: Which channel to set
    GroupID: Which group to set. Record, Snapshot, Email or FTP

Parameter: JSON  object
  {
    Normal: Onject | Required when the Group is Record, Snapshot and FTP
```

```
  {
      Map:   object  | Required
      [                                   //array has to be 7 lines to match 7 days a week, first line is
Sunday
        "----------------------------------------",      // '-' is not scheduled, '+' is scheduled
        "----------------------------------------",      // note count would be 48 for each day
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++"
      ]
  }
    Motion: Object | Required when the Group is Record, Snapshot, Email and FTP
    {
      Map:   object  | Required
      [
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++"
      ]
    }
    InputAlarm: Object | Required when the Group is Record, Snapshot, Email and FTP
    {
      Map:   object  | Required
      [
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++"
      ]
    }
    Intellegent: Onject | Required when the Group is Record, Email and FTP
    {
      Map:   object  | Required
      [
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "----------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++"
      ]
```

```
    }
  }

Return: JSON string
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, set schedule successfully.
        If the "Result" is “Error”, fail to set shcedule.
  }

Description:
  Set schedules by channel and group.
```

---

## 6.2 Set Schedule by group Multi channels

```
Url: http://ip:port/schedule/setScheduleByGroupMultiChannels
Method: POST

Parameter: JSON  object
  [
    {
      channel:   Number |  required | channel ID
      group:    String |  required | group ID
      schedules:
      {
        Normal: Onject | Required when the Group is Record, Snapshot and FTP
        {
          Map:   object  | Required
          [                                    //array has to be 7 lines to match 7 days a week, first line is
Sunday
            "----------------------------------------------",      // '-' is not scheduled, '+' is scheduled
            "----------------------------------------------",      // note count would be 48 for each day
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
            "++++++++++++++++++++-++++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
        Motion: Object | Required when the Group is Record, Snapshot, Email and FTP
        {
          Map:   object  | Required
          [
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
```

```
            "-------------------------------------------",
            "++++++++++++++++++++-++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
        InputAlarm: Object | Required when the Group is Record, Snapshot, Email and FTP
        {
          Map:   object  | Required
          [
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "++++++++++++++++++++-++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
        Intellegent: Onject | Required when the Group is Record, Email and FTP
        {
          Map:   object  | Required
          [
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "-------------------------------------------",
            "++++++++++++++++++++-++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
      }
    }
  ]

Return: JSON string
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “OK”, set schedule successfully.
        If the "Result" is “Error”, fail to set shcedule.
  }

Description:
  Set schedules with multi channels.
```

---

## 6.3 Get Schedule by group

```

Url: http://ip:port/schedule/getSchedulesByGroup?ChannelID={ChannelID}&GroupID={GroupID}
Method: GET
Parameter:
    ChannelID: Which channel to get
    GroupID: Which group to get. Record, Snapshot, Email or FTP


Return: JSON string
  {
    Normal: Onject | Required when the Group is Record, Snapshot and FTP
    {
      Map:  object  | Required
      [                                    //array has to be 7 lines to match 7 days a week, first line is
Sunday
        "----------------------------------------------",      // '-' is not scheduled, '+' is scheduled
        "----------------------------------------------",      // note count would be 48 for each day
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++++++++++"
      ]
    }
    Motion: Object | Required when the Group is Record, Snapshot, Email and FTP
    {
      Map:  object  | Required
      [
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++++++++++"
      ]
  }
    InputAlarm: Object | Required when the Group is Record, Snapshot, Email and FTP
    {
      Map:  object  | Required
      [
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "----------------------------------------------",
        "++++++++++++++++++++-++++++++++++++++++++++++",
        "++++++++++++++++++++++++++++++++++++++++++++++++"
```

```
        ]
      }
      Intellegent: Onject | Required when the Group is Record, Email and FTP
      {
        Map:   object  | Required
        [
          "-----------------------------------------------",
          "-----------------------------------------------",
          "-----------------------------------------------",
          "-----------------------------------------------",
          "-----------------------------------------------",
          "++++++++++++++++++++-++++++++++++++++++++++++++",
          "++++++++++++++++++++++++++++++++++++++++++++++++"
        ]
      }
   }
Description:
   Get schedules by channel and group.
```

---

## 6.4 Get Schedule by group Multi Channels

```

Url:
http://ip:port/schedule/getSchedulesByGroupMultiChannels?ChannelID={ChannelID,,,}&GroupID={GroupID}
Method: GET
Parameter:
      ChannelID: Which channels to get
      GroupID: Which group to get. Record, Snapshot, Email, FTP or Alarm

Return: JSON string
   [
      {
        channel:   Number |  required | channel ID
        group:     String |  required | group ID
        schedules: {
          Normal: Onject | Required when the Group is Record, Snapshot and FTP
          {
            Map:   object  | Required
            [                                   //array has to be 7 lines to match 7 days a week, first line is Sunday
              "-----------------------------------------------",       // '-' is not scheduled, '+' is scheduled
              "-----------------------------------------------",       // note count would be 48 for each day
              "-----------------------------------------------",
              "-----------------------------------------------",
              "-----------------------------------------------",
              "++++++++++++++++++++-++++++++++++++++++++++++++",
```

```
    "++++++++++++++++++++++++++++++++++++++++++++++++++"
  ]
}
```

Motion: Object | Required when the Group is Record, Snapshot, Email and FTP

```
{
  Map:   object  | Required
  [
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "++++++++++++++++++++-++++++++++++++++++++",
    "++++++++++++++++++++++++++++++++++++++++++++++++++"
  ]
}
```

InputAlarm: Object | Required when the Group is Record, Snapshot, Email and FTP

```
{
  Map:   object  | Required
  [
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "++++++++++++++++++++-++++++++++++++++++++",
    "++++++++++++++++++++++++++++++++++++++++++++++++++"
  ]
}
```

Intellegent: Onject | Required when the Group is Record, Email and FTP

```
{
  Map:   object  | Required
  [
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "++++++++++++++++++++-++++++++++++++++++++",
    "++++++++++++++++++++++++++++++++++++++++++++++++++"
  ]
}
```

Alarm: Onject | Required when the Group is Alarm

```
{
  Map:   object  | Required
  [
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
    "----------------------------------------",
```

```
            "++++++++++++++++++++-++++++++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
        FtpUpload: Onject | Required when the Group is Alarm
        {
          Map:   object  | Required
          [
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
            "----------------------------------------------",
            "++++++++++++++++++++-++++++++++++++++++++++++++++",
            "++++++++++++++++++++++++++++++++++++++++++++++++++"
          ]
        }
      }
    }
  }
Description:
  Get schedules by multi channels and group.
```

---

# 7. Network setting

---

## 7.1 Get network setting

```
URL: http://ip:port/network/getNetwork
Method:
  GET
Parameter: None
Return: JSON format object
  {
    "pppoe":
    {
      "active": false,
      "username": "",
      "password": "",
      "interface": "enp2s0",
    },
    "interfaces": [
      {
        "device": "enp2s0",
        "type": "ethernet",    String | This interface is an ethernet
        "dhcp": true,
        "address": "192.168.33.165",
```

```
            "netmask": "255.255.255.0",
            "gateway": "192.168.33.254",
            "mac": "40:8d:5c:2b:16:23"
        },
        {
            "interface": "wlx000e8e9728a2",
            "type": "wifi",       String | This interface is an wireless interface
            "dhcp": true,
            "address": "",
            "netmask": "",
            "gateway": "",
            "mac": "00:0E:8E:97:28:A2",
            "Enable": true,       Bool | enable / disable the wifi interface
            "wifiSSID": "",       String | Which wifi ssid is being connected to
            "wifiPwd": "",        String | The password of connected ssid
            "dns1": "",
            "dns2": ""
        },
        {
            "interface": "cdc-wdm0",
            "type": "gsm",        String | This interface is a 3G/4G interface
            "dhcp": true,
            "address": "10.38.36.128",
            "netmask": "255.255.255.0",
            "gateway": "10.38.36.129",
            "mac": "",
            "Enable": true,       Bool | enable / disable the 3G/4G interface
            "APN": "internet",    String | The provider APN
            "Number": "",         String | The provider number
            "User": "",
            "Pwd": "",
            "dns1": "168.95.1.1",
            "dns2": "168.95.192.1"
        }
    ],
    "general": {
        "dnsserver1": "",
        "dnsserver2": "",
        "default_interface": "enp2s0"
    }
  }
Note:
```

---

## 7.2 Scan available wifi devices

```
URL: http://ip:port/network/scanWifiDevices
Method:
  GET
```

Parameter:

Return: JSON format object


```
{
   "wifi-list": [
      {
         "ssid": "N200RE",
         "signal": 100,
         "security": "WPA1"
      },
      {
         "ssid": "odin-ssid",
         "signal": 100,
         "security": "WPA2"
      },
      {
         "ssid": "N200RE",
         "signal": 100,
         "security": "WPA2"
      },
      {
         "ssid": "N200RE",
         "signal": 100,
         "security": "WPA1"
      },
      {
         "ssid": "Ever-meeting-2F_5G",
         "signal": 82,
         "security": "WPA2"
      },
      {
         "ssid": "ISS-Lab",
         "signal": 79,
         "security": "WPA1"
      },
      {
         "ssid": "Ever-OBM_5G",
         "signal": 79,
         "security": "WPA2"
      },
      {
         "ssid": "Ever-MCC_5G",
         "signal": 77,
         "security": "WPA2"
      },
      {
         "ssid": "TOTOLINK_A3002RU",
         "signal": 70,
         "security": "WPA1"
      },
      {
```

```
        "ssid": "Ever-ODM_5G",
        "signal": 70,
        "security": "WPA2"
      }
   ]
}
```

---

## 7.3 Clear the wifi device that has been recorded

```
URL: http://ip:port/network/forgetWifi
Method:
   GET
Parameter: JSON format object
[
   {
     "wifiSSID": "Ever-MCC_5G"
   },
   {
     "wifiSSID": "Ever-ODM_2.4G"
   },
   {
     "wifiSSID": "Ever-OBM_2.4G"
   }
]

Return:


Note:
```

---

## 7.4 Get 3G/4G a average download / upload bitrate

```
URL: http://ip:port/network/gsmBitrate
Method:
   GET
Parameter: JSON format object

Return:
{
   "upload": "0.22 Kb/s",
   "download": "0.34 Kb/s"
}

Note:
```

```

---

## 7.5 Get POE network interface status

```
URL: http://ip:port/network/getPOEStates
Method:
  GET
Parameter: JSON format object

Return:
[
  {
    "interface": "enp5s0",  string | the network interface name
    "on": false          Bool | false = no internet connection
  },
  {
    "interface": "enp6s0",
    "on": false
  },
  {
    "interface": "enp7s0",
    "on": false
  },
  {
    "interface": "enp10s0",
    "on": true          Bool | true = has internet connection
  }
]
Note:
```

---

## 7.6 Set network setting

```
URL: http://ip:port/network/setNetwork
Method:
  POST
Parameter:
  POST a JSON ScheduleSetting object.
  {
    "pppoe": {
      "active": false,
      "username": "",
      "password": "",
      "interface": "enp2s0",
    },
    "interfaces": [
```
```

```
        {
          "device": "enp2s0",
          "type": "ethernet",     String | This interface is an ethernet
          "dhcp": true,
          "address": "192.168.33.165",
          "netmask": "255.255.255.0",
          "gateway": "192.168.33.254",
          "mac": "40:8d:5c:2b:16:23"
        },
        {
          "interface": "wlx000e8e9728a2",
          "type": "wifi",        String | This interface is an wireless interface
          "dhcp": true,
          "address": "",
          "netmask": "",
          "gateway": "",
          "mac": "00:0E:8E:97:28:A2",
          "Enable": true,        Bool | enable / disable the wifi interface
          "wifiSSID": "",        String | Which wifi ssid is being connected to
          "wifiPwd": "",         String | The password of connected ssid
          "dns1": "",
          "dns2": ""
        },
        {
          "interface": "cdc-wdm0",
          "type": "gsm",         String | This interface is a 3G/4G interface
          "dhcp": true,
          "address": "10.38.36.128",
          "netmask": "255.255.255.0",
          "gateway": "10.38.36.129",
          "mac": "",
          "Enable": true,        Bool | enable / disable the 3G/4G interface
          "APN": "internet",     String | The provider APN
          "Number": "",          String | The provider number
          "User": "",
          "Pwd": "",
          "dns1": "168.95.1.1",
          "dns2": "168.95.192.1"
        }
      ],
      "general": {
        "dnsserver1": "",
        "dnsserver2": "",
        "default_interface": "enp2s0"
      }
    }
```

Return: JSON format object

```
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
```

```
  }
Note:
```

---

## 7.7 Get DDNS setting

```
URL: http://ip:port/network/getDDNS
Method:
  GET
Parameter: None
Return: JSON format object
  {
    "active": true,
    "mac": "40:8d:5c:2b:16:23",
    "server": "EverfocusDDNS",
    "domain": "xnvr",
    "port": 1680,
    "status" : "DDNS_UPDATE_OK"
  }
Note:
```

---

## 7.8 Set DDNS setting

```
URL: http://ip:port/network/setDDNS
Method:
  POST
Parameter:
  POST a JSON ScheduleSetting object.
  {
    "active": true,
    "domain": "xnvr",
    "port": 1680
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
Note:
```

---

## 7.9 Get Port

```
URL: http://ip:port/network/getPort
```

Method:
GET
Parameter: None
Return: JSON format object
{
"http_port": http_port,
"rtsp_port": rtsp_port,
"https_port": https_port,
}
Note:
```

---

## 7.10 Set Port

```
URL: http://ip:port/network/setPort
Method:
POST
Parameter:
POST a JSON ScheduleSetting object.
{
"http_port": http_port,
"rtsp_port": rtsp_port,
"https_port": https_port,
}
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}
Note:
```

---

## 7.11 Get EMail setting

```
URL: http://ip:port/network/getEMail
Method:
GET
Parameter: None
Return: JSON format object
{
"active": true,
"encryption": "Disbale",
"smpt_port": 25,
"smpt_server": "email.server",
"username": "username",

```
    "password": "password",
    "sender": "sender@email.address,
    "receiver1": "receiver1@email.address",
    "receiver2": "receiver2@email.address",
    "receiver3": "receiver3@email.address",
    "interval = 300
  }
Note:
  encryption have three values "Disable", "Enable" or "Auto"
  "Enable" means communicate with mail server via SSL/TLS secure protocol,
  "Auto" means communicate with mail server via SSL/TLS secure protocol when port is 465 or 587
```

---

## 7.12 Test EMail setting

```
URL: http://ip:port/network/testEMail
Method:
  POST
Parameter:
  POST a JSON ScheduleSetting object.
  {
    "active": true,
    "encryption": "Disbale",
    "smpt_port": 25,
    "smpt_server": "email.server",
    "username": "username",
    "password": "password",
    "sender": "sender@email.address,
    "receiver1": "receiver1@email.address",
    "receiver2": "receiver2@email.address",
    "receiver3": "receiver3@email.address",
    "interval = 300
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
Note:
  Send a test email via desired setting.
```

---

## 7.13 Set EMail setting

```
URL: http://ip:port/network/setEMail
Method:
  POST
```

Parameter:
POST a JSON ScheduleSetting object.
{
"active": true,
"encryption": "Disbale",
"smpt_port": 25,
"smpt_server": "email.server",
"username": "username",
"password": "password",
"sender": "sender@email.address,
"receiver1": "receiver1@email.address",
"receiver2": "receiver2@email.address",
"receiver3": "receiver3@email.address",
"interval = 300
}
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}
Note:
```
---

# 8. HTTPS setting

---

## 8.1 Get Certificate information

```
URL: http://ip:port/https/getCertificateInfo
Method:
GET
Parameter: None
Return: JSON format object
{
"subject": {
"countryName": "TW",
"stateOrProvinceName": "New Taipei City",
"localityName": "Sheng Keng",
"organizationName": "Everfocus Electronic Corp.",
"organizationalUnitName": "R&D Division",
"commonName": "www.everfocus.com.tw",
"emailAddress": "receiver@email.address"
},
"issuer": {
"countryName": "TW",
"stateOrProvinceName": "New Taipei City",
"localityName": "Sheng Keng",

```
    "organizationName": "Everfocus Electronic Corp.",
    "organizationalUnitName": "R&D Division",
    "commonName": "www.everfocus.com.tw",
    "emailAddress": "receiver@email.address"
  },
  "notBefore": "2019-12-10T03:37:24.000Z",
  "notAfter": "2029-12-07T03:37:24.000Z",
  "serial": "14ABDBF9DB160D66B709928BC8F8A837CA92AB85"
}
Note:
```

---

## 8.2 Generate self-signed certificate

```
URL: http://ip:port/https/generateCertificate
Method:
  POST
Parameter:
  POST a JSON ScheduleSetting object.
  {
    "country_name": "TW",                    // Country Name
    "state_or_province_name": "New Taipei City",    // State or Province Name:
    "locality_name": "Sheng Keng",              //  Locality Name
    "organization_name": "Everfocus Electronic Corp.", // Organization Name
    "organization_unit_name": "R&D Division",       // Organization Unit Name
    "common_name": "www.everfocus.com.tw",          // Common Name
    "email_address": "receiver@email.address",      // Email Address
    "validity_days": "3650"                  // Validity, default 3650 days
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
Note:
```

---

## 8.3 Upload certificate

```
URL: http://ip:port/https/uploadCertificate
Method:
  POST
Parameter:
  multipart/form-data
    field name="key":   PEM format private key
```

field name="cert": PEM format certificate
Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}
Note:
```
---

# 9. Storage

---

## 9.1 Get USB Storage

```
URL: http://ip:port/storage/getUSBStorages
Method:
GET
Parameter:
Return: JSON format object array
[{
"model": "XXX",
"size": "32.5G",
"mountpoint": "/mnt/sdb",
"fstype": "vfat",
"label": "MYDISK"
}]
Note:
```
---

## 9.2 Set disk group setting

```
URL: http://ip:port/storage/setDiskGroups
Method:
POST
Parameter:
[
{
"name": "DiskGroup1",
"storages": ["HDD-1", "HDD-2"],
"rec_channels": [1,2,3,4]
},
{
"name": "DiskGroup2",
"storages": ["HDD-3"],

```
  "rec_channels": [5,6]
 },
 {
  "name": "DiskGroup3",
  "storages": ["HDD-4"],
  "rec_channels": [7,8]
 }
 ........
]
```

Return: JSON format object array

Note:
```
---

## 9.3 Get disk group setting

```
URL: http://ip:port/storage/getDiskGroups
Method:
   GET
Parameter:

Return: JSON format object array
[
 {
  "name": "DiskGroup1",
  "storages": ["HDD-1", "HDD-2"],
  "rec_channels": [1,2,3,4]
 },
 {
  "name": "DiskGroup2",
  "storages": ["HDD-3"],
  "rec_channels": [5,6]
 },
 {
  "name": "DiskGroup3",
  "storages": ["HDD-4"],
  "rec_channels": [7,8]
 }
 ........
]

Note:
```
---
```

## 9.4 Get Storage setting

```
URL: http://ip:port/storage/getStorages
Method:
  GET
Parameter:
Return: JSON format object array
  [{
    "active": true,
    "device": "/dev/sdb",
    "model": "WDC WD10EURX-63C",
    "serial": "WD-WMC4J0231397",
    "size": "931.5G",
    "mountpoint": "/mnt/sdb",
    "fstype": "ext4",
    "slot": "HDD-1"
    "group": 1
  }]
Note:
```

---

## 9.5 Set Storage setting

```
URL: http://ip:port/storage/setStorages
Method:
  POST
Parameter:
Return: JSON format object array
  [{
    "active": true,
    "device": "/dev/sdb",
    "group": 1
  }]
Note:
```

---

## 9.6 Get Storage S.M.A.R.T. information

```
URL: http://ip:port/storage/smart?device={device}
Method:
  GET
Parameter:
  device: "/dev/sdb"
Note:
```

---

## 9.7 Format Storage

```
URL: http://ip:port/storage/format?device={device}
Method:
  GET
Parameter: Json Array
[
  "/dev/sda",
  "/dev/sdb",
  "/dev/sdc",
  "/dev/sdd"
]

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
Note:
```

---

## 9.8 GetImagingSettings

```
Url: http://ip:port/device/GetImagingSettings?ChannelID={ChannelID}&mainsub={mainsub}
Method: GET

Return: JSON string
  {
    "ImagingSettings":{
      "Brightness":      Float number, Optional.  Brightness.
      "ColorSaturation":  Float number, Optional.  ColorSaturation.
      "Contrast":        Float number, Optional.  Contrast.
      "Sharpness":       Float number, Optional.  Sharpness.
    }
  }
Description:
  Get the current imaging setttings on the device.

```

---

## 9.9 GetImagingOptions

```
Url: http://ip:port/device/GetImagingOptions?ChannelID={ChannelID}&mainsub={mainsub}
```

Method: GET

Return: JSON string

```
{
  "ImagingOptions":{
    "Brightness":      Optional
      {
        Min:       Float Number, minimum for Briaghtness setting.
        Max:       Float Number, maximum for Briaghtness setting.
      }
    "ColorSaturation":     Optional
      {
        Min:       Float Number, minimum for ColorSaturation setting.
        Max:       Float Number, maximum for ColorSaturation setting.
      }
    "Contrast":     Optional
      {
        Min:       Float Number, minimum for Contrast setting.
        Max:       Float Number, maximum for Contrast setting.
      }
    "Sharpness":    Optional
      {
        Min:       Float Number, minimum for Sharpness setting.
        Max:       Float Number, maximum for Sharpness setting.
      }
  }
}
```

Description:

Get the current imaging setttings options on the device.

```
---
```

## 9.10 SetImagingSettings

```
Url: http://ip:port/device/SetImagingSettings?ChannelID={ChannelID}&mainsub={mainsub}
Method: POST
Parameter: JSON  object
{
  "brightness":      Float number, Optional.  Set Brightness.
  "colorSaturation":  Float number, Optional.  Set ColorSaturation.
  "contrast":        Float number, Optional.  Set Contrast.
  "sharpness":       Float number, Optional.  Set Sharpness.
}

Return: JSON format object
{
  "Result":  "OK", or “Error”
```

"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Set imaging setttings.

```
---

## 9.11 Request Video Clips

```
Url: http://ip:port/record/requestVideoClips?ChannelID={ChannelID}
Method: POST
Parameter: JSON object
{
startTime: String | required | UTC time for start time
endTime: String | required | UTC time for end time
};

Return: JSON string
[
{
startTime: UTC UnixTime
endTime: UTC UnixTime
}
]

Description:
Search video clips with channel, start time, and end time.

```
---

## 9.12 Download Video Clip

```
Url:
http://ip:port/record/backupVideoClip?ChannelID={ChannelID}&stream={stream}&startTime={startTime}&endTime={endTime}
Method: GET
Parameter:
ChannelID: channel ID
stream: 0 is main, 1 is sub
startTime: start time of the clip in UTC Unix time
endTime: end time of the clip in UTC Unix time

Return:
```
```

avi binary data

or
when error

return JSON format object
{
"Result": “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Search snapshot with channel, start time, and end time.

```
---

## 9.13 Set Snapshot Settings

```
Url: http://ip:port/device/setSnapshotSettings
Method: POST
Parameter: JSON object
{
settings: required | Object
[
{
channel: required | channel ID to set.
snapshotSettings: optional | Object
{
Auto: required | Number 1 is to try to snapshot on period automatically. 0 is not.
streamType: required | Number 0 is to snapshot main stream, 1 is to snapshot sub stream
Normal: optional | Object | settings about fetching snapshot when normal times.
{
period: required | Number | period of snapshot fetching timer. If period is equal or less than 0, no fetching snapshot, no timer.
},
Alarm: optional | Object | settings about fetching snapshot during input alarm or motion event issued.
{
period: required | Number | period of snapshot fetching timer. If period is equal or less than 0, no fetching snapshot, no timer.
}
}
}
]
};
```
```

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  Set snapshot settings

```
---

## 9.14 Get Snapshot Settings

```
Url: http://ip:port/device/getSnapshotSettings
Method: GET

Return: JSON string
    same as "Set Snapshot Settings" post data

Description:
  Get snapshot settings
```
---

## 9.15 Set Record Settings

```
Url: http://ip:port/device/setRecordSettings
Method: POST
Parameter: JSON  object
  {
    settings:       required | Object
    [
      {
        channel:            required | String | channel ID to set.
        setting:            optional | Object
        {
          enable:           required | String | 1 to enable record, 0 to diable record
          streamType:       required | String | 0 is the stream type is main, 1 is the stream type is sub,
and 2 is stream type is both main and sub
          enablePreRecord:  required | String | 1 to enable pre-record, 0 to diable pre-record
        }
      }
    ]
  };
```
```

Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Set Record settings

```
---

## 9.16 Get Record Settings

```
Url: http://ip:port/device/getRecordSettings
Method: GET

Return: JSON string
same as "Set Record Settings" post data

Description:
Get Record settings
```
---

## 9.17 Set Alarm Settings

```
Url: http://ip:port/device/setAlarmSettings?alarmType={alarmType}
Method: POST
URL Parameter:
alarmType : required | String | Motion, InputAlarm, GPIO

Parameter: JSON object ( when alarmType = Motion or InputAlarm )
{
settings: required | Object
[
{
channel: required | Number | channel ID to set.
setting: optional | Object
{
switch: Optional | boolean | Only Motion alarm setting has this field. To accept the Motion event or not
buzzer: required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this filed is the seconds the buzzer is to sound.
recordChannel: required | Object
```
```

```
            {
              ON:            required | Number | 1 is the record channel is enabled, 0 is the record channel
is disabled.
              IP:            required | Number array | the IP channels array to be record when alarmType is
active. The entities of the array are the Channel numbers.
            }
            alarmOut:          required | Object
            {
              Pin:           required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
              "camOut": [        required | json array | which camera and its alarmout pin will be triggered.
                {
                  "channel": 2,  required | Number | channel id
                  "outPin": 1    required | Number | the alarm out pin number of this channel
                },
                {
                  "channel": 2,
                  "outPin": 2
                }
              ],
              Time:          required | Number | period of alarm out fetching timer.
            }
            preRecord:         required | Number | Seconds for the pre record.
            postRecord:        required | Number | Seconds for the post record.
            showMessage:       required | Number | 1 is to show message when the alarm type is active,
0 is not to.
            sendEmail:         required | Number | 1 is to send Email when the alarm type is active, 0 is not
to.
            fullScreen:        required | Number | 1 is to switch to full screen when the alarm type is active,
0 is not to.
                                    ftpPicture:                    required | boolean| To upload
snapshot picture to ftp or not
          }
        }
      ]
  };

Parameter: JSON  object ( when alarmType = GPIO )
  {
    settings:      required | Object
    [
      {
        alarmIn:           required | Number | alarm in (DI) pin number to set.
        setting:           optional | Object
        {
          alarmType:         required | Number | 1 = Normally Open, 2 = Normally Close, 3 = off.
          buzzer:            required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and
this filed is the seconds the buzzer is to sound.
          recordChannel:     required | Object
          {
            ON:            required | Number | 1 is the record channel is enabled, 0 is the record channel
is disabled.
```

IP: required | Number array | the IP channels array to be record when alarmType is active. The entities of the array are the Channel numbers.
}
alarmOut: required | Object
{
Pin: required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
"camOut": [ required | json array | which camera and its alarmout pin will be triggered.
{
"channel": 2, required | Number | channel id
"outPin": 1 required | Number | the alarm out pin number of this channel
},
{
"channel": 2,
"outPin": 2
}
],
Time: required | Number | period of alarm out fetching timer.
}
preRecord: required | Number | Seconds for the pre record.
postRecord: required | Number | Seconds for the post record.
showMessage: required | Number | 1 is to show message when the alarm type is active, 0 is not to.
sendEmail: required | Number | 1 is to send Email when the alarm type is active, 0 is not to.
fullScreen: required | Number | 1 is to switch to full screen when the alarm type is active, 0 is not to.
}
}
]
};

Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Set Alarm settings

```

---

## 9.18 Get Alarm Settings

```

Url: http://ip:port/device/getAlarmSettings?alarmType={alarmType}
Method: GET

URL Parameter:
  alarmType :  required | String | Motion, InputAlarm, GPIO

Return: JSON string
    same as "Set Alarm Settings" post data

Description:
  Get Alarm settings
```
---

## 9.19 Get Time Settings

```
Url: http://ip:port/system/getTime
Method: GET

Return: JSON Object
  {
    "time": 1568609673430,
    "localtime": "2019-09-16 12:54:33.430",
    "date_format": 0,
    "time_format": 0,
    "ntp_enable": false,
    "ntp_server": "pool.ntp.org",
    "time_zone": 480,
    "dst_enable": false,
    "dst_mode": 1,
    "dst_offset": 0,
    "dst_start_month": 2,
    "dst_start_date": 1,
    "dst_start_hour": 1,
    "dst_start_minute": 0,
    "dst_start_day": 0,
    "dst_start_week": -1,
    "dst_end_month": 9,
    "dst_end_date": 1,
    "dst_end_hour": 1,
    "dst_end_minute": 0,
    "dst_end_day": 0,
    "dst_end_week": -1
  }

Description:
  Get Time settings
    "time": 1568609673430, // miliseconds since 1970/01/01 00:00:00 UTC
    "localtime": "2019-09-16 12:54:33.430", // local time in string format
    "date_format": 0,
    "time_format": 0,
    "ntp_enable": false,
```
```

```
    "ntp_server": "pool.ntp.org",
    "time_zone": 480,
    "dst_enable": false,
    "dst_mode": 1,        // 0:date_mode, 1:week_mode
    "dst_offset": 0,
    "dst_start_month": 2,  // 0~11
    "dst_start_date": 1,   // 1~31
    "dst_start_hour": 1,
    "dst_start_minute": 0,
    "dst_start_day": 0,    // Sun:0, Mon:1
    "dst_start_week": -1,  // 0: first week, 1:second week, -1:last week
    "dst_end_month": 9,    // 0~11
    "dst_end_date": 1,     // 1~31
    "dst_end_hour": 1,
    "dst_end_minute": 0,
    "dst_end_day": 0,      // Sun:0, Mon:1
    "dst_end_week": -1     // 0: first week, 1:second week, -1:last week
```

---

## 9.20 Set Time Settings

```
Url: http://ip:port/system/setTime
Method: POST

Parameter:
  POST a JSON object.
  {
    "time": 1568609673430,
    "localtime": "2019-09-16 12:54:33.430",
    "date_format": 0,
    "time_format": 0,
    "ntp_enable": false,
    "ntp_server": "pool.ntp.org",
    "time_zone": 480,
    "dst_enable": false,
    "dst_mode": 1,
    "dst_offset": 0,
    "dst_start_month": 2,
    "dst_start_date": 1,
    "dst_start_hour": 1,
    "dst_start_minute": 0,
    "dst_start_day": 0,
    "dst_start_week": -1,
    "dst_end_month": 9,
    "dst_end_date": 1,
    "dst_end_hour": 1,
    "dst_end_minute": 0,
```

```
    "dst_end_day": 0,
    "dst_end_week": -1
  }

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  Set Time settings
    "time": 1568609673430, // miliseconds since 1970/01/01 00:00:00 UTC
    "localtime": "2019-09-16 12:54:33.430",  // local time in string format
    "date_format": 0,
    "time_format": 0,
    "ntp_enable": false,
    "ntp_server": "pool.ntp.org",
    "time_zone": 480,
    "dst_enable": false,
    "dst_mode": 1,        // 0:date_mode, 1:week_mode
    "dst_offset": 0,
    "dst_start_month": 2,  // 0~11
    "dst_start_date": 1,   // 1~31
    "dst_start_hour": 1,
    "dst_start_minute": 0,
    "dst_start_day": 0,    // Sun:0, Mon:1
    "dst_start_week": -1,   // 0: first week, 1:second week, -1:last week
    "dst_end_month": 9,    // 0~11
    "dst_end_date": 1,     // 1~31
    "dst_end_hour": 1,
    "dst_end_minute": 0,
    "dst_end_day": 0,      // Sun:0, Mon:1
    "dst_end_week": -1      // 0: first week, 1:second week, -1:last week

  if both "time" and "localtime" are set, "time" will be ignored.
```

---

## 9.21 Set overwrite

```
Url: http://ip:port/storage/setOverwrite
Method: POST

Parameter: JSON Object
  {
    overwriteDays :        required | Number | Days to overwrite. If 0, it is not to overwrite by days.
  }
Return: JSON format object
```

```
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  Set overwrite days.
```

---

## 9.22 Get overwrite

```
Url: http://ip:port/storage/getOverwrite
Method: GET

Return: JSON format object
  {
    overwriteDays :        required | Number | Days to overwrite. If 0, it is not to overwrite by days.
  }

Description:
  Get overwrite days.
```

---

## 9.23 Set Exception Alarm

```
Url: http://ip:port/system/setExceptionAlarm
Method: POST

Parameter: JSON Object
  {
    DiskFull:{
      ON:              required | Number | 1 is the exception DiskFull is enabled, 0 is the exception DiskFull is disabled.
      buzzer:          required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this field is the seconds the buzzer is to sound.
      alarmOut:        required | Object
      {
        Pin:           required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
        Time:          required | Number | period of alarm out fetching timer.
      }
      showMessage:     required | Number | 1 is to show message when the alarm type is active, 0 is not to.
      sendEmail:       required | Number | 1 is to send Email when the alarm type is active, 0 is not to.
    },
    DiskError:{
```

ON: required | Number | 1 is the exception DiskError is enabled, 0 is the exception DiskError is disabled.
buzzer: required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this field is the seconds the buzzer is to sound.
alarmOut: required | Object
{
Pin: required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
Time: required | Number | period of alarm out fetching timer.
}
showMessage: required | Number | 1 is to show message when the alarm type is active, 0 is not to.
sendEmail: required | Number | 1 is to send Email when the alarm type is active, 0 is not to.
},
VideoLoss:{
ON: required | Number | 1 is the exception VideoLoss is enabled, 0 is the exception VideoLoss is disabled.
buzzer: required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this field is the seconds the buzzer is to sound.
alarmOut: required | Object
{
Pin: required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
Time: required | Number | period of alarm out fetching timer.
}
showMessage: required | Number | 1 is to show message when the alarm type is active, 0 is not to.
sendEmail: required | Number | 1 is to send Email when the alarm type is active, 0 is not to.
},
};

Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Set exception alarm.
```

---

## 9.24 Get Exception Alarm

```
Url: http://ip:port/system/getExceptionAlarm
Method: GET

Return: JSON format object
same as "Set Exception Alarm" post data

Description:
```
```

Get exception alarm.
```
---

## 9.25 Search Logs

```
Url: http://ip:port/system/searchLogs
Method: POST
Parameter: JSON Object
  {
    type:  required | string| ['All'|'System'|'Configuration'|'Account'|'Alarm'|'Record'|Storage'].  All is to search for logs of all types
    startTime:  required | number| start time, UTC in UNIX Time format, in milisecond. 0 is to search for all logs till endTime.
    endTime:  required | number| end time, UTC in UNIX Time format, in milisecond. 0 is to search for all logs from startTime till now.
  }

Return: JSON format object
  [
    {
      type:  required | string | ['System'|'Configuration'|'Account'|'Alarm'|'Record'|Storage'] type of the log
      description:  required | string | More detail about this log.
      channel:  optional | number | Channel about this log if it is related to a channel.
      createTime:  required | number | create time, UTC in UNIX Time format, in milisecond. create time of the log.
      user:  optional | string | The current user when this log occured.
      hasRecord:  required | number | 1, if there is record when the record occured. 0 is no record.
    }
  ]

Description:
  search logs. the return datas maximum are 4000.
```
---

## 9.26 Add Log

```
Url: http://ip:port/system/addLogs
Method: POST
Parameter: JSON Obje ct
  [
    {
      type:  required | string | ['System'|'Configuration'|'Account'|'Alarm'|'Record'|Storage'] type of the log
      description:  required | string | More detail about this log.
      channel:  optional | number | Channel about this log if it is related to a channel.
```
```

createTime: required | number | create time, UTC in UNIX Time format, in milisecond. create time of the log.
user: optional | string | The current user when this log occured.
hasRecord: required | boolean| If there is record when the record occured.
}
]

Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Add Log to server.
type, desciprtion, related GUI
System: Login, System Startup, Shutdown, Reboot
Configuration:
Delete Channel(Channel.Channel.IP Channels), Add Channel(Channel.Channel.IP Channels), Modify(channel)(Channel.Channel.IP Channels)
Channel FPS Settings(Record.Stream), Record Mode Settings(Alarm.Record.Record), Record Schedule Settings(Alarm.Record.Record Schedule), Snapshot Settings(Alarm.Snapshot),
Motion(Alarm.Motion), Abnormal Settings(Alarm.Exception),
Network Settings(Network.General), Email Setting(Network.Email)
HDD Settings(Device.Disk.Disk.Overwrite),
General(System.General), Video Settings(System.General.Video Output), User Account(System.User Account),
Maintenance(system.Maintenace.Auto Reboot)
Account: Modify User(System.User Account.Permission/User Edit), User Account(System.User Account.Default User)
Alarm: Motion Start, Motion End, Video Loss,
Record: Playback, Record Search
Storage: No.1 HDD Smart Check(Device.SMART.check)

```
---

## 9.27 Manual Record

```
Url: http://ip:port/record/manualRecord
Method: POST
Parameter: JSON Object
{
channel: required | int | channel number
command: required | string| "start" is to start manual record. "stop" is to stop manual record
}

Return: JSON format object

```
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
```

Description:
Command to start or stop manual record.
Server will add meta Type:  'manualStart' and 'manualStop'.
Use command metaSearch to search manualStart and manualStop, which represents the manaul record took place in this period.
The Data field of the command metaSearch result is the ID for manualStart and manualStop.
Like { id: 1564741064}. The matching manualStart and manualStop have the same ID.

```
---
```

## 9.28 Manual Alarm

```
Url: http://ip:port/record/manualAlarm
Method: POST
Parameter: JSON Object
  {
    channel:              required | Number | channel number
    command:              required | string| "start" is to start manual alarm. "stop" is to stop manual alarm
  }

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
```

Description:
Command to start or stop manual alarm.
Server will add meta Type:  'manualAlarmStart' and 'manualAlarmStop'.
Use command metaSearch to search manualAlarmStart and manualAlarmStop, which represents the manaul alarm took place in this period.
The Data field of the command metaSearch result is the ID for manualAlarmStart and manualAlarmStop.
Like { id: 1564741064}. The matching manualAlarmStart and manualAlarmStop have the same ID.

```
---
```

# 10. Account setting

---

## 10.1 Get Login parameters

```
Url: http://ip:port/admin/login
Method: GET
Return: JSON format object
  {
    realm: 'xnvr',
    qop: 'auth',
    nonce:                A random string for
    opaque: '',
    change_pwd: first
  }

Description:
  Get parameters for login.
```

---

## 10.2 Login

```
Url: http://ip:port/admin/login
Method: POST
Parameter: JSON Object
  {
    username:
    nc:
    cnonce:
    response:
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
          If the Result is OK, then the Description is userID=xxx.
    "option":     optional. If the login is by temperarly password, then has this field. And the value is
needToChangePasswd. Like this:   "option":"needToChangePasswd"
  }

Description:
  If return is Result OK and has option of "needToChangePasswd". Client has to show message to inform
the user to change password soon.

```

---

## 10.3 Logout

```
Url: http://ip:port/admin/logout
Method: GET
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  Get parameters for login.
```

---

## 10.4 Set administrator account

```
Url: http://ip:port/admin/setAdminAccount
Method: POST
Parameter: JSON Object
  {
    username:
    password:
  }
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  Get parameters for login.
```

---

## 10.5 Get user list

```
Url: http://ip:port/account/userList
Method: GET
Return: JSON format array
  [{
      userID: ,
      active: true or false,
      username: ,
      password: true or false
   }
  ]
```

Description:

```
---

## 10.6 Get user

```
Url: http://ip:port/account/getUser?userID={userID}
Method: GET
Return: JSON format array
  {  userID: ,
     active: true or false,
     username: ,
     password: ,
     permissions: {
        live_on
        live
        playback_on
        playback
        ptz_on
        ptz_control
        copy_on
        copy
        log_search
        parameter
        system_maintenance
        manual_record
        manage_disk
        remote_login
        sequence_control
        manual_snapshot
     }
  }
```

Description:

```
---

## 10.7 Set user

```
Url: http://ip:port/account/setUser?userID={userID}
Method: POST
Parameter: JSON Object
  [{
        userID: ,
        active: true or false,
        username: ,
        password: ,
```

```
        permissions: {
          live_on
          live
          playback_on
          playback
          ptz_on
          ptz_control
          copy_on
          copy
          log_search
          parameter
          system_maintenance
          manual_record
          manage_disk
          remote_login
          sequence_control
          manual_snapshot
        }
    }
  ]
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
  }
Description:

```

---

## 10.8 Shutdown

```
Url: http://ip:port/system/shutdown
Method: get
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
  }
Description:
  Shutdown the system

```

---

## 10.9 Reboot

```
Url: http://ip:port/system/reboot
```

Method: get
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
Description:
  Reboot the system

```
---

## 10.10 LoadDefault

```
Url: http://ip:port/system/loadDefault
Method: post
Parameters: JSON format object
  [
    categories to do load default, examples: channels,record,alarm,network,device,system
  ]
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  To load default of the config. It will reboot the system after load default.

```
---

## 10.11 Backup config file

```
Url: http://ip:port/system/backupConfig
Method: get

Return:
  text , config file

Description:
  To backup the current config file.

```
---
```

## 10.12 Restore config file

```
Url: http://ip:port/system/restoreConfig
Method: post
  Upload file.

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
  To upload config file. It will reboot the system after load default.
```

---

## 10.13 Listener to Exception

```
Websocket
Url: ws://ip:port/system/exceptionListen
Message: JSON format object array
  {
    "exception":   exception   (DiskFull, DiskError)
    "target":      Optional. If exception is DiskError, then the target is the disk name. If exception is DiskFull, there is no target.
    "time":        UTC in UNIX Time format, in milisecond. time of the exception.
  }
Description:
  Exception sent when exception took place.
```

---

## 10.14 Listener to Message

```
Websocket
Url: ws://ip:port/system/messageListen
Message: JSON format object array
  {
    "message":     Required. messages : "network", "cameraSubnetMask", "cameraDetail", "portList", "sysTimeChanged"
    "time":        Required. UTC in UNIX Time format, in milisecond. time of the message.
    "target":      Optional.
                for message "cameraSubnetMask", the target is channel ID
                for message "cameraDetail", the target is channel ID
    "content":     Optional. Object.
```

for message "network", the content is same as cgi getNetwork return.
for message "cameraSubnetMask", the content is {"subnetMask":"255.255.255.0"}
for message "cameraDetail", the content is same as cgi cameraDetail return.
for message "portList", the content is same as cgi getPort return.
for message "sysTimeChanged", the content is {"sysTimeChanged": true }
}
Description:
Notify message to clients.
```
---

## 10.15 Get Analytics Motion settings

```
Url: http://ip:port/device/getAnalyticsMotionSettings?ChannelID={ChannelID}&mainsub={mainsub}
Method: get

Return: JSON format object
[
{
channel: channel to set
switch: switch to have motion event
cellLayout: {
columns: available columnns for the cells layout
rows: available rows for the cells layout
transformation: {
translate: {
x: x translate of the cell layout
y: y translate of the cell layout
},
scale: {
x: x scale of the cell layout
y: y scale of the cell layout
}
}
}

sensitivity: sensitivity (0~100)
cellSettings: Bits represent Cells are defined by columns and rows of cellLayout in the analytics module(CellMotionEngine).
From left to right, top to bottom. 0 is off, 1 is on. Then the cellSettings are these bits encoded as Hex Decimal string.
If the number of cells is not a multiple of 8 the last byte is filled with zeros.
}
]

Description:
To get analytics motion settings.

If return channel and switch only, without CallLayout, sensitivity and cellSettings, it means the channel does not support the analytics module for cell motion settings.

```
---

## 10.16 Set Analytics Motion

```
Url: http://ip:port/device/setAnalyticsMotionSettings
Method: post

Parameters: JSON format object
  [
    {
      channel:          channel to set
      mainsub:          mainsub  (0 is main, 1 is sub)
                switch:                    switch to have motion event
      sensitivity:      sensitivity to set (0~100)
      cellSettings:     Bits represent Cells are defined by columns and rows of cellLayout in the analytics module(CellMotionEngine).
                From left to right, top to bottom. 0 is off, 1 is on. Then the cellSettings are these bits encoded as Hex Decimal string.
                If the number of cells is not a multiple of 8 the last byte is filled with zeros.
    }
  ]

Return: JSON format object
  [
    {
      channel:          channel to set
      moduleResult:     set module (sensitivity) result. SUCCESS or FAIL
      cellResult:       set cell (cellSettings) result.  SUCCESS or FAIL
      description:      Description for module results and cell results if there is error.
    }
  ]

Description:
  To set switch and analytics motion if the camera supports analytics module and cell motion settings.
    Or to set switch only if the camera does not support analytics module and cell motion settings.

```
---

## 10.17 Get system info

```

Url: http://ip:port/system/getSystemInfo
Method: get

Return: JSON format object
```
{
   "analog_channels":          number, count of analog channels
   "ipcam_channels":           number, count of ipcam channels,
   "device_name":              string, device name
   "device_id":                string, device id
   "software_version":         string, software version
   "model":                    string, model name
   "serial":                   string, serial no
}
```

Description:
To get system info

```
---

## 10.18 Set system info

```
Url: http://ip:port/system/setSystemInfo
Method: post

Parameters: JSON format object
```
{
   "device_name":              string, device name
   "device_id":                string, device id
}
```

Return: JSON format object
```
{
   "Result":  "OK", or “Error”
   "Description":
        If the "Result" is “Error”, the description is about the error.
}
```

Description:
To set system info

```
---

## 10.19 get GPIO information

```
Url: http://ip:port/dio/GetGPIOInfo

Method: get

Parameters:
none

Return: JSON format object
```
[
  {
    ch: 0,
    alarmIn: [1,2,3,4],
    alarmOut: [5,6,7,8]
  },
  {
    ch: 1,
    alarmIn: [1],
    alarmOut: [1]
  },
  {
    ch: 2,
    alarmIn: [1],
    alarmOut: [1,2]
  }
]
```

Description:
To get the GPIO information of this machine.

```
---

## 10.20 Get alarm in/out pin status

```
Url: http://ip:port/dio/GetPinStatus
Method: get

Parameters:
none

Return: JSON format object
{
  "pin_value": [        Number array | current di/do pin value
    {
      "pinNum": 1,    Number | indicate the number of pin.
      "pinVal": 0     Number | indicate the current value of this pin.
    },
    {
      "pinNum": 2,
      "pinVal": 0
    },
```
```

```
        {
          "pinNum": 3,
          "pinVal": 0
        },
        {
          "pinNum": 4,
          "pinVal": 0
        },
        {
          "pinNum": 5,
          "pinVal": 0
        },
        {
          "pinNum": 6,
          "pinVal": 0
        },
        {
          "pinNum": 7,
          "pinVal": 0
        },
        {
          "pinNum": 8,
          "pinVal": 0
        }
    ]
  }
```

Description:
To get the GPIO information of this machine.

```
---
```

## 10.21 Pin Manual Setting

```
Url: http://ip:port/dio/pinManualSetting
Method: POST
Parameter: JSON Object
  {
    pinID:              required | Number | ID of Pin
    pinValue:           required | Number | value of this pin to set. 0 is off, 1 is on.
  }

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
```

Description:
Command to set value of the Do Pin.

```
---

## 10.22 Get xFleet config

```
Url: http://ip:port/ttxnet/getConfig
Method: get

Parameters:
none

Return: JSON format object
{
"enable": true,
"DeviceID": "30005",
"ServerIP": "172.20.4.169",
"ServerPort": "6608"
}

Description:

```
---

## 10.23 Set xFleet config

```
Url: http://ip:port/ttxnet/setConfig
Method: POST
Parameter: JSON Object
{
"enable": true,
"DeviceID": "30005",
"ServerIP": "172.20.4.169",
"ServerPort": "6608"
}

Return: JSON format object
{
"Result":  "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
```
```

```
---
```

## 10.24 Set GPS Event Settings

```
Url: http://ip:port/device/setGPSEventSettings
Method: POST

Parameter: JSON  object
        {
                "OverSpeed":{
                        switch:                                 required | boolean
                        limit:                                  optional | Number | kmph or mph
                        unit:                                   optional | string | "kmph" or "mph"
                        overTime:                               optional | Number | over time in
seconds
                        buzzer:         required | Number | 0 to disable buzzer. If bigger than 0, the
buzzer is on and this filed is the seconds the buzzer is to sound.
                        sendEmail:      required | Number | boolean
                        sendToXfleet:   required | Number | boolean
                        alarmOut:               required | Object
                        {
                                Pin:    required | Number array | the alarm out (DO) pin array to
trigger alarm out (DO).
                                Time:           required | Number | period of alarm out fetching
timer.
                        }
                },
                "LowSpeed":{
                        switch:                                 required | boolean
                        limit:                                  optional | Number | kmph or mph
                        unit:                                   optional | string | "kmph" or "mph"
                        overTime:                               optional | Number | over time in
seconds
                        buzzer:         required | Number | 0 to disable buzzer. If bigger than 0, the
buzzer is on and this filed is the seconds the buzzer is to sound.
                        sendEmail:      required | Number | boolean
                        sendToXfleet:   required | Number | boolean
                        alarmOut:               required | Object
                        {
                                Pin:    required | Number array | the alarm out (DO) pin array to
trigger alarm out (DO).
                                Time:           required | Number | period of alarm out fetching
timer.
                        }
                },
                "LongParking":{
                        switch:                                 required | boolean
```

overTime: optional | Number | over time in seconds
buzzer: required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this filed is the seconds the buzzer is to sound.
sendEmail: required | Number | boolean
sendToXfleet: required | Number | boolean
alarmOut: required | Object
{
Pin: required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
Time: required | Number | period of alarm out fetching timer.
}
}
}

Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Set GPS event settings

```
---

## 10.25 Get GPS Event Settings

```
Url: http://ip:port/device/getGPSEventSettings
Method: GET

Return: JSON string
same as "Set GPS Event Settings" post data


Return: JSON format object
{
"Result": "OK", or “Error”
"Description":
If the "Result" is “Error”, the description is about the error.
}

Description:
Get GPS event settings

```
```

---

## 10.26 Set GSensor Event Settings

```
Url: http://ip:port/device/setGSensorEventSettings
Method: POST

Parameter: JSON  object
        {
                "RapidAcceleration":{
                        switch:                                 required | boolean
                        limit:
        {
          x:
          y:
          z:
        }
                        buzzer:                 required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this filed is the seconds the buzzer is to sound.
                        sendEmail:              required | Number | boolean
                        sendToXfleet:           required | Number | boolean
                        alarmOut:                       required | Object
                        {
                                Pin:            required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
                                Time:                   required | Number | period of alarm out fetching timer.
                        }
                },
                "RapidDeceleration":{
                        switch:                                 required | boolean
                        limit:
        {
          x:
          y:
          z:
        }
                        buzzer:                 required | Number | 0 to disable buzzer. If bigger than 0, the buzzer is on and this filed is the seconds the buzzer is to sound.
                        sendEmail:              required | Number | boolean
                        sendToXfleet:           required | Number | boolean
                        alarmOut:                       required | Object
                        {
                                Pin:            required | Number array | the alarm out (DO) pin array to trigger alarm out (DO).
                                Time:                   required | Number | period of alarm out fetching timer.
                        }
```

```
            },
            "Collision":{
                switch:                                    required | boolean
                limit:
    {
      x:
      y:
      z:
    }
                buzzer:          required | Number | 0 to disable buzzer. If bigger than 0, the
buzzer is on and this filed is the seconds the buzzer is to sound.
                sendEmail:       required | Number | boolean
                sendToXfleet:    required | Number | boolean
                alarmOut:              required | Object
                {
                    Pin:         required | Number array | the alarm out (DO) pin array to
trigger alarm out (DO).
                    Time:            required | Number | period of alarm out fetching
timer.
                }
            },
            "Sharp":{
                switch:                                    required | boolean
                limit:
    {
      x:
      y:
      z:
    }
                buzzer:          required | Number | 0 to disable buzzer. If bigger than 0, the
buzzer is on and this filed is the seconds the buzzer is to sound.
                sendEmail:       required | Number | boolean
                sendToXfleet:    required | Number | boolean
                alarmOut:              required | Object
                {
                    Pin:         required | Number array | the alarm out (DO) pin array to
trigger alarm out (DO).
                    Time:            required | Number | period of alarm out fetching
timer.
                }
            }
     }
```

Return: JSON format object

```
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }
```

Description:

Set GSensor event settings

```
---

## 10.27 Get GSensor Event Settings

```
Url: http://ip:port/device/getGSensorEventSettings
Method: GET

Return: JSON string
    same as "Set GSensor Event Settings" post data


Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
  }

Description:
        Get GSensor event settings

```
---


---

## 10.28 Set FTP settings

```
Url: http://ip:port/network/setFTP
Method: POST

Parameter: JSON  object
  {
    active:       required| boolean| active or not for FTP
    host:         required| string| host address
    port:         required| string| port of the FTP server
    username:     required| string| user name
    password:     required| string| password
    path:         required| string| path in FTP server to put data
  }

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
```
```

}

Description:
Set FTP settings

```
---

## 10.29 Get FTP settings

```
Url: http://ip:port/network/getFTP
Method: GET

Return: JSON string
    same as "Set FTP settings" post data



Description:
Get FTP settings

```
---

---

## 10.30 Test FTP

```
Url: http://ip:port/network/testFTP
Method: POST

Parameter: JSON object
{
    host:       required| string| host address
    port:       required| string| port of the FTP server
    username:   required| string| user name
    password:   required| string| password
}

Return: JSON format object
{
    "Result": "OK", or “Error”
    "Description":
        If the "Result" is “Error”, the description is about the error.
}

Description:
Set FTP settings

```
```

---

## 10.31 Set IP Filter settings

```
Url: http://ip:port/network/setIPFilter
Method: POST

Parameter: JSON  object
  {
    "active": true,       boolean| active or not for IP Filter
    "mode": 0,            number 0: balacklist, 1: Whitelist
    "blacklist": [        array of ip address to deny access
      "192.168.1.1",            single ip address
      "192.168.2.0/24",         a segment of ip address
      "192.168.3.1-192.168.3.10"  a range of ip address
    ],
    "whitelist": []       array of ip address to allow access
  }

Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
  }

Description:
  Set IP Filter settings
```

---

## 10.32 Get IP Filter settings

```
Url: http://ip:port/network/getIPFilter
Method: GET

Return: JSON string
      same as "Set IP Filter settings" post data



Description:
  Get IP Filter settings
```

---

## 10.33 Call Buzzer

```
Url: http://ip:port/system/callBuzzer?timeout
Method: GET

Parameter: timeout : time (seconds) to Buzzer Maximum limit is 60
Return: JSON format object
  {
    "Result":  "OK", or “Error”
    "Description":
          If the "Result" is “Error”, the description is about the error.
  }



Description:
        call server Buzzer

```

---