HIKCGI Integration Guide

The document describes the method to integrate the Hikvision cameras with HIKCGI protocols. The HIKCGI protocol defines universal interfaces to access the resources of the camera. The document will illustrate how to access and control IP camera based on HIKCGI protocol.

The integration solution includes 5 parts: "*HIKCGI integration Guide*"," *RTSP Interface and Development Guide*" "*HIKCGI Event Function*"," *HIKCGI PTZ Function*" and "*HIKCGI Image Display Function*".

"*HIKCGI integration Guide*" is a fundamental part of the integration solution. The document mainly describes the principle of HIKCGI protocol, operation mechanism and development of basic functions, such as system maintenance, video parameter control, development of network functions and alarm input /output control. This document is the basis of "*HIKCGI event function*", "*HIKCGI PTZ Function*" and "*HIKCGI Image Display Function*".

"*RTSP Interface and Development Guide*" mainly describes how to use RTSP protocol to get the stream from the device.

"*HIKCGI Event Function*" is such a document that describes development of event function. The events mainly include Motion Detection, Alarm Output and Video Tamper and so on.

"*HIKCGI Image Display Function*" is such a document that describes how to integrate front-end image parameters and image display operation. The main content includes: Text Overlay Display, Front-end parameters configuration (Brightness, Hue, Contrast, Saturation and Day/Night auto switch) and so on.

"*HIKCGI PTZ Function*" is such a document that describes how to integrate PTZ control function, and the main content is to introduce some corresponding PTZ commands.

Contents

1 Introduction of HIKCGI Protocol

1.1 Overview of HIKCGI Protocol

HIKCGI protocol is a set of CGI interfaces based on HTTP protocols. HIKCGI protocol supports 4 methods defined by HTTP protocol: GET, PUT, POST and DELETE.

GET method: request a representation of the specified resource. Requests using GET method should only retrieve data and should have no other effect. If requests are right, resource information will be returned according to format of protocol.

PUT method: update a specific resource or reset a specific resource. PUT method will change the resource, and you can check specific information of the operation in the "ResponseStatus" page.

POST method: this may result in the creation of a new resource. The device will create a new resource using these parameters passed by POST method and return ID of the new resource. The response of creating a resource is the same as PUT method, and the result of the operation is shown in the “ResponseStatus” page. In addition, POST method supports to search some complicated requests, you can put them into HTTP Body and send them by POST method.

DELETE method: delete the specific resource. Before using DELETE method, the specific resource must exist; otherwise the operation will be failed. The result of the operation is shown in the "ResponseStatus" page.

1.2 Overview of HTTP protocol

HIKCGI protocol is based on HTTP protocol, so it is necessary to understand what HTTP protocol is.

HTTP (Hypertext Transfer Protocol) is character-oriented protocol. It includes two parts: HTTP Header and HTTP Body.

HTTP header is components of the message header of requests and responses. It includes HTTP method, URL, content-length and content-type. The content of header is illustrated in separate lines and \r\n or \n indicates the end of a line. The end of the header fields is indicated by an empty field or two continuous “\r\n” or “\n”. Except the first line, other lines of the header fields are used to illustrate HTTP Body.

Example:

(1) Check status information of the device

```
GET /System/deviceInfo HTTP/1.1
Host: 172.8.6.176
Authorization: Basic YWRtaW46MTIzNDU=

```

This is a HTTP request using GET method to get the information of the device. The URL is /System/deviceInfo.

"Host" of the second line indicates that IP address of the destination host is 172.8.6.176.

The third line of the header fields is authentication field. Currently, the authentication encryption method which HIK cameras support is Basic, so it shows Authorization: Basic in this request. In addition, the string "YWRtaW46MTIzNDU=" is a kind of encryption based on base64 for admin (user) and 12345 (password).

Each line of the header fields ends by a"\r\n". The end of the header fields is indicated by an empty field.

The following request is to get information of the device. The camera response as follows:

```
HTTP/1.1 200 OK
Date: Fri, 13 Jul 2012 16:38:16 GMT
Server: App-webs/
Connection: close
Content-Length: 693
Content-Type: application/xml
X-Appweb-Seq: 19697

<?xml version="1.0" encoding="UTF-8"?>
<DeviceInfo version="1.0" xmlns="http://www.hikvision.com/ver10/XMLSchema">
<deviceName>IP CAMERA</deviceName>
…….
</DeviceInfo>
```

The response is made of HTTP header and HTTP body. If the first line of the HTTP header returns 200 and OK, it indicates that the request is executed successfully.

Please refer to Appendix of *HIKCGI Integration Guide* to check the information of status code.

The fifth line of the header fields is Content-Length which indicates the length of HTTP body is 693 bytes.

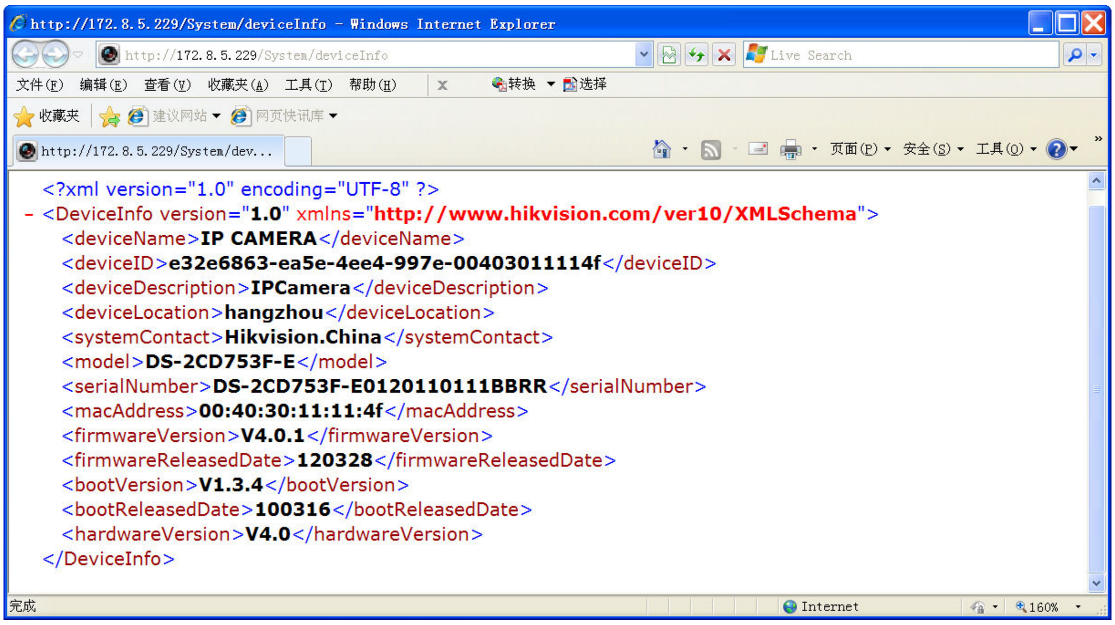
The sixth line of the header fields is Content-Type which indicates the format of

HTTP body is XML format (*Section 8.1* for reference).

HTTP body is about information of the device and adopts XML format. Here is a sample; we don’t list all the content.

GET method is used to get information of the device. In general, requests don’t contain HTTP body, so HTTP Header doesn’t contain Content-Length and Content-Type, either.

In fact, some common browsers support HTTP GET method, so you can input http://ipaddress/System/deviceInfo in the address bar of IE explorer to get the information of the device. If the explorer prompts to input user and password, please input right information. The terminal interface is as follows:


http://172.8.5.229/System/deviceInfo - Windows Internet Explorer
http://172.8.5.229/System/deviceInfo
Live Search
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
转换 选择
收藏夹 建议网站 网页快讯库
http://172.8.5.229/System/dev...
页面(P) 安全(S) 工具(O)
完成
Internet
160%

```
<?xml version="1.0" encoding="UTF-8" ?>
- <DeviceInfo version="1.0" xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <deviceName>IP CAMERA</deviceName>
    <deviceID>e32e6863-ea5e-4ee4-997e-00403011114f</deviceID>
    <deviceDescription>IPCamera</deviceDescription>
    <deviceLocation>hangzhou</deviceLocation>
    <systemContact>Hikvision.China</systemContact>
    <model>DS-2CD753F-E</model>
    <serialNumber>DS-2CD753F-E0120110111BBRR</serialNumber>
    <macAddress>00:40:30:11:11:4f</macAddress>
    <firmwareVersion>V4.0.1</firmwareVersion>
    <firmwareReleasedDate>120328</firmwareReleasedDate>
    <bootVersion>V1.3.4</bootVersion>
    <bootReleasedDate>100316</bootReleasedDate>
    <hardwareVersion>V4.0</hardwareVersion>
  </DeviceInfo>
```

(2) Set the time of IP camera

```
PUT /System/time HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type: text/plain
Host: 172.8.6.176
Content-Length: 19

2012-07-13T17:02:33
```

The format of HTTP body of the header fields is text format, and its length is 19 bytes. After the header, that is a string of time information which needs to be set.

PUT method is used to set the operation. Because in general, PUT method contains HTTP body which contains some parameters needed to be updated or reset, PUT request of the header fields will contain Content-Type and Content-Length.

IP camera responses as shows:

```
       HTTP/1.1 200 OK
       Content-Length: 251
       Content-Type: application/xml

       <?xml version="1.0" encoding="UTF-8"?>
       <ResponseStatus version="1.0" xmlns="
http://www.hikvision.com/ver10/XMLSchema">
       <requestURL>/system/time/localtime</requestURL>
       <statusCode>1</statusCode>
       <statusString>OK</statusString>
       </ResponseStatus>
```

The response is made of HTTP header and HTTP body. If the first line of the HTTP header returns 200 and OK, it indicates that the request is executed successfully.

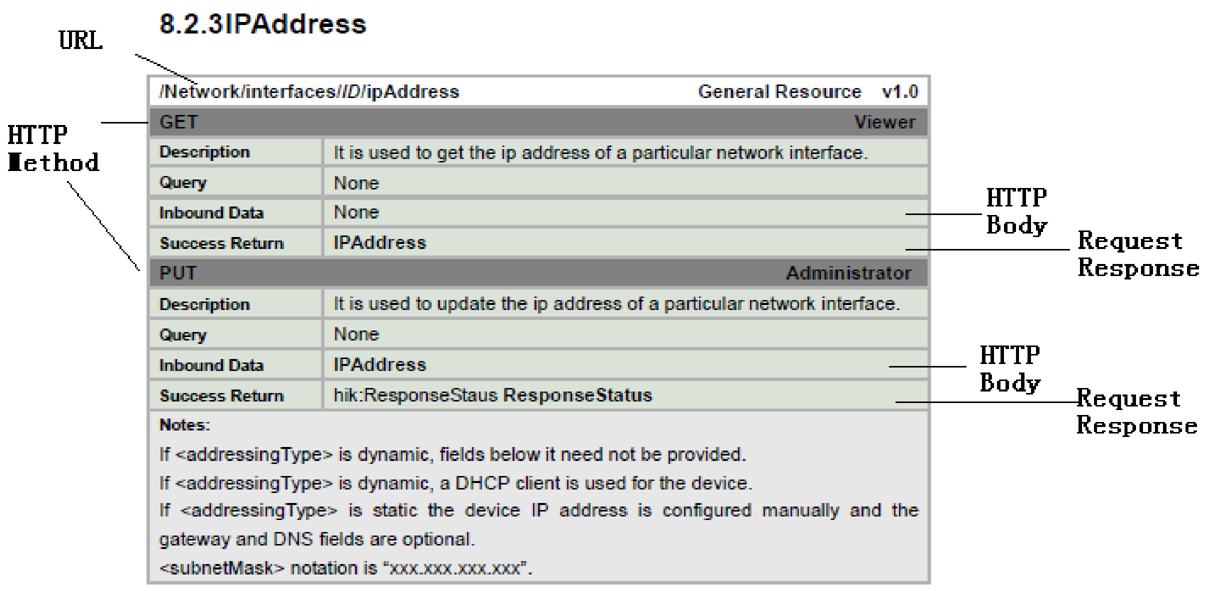
HTTP body belongs to ResponseStatus page. The detailed response information from IP camera is in the ResponseStatus page.

<requestURL> indicates the URL of request.

<statusString> contains the detailed response information.

1.3 How to use HIKCGI Protocol Guide

Each functional module of HIKCGI protocol is called a Service and each service can be further divided into various resources. Please refer to the following diagram:



8.2.3IPAddress

/Network/interfaces/*ID*/ipAddress	General Resource v1.0
GET	Viewer
Description	It is used to get the ip address of a particular network interface.
Query	None
Inbound Data	None
Success Return	IPAddress
PUT	Administrator
Description	It is used to update the ip address of a particular network interface.
Query	None
Inbound Data	IPAddress
Success Return	hik:ResponseStaus ResponseStatus

Notes:
If <addressingType> is dynamic, fields below it need not be provided.
If <addressingType> is dynamic, a DHCP client is used for the device.
If <addressingType> is static the device IP address is configured manually and the gateway and DNS fields are optional.
<subnetMask> notation is “xxx.xxx.xxx.xxx”.

"URL" is the HTTP of the resource.

We can use GET method and PUT method to access this resource. It indicated that not only you retrieve the resource, but also update and reset the resource.

"Description" is description of this method and explains the function of the method.

"Inbound Data" designates whether the method need bring HTTP body, and meanwhile designates the type of HTTP body.

For example, if Inbound Data of GET method is "None", it indicates that it doesn't need HTTP body when you use GET method to access a resource, you only need to construct the header of HTTP and send it to IP camera; Inbound Data of PUT method is "IP Address", it indicates that PUT method needs to bring HTTP body and the content of HTTP body must be constructed according to the XML block of "IPAddress" of HIKCGI protocol.

IPAddress XML Block

```
    <IPAddress version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <ipVersion>          <!-- req, xs:string, "v4" -->    </ipVersion>
        <addressingType>    <!-- req, xs:string, "static,dynamic" -->
</addressingType>
        <ipAddress>          <!-- req, xs:string -->                    </ipAddress>
   <subnetMask>       <!-- req, xs:string, subnet mask for IPv4 address -->
   </subnetMask>
        <DefaultGateway>   <!-- dep -->
           <ipAddress>       <!-- req, xs:string -->       </ipAddress>
        </DefaultGateway>
        <PrimaryDNS>       <!-- dep -->
           <ipAddress>       <!-- req, xs:string -->       </ipAddress>
        </PrimaryDNS>
    </IPAddress>
```

Note:

XML block defines the attribute of each tag in detail. Within each tag, "req" means the tag is necessary, you need to construct the tag when sending PUT request; "opt" means that the tag is optional (*Section 6.3* for reference). In addition, as for tag with "opt", if PUT request contains this tag, the value of tag will be updated or reset, or if PUT request doesn't contain this tag, the value will keep the same.

2 Getting Capabilities of the device

When accessing to a device, the first step is to get capabilities of the device, it mainly includes: Device Basic Information, Hardware Ability Information and so on.

Hardware Ability Information mainly includes Alarm Input, Alarm Output, the number of Audio Channel, the number of Video Channel and PTZ information.

2.1 Getting basic information of the device

The basic information of the device includes device type, device model and serial number and so on.

Command: GET /System/deviceInfo

Example:

```
GET /System/deviceInfo HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Host:172.8.6.153
Content-Type:text/xml
```

IP camera responses as follows:

```
HTTP/1.1 200 OK
Date: Wed, 28 Mar 2012 10:56:50 GMT
Connection: close
Content-Length: 714
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<DeviceInfo
version="1.0"xmlns="http://www.hikvision.com/ver10/XMLSchema">
<deviceName>IP CAMERA</deviceName>
<deviceID>e32e6863-ea5e-4ee4-997e-00403011114f</deviceID>
<deviceDescription>IPCamera</deviceDescription>
<deviceLocation>hangzhou</deviceLocation>
<systemContact>Hikvision.China</systemContact>
<model>DS-2CD753F-E</model>
<serialNumber>DS-2CD753F-E0120110111BBRR</serialNumber>
<macAddress>00:40:30:11:11:4f</macAddress>
<firmwareVersion>V4.0.1</firmwareVersion>
<firmwareReleasedDate>120328</firmwareReleasedDate>
<bootVersion>V1.3.4</bootVersion>
<bootReleasedDate>100316</bootReleasedDate>
<hardwareVersion>V4.0</hardwareVersion>
```

```
        </DeviceInfo>
```

<deviceName>: the name of the device, editable;

<deviceDescription>: the description of the device, read-only;

<model>: the type of the device;

Note:

Currently, Hikvision describes Network Camera as “IPCamera”, Network Speed Dome as “IPDome” and DVR as “DVRDVS”.

2.2 Getting the hardware information

The functions of the device depend on the hardware of the device. For example, if a device doesn’t support audio input, the device can’t provide video & audio stream, just video stream.

2.2.1 Getting the number of alarm inputs

Command: GET /IO/inputs

Example:

```
        GET /IO/inputs HTTP/1.1
        Host: 172.8.6.228
        Connection: Keep-Alive
        Authorization: Basic YWRtaW46MTIzNDU=
```

IPC responses as follows:

```
        HTTP/1.1 200 OK
        Date: Sat, 14 Jul 2012 05:42:40 GMT
        Server: App-webs/
        Connection: close
        Content-Length: 690
        Content-Type: application/xml

        <?xml version="1.0" encoding="UTF-8"?>
        <IOInputPortList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <IOInputPort version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>1</id>
        <triggering>high</triggering>
        </IOInputPort>
        <IOInputPort version="1.0"
```

```
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>2</id>
        <triggering>high</triggering>
        </IOInputPort>
        <IOInputPort version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>3</id>
        <triggering>high</triggering>
        </IOInputPort>
        <IOInputPort version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>4</id>
        <triggering>high</triggering>
        </IOInputPort>
        </IOInputPortList>
```

<IOInputPortList> is a list. All the alarm inputs are listed in <IOInputPortList> page. Each <IOInputPort> stands for an alarm input. In terms of above example, there are 4 alarm inputs.

<triggering> stands for a level signal which triggers alarm input. If the attribute of <triggering> is “high”, it means that a high level signal would trigger the alarm input event.

If a device doesn’t support alarm input, <IOInputPortList> will be an empty table.

```
   <?xml version="1.0" encoding="UTF-8"?>
   <IOInputPortList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
   </IOInputPortList>
```

As shown above, < IOInputPortList > doesn’t contain any information of <IOInputPort>. It indicates that the device doesn’t support alarm input.

2.2.2 Getting the number of alarm outputs

Command: GET /IO/outputs

Example:

```
   GET /IO/outputs HTTP/1.1
   Host: 172.8.6.228
   Authorization: Basic YWRtaW46MTIzNDU=
```

IP camera responses as follows:

```
HTTP/1.1 200 OK
Server: App-webs/
Connection: close
Content-Length: 883
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<IOOutputPortList version="1.0" 
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<IOOutputPort version="1.0" 
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>1</id>
<PowerOnState>
<defaultState>low</defaultState>
<outputState>pulse</outputState>
<pulseDuration>30000</pulseDuration>
</PowerOnState>
</IOOutputPort>
<IOOutputPort version="1.0" 
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>2</id>
<PowerOnState>
<defaultState>low</defaultState>
<outputState>pulse</outputState>
<pulseDuration>5000</pulseDuration>
</PowerOnState>
</IOOutputPort>
<IOOutputPort version="1.0" 
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>3</id>
<PowerOnState>
<defaultState>low</defaultState>
<outputState>pulse</outputState>
<pulseDuration>5000</pulseDuration>
</PowerOnState>
</IOOutputPort>
</IOOutputPortList>
```

<IOOutputPortList> is a list. All the alarm output information is listed in this list. In terms of above example, the information of three alarm outputs is listed. The configuration information of each alarm output is contained in an <IOOutputPort>.

<outputState> means that when an event triggers alarm output, alarm output will

output a pulse whose width is depended on <pulseDuration>. The unit of pulse is mm.

If a device doesn't support alarm output, <IOOutputPortList> will be an empty table. The response message is as follows:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <IOOutputPortList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    </IOOutputPortList>
```

2.2.3 Getting the number of audio channels

Command: GET /Audio/channels

Example:

```
    GET /Audio/channels HTTP/1.1
    Host: 172.8.6.228
    Connection: Keep-Alive
    Authorization: Basic YWRtaW46MTIzNDU=
```

IP camera responses as follows:

```
    HTTP/1.1 200 OK
    Connection: close
    Content-Length: 525
    Content-Type: application/xml
    <?xml version="1.0" encoding="UTF-8"?>
    <AudioChannelList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <AudioChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <id>1</id>
    <enabled>false</enabled>
    <audioMode>talkandlisten</audioMode>
    <microphoneEnabled>true</microphoneEnabled>
    <microphoneSource>external</microphoneSource>
    <microphoneVolume>100</microphoneVolume>
    <speakerEnabled>true</speakerEnabled>
    <speakerVolume>100</speakerVolume>
    </AudioChannel>
    </AudioChannelList>
```

<AudioChannelList> is a list. All the audio channels are listed in this list. Currently, both of IP camera and Speed Dome support 1-ch audio channel, so there is

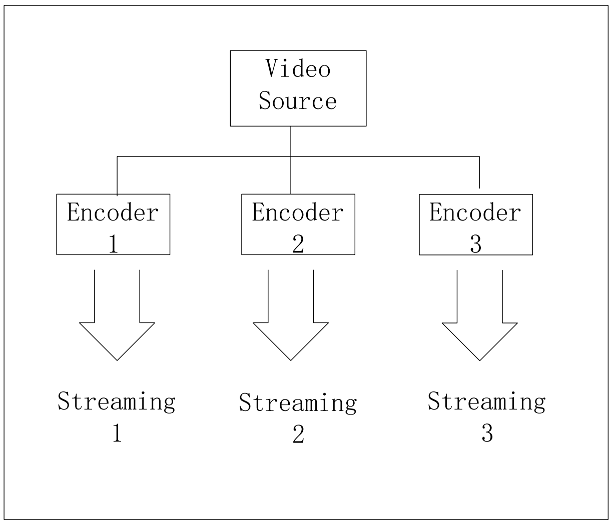
one <AudioChannel> in the <AudioChannelList>.

IP camera, the model is DS-2CD7153-E, doesn’t support audio channel. <AudioChannelList> is an empty table.

```
	<?xml version="1.0" encoding="UTF-8"?>
	<AudioChannelList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
	</AudioChannelList>
```

2.2.4 Getting the number of video channels

Video channel refers to video source. The number of video channels is the same as the number of video source. 1-ch video source supports to encode and output multiple streams. Please refer to the following diagram:



Command: GET /Video/inputs/channels

Example:

```
	GET /Video/inputs/channels HTTP/1.1
	Host: 172.8.6.228
	Authorization: Basic YWRtaW46MTIzNDU=
```

This camera only supports 1 video channel, so there is only 1 channel in the video list of the camera.

```
	HTTP/1.1 200 OK
	Connection: close
	Content-Length: 1184
	Content-Type: application/xml

	<?xml version="1.0" encoding="UTF-8"?>
```

```
        <VideoInputChannelList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <VideoInputChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>1</id>
        <inputPort>1</inputPort>
        <powerLineFrequencyMode>60hz</powerLineFrequencyMode>
        <whiteBalanceMode>outdoor</whiteBalanceMode>
        <whiteBalanceLevel>0</whiteBalanceLevel>
        <exposureMode>auto</exposureMode>
        <Exposure>
        <exposureTarget>0</exposureTarget>
        <exposureAutoMin>0</exposureAutoMin>
        <exposureAutoMax>0</exposureAutoMax>
        </Exposure>
        <gainLevel>100</gainLevel>
        <brightnessLevel>50</brightnessLevel>
        <contrastLevel>50</contrastLevel>
        <sharpnessLevel>50</sharpnessLevel>
        <saturationLevel>50</saturationLevel>
        <gammaCorrectionEnabled>false</gammaCorrectionEnabled>
        <gammaCorrectionLevel>0</gammaCorrectionLevel>
        <WDREnabled>false</WDREnabled>
        <WDRLevel>0</WDRLevel>
        <DayNightFilter>
        <dayNightFilterType>auto</dayNightFilterType>
        </DayNightFilter>
        </VideoInputChannel>
        </VideoInputChannelList>
```

<VideoInputChannelList> is a list. All the video channels are listed in this list.

In terms of above example, this is a response from DS-2CD864FWD-E, and it only supports 1 video channel and the attribute of <powerLineFrequencyMode> is 60Hz, indicating the standard is “NTSC”.

Note:

As for IP camera and Speed Dome, one channel is supported at least.

2.2.5 Checking RS485 Interface

The network camera connects to PTZ device via RS485 serial port. If there is a RS485 serial port on the camera, the device supports PTZ function. You can use the following command check whether the device supports a RS485 serial port.

Command: GET /Serial/ports/1/index

Example:

```
GET /Serial/ports/1/index HTTP/1.1
Host: 172.8.6.228
Connection: Keep-Alive
Authorization: Basic YWRtaW46MTIzNDU=

```

The device responses as follows:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 679
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<ResourceList version="1.0" xmlns="urn:psialliance-org">
<Resource version="1.0">
… …
</ResourceList>
```

If the device returns HTTP OK, the device supports a RS485 serial port, otherwise the device doesn’t.

```
TTP/1.1 400 Bad Request
Connection: close
Content-Length: 258
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<requestURL>/Serial/ports/1/index</requestURL>
<statusCode>4</statusCode>
<statusString>Invalid Operation</statusString>
</ResponseStatus>
```

3 Stream Configuration

A device can support multiple video channels and audio channels. A video channel refers to a video source and a video source can output multiple streams. (*Section 2.2.3and Section 2.2.4* for reference)

3.1 Getting stream parameters of the device

There may be many streams for the device with video source to be applied to different scenario, so the first step is to check the number of the streams of the device.

3.1.1 Getting the number of video streams

You can get all the information of video streams by GET /Streaming/channels.

Example:

```
GET /Streaming/channels HTTP/1.1
Host: 172.8.6.228
Connection: Keep-Alive
Authorization: Basic YWRtaW46MTIzNDU=
```

The device responses as follows:

```
HTTP/1.1 200 OK
Date: Sat, 14 Jul 2012 07:50:24 GMT
Server: App-webs/
Connection: close
Content-Length: 2802
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<StreamingChannelList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<StreamingChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>1</id>
<channelName>Camera 01</channelName>
<enabled>true</enabled>
<Transport>
   ... ...
</Transport>
<Video>
  ... ...
</Video>
<Audio>
  ... ..
</Audio>
</StreamingChannel>
<StreamingChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
```

```
<id>2</id>
<channelName>Camera 01</channelName>
<enabled>true</enabled>
<Transport>
   ... ...
</Transport>
<Video>
 ... ...
</Video>
<Audio>
 ... ...
</Audio>
</StreamingChannel>
</StreamingChannelList>
```

<StreamingChannelList> is a list. All the information of video streams is listed in this list. Each stream is contained in a <StreamngChannel>. In terms of above example, the device supports 2-ch video streams.

3.1.2 Explanation of Stream Parameters

After the number of video streams of a device is clarified, you can get access to a certain stream. But firstly, we need to understand these parameters in HIKCGI protocol.

GET /Streaming/channels/1 or GET /Streaming/channels/101 is used to access to the stream of the first channel. Taking DVR into consideration, we add this kind of format of /Streaming/channels/101, because a DVR supports many channels and each channel can output multiple streams. Therefore, the sequence of ID is: channel number+ stream number of the channel. Please refer to the following examples:

1st stream of channel 1: /Streaming/channels/101;

2nd stream of channel 2: /Streaming/channels/202.

Example:

```
GET /Streaming/channels/101 HTTP/1.1
Host: 172.8.6.228
Connection: Keep-Alive
Authorization: Basic YWRtaW46MTIzNDU=

HTTP/1.1 200 OK
Date: Sat, 14 Jul 2012 08:17:32 GMT
```

```
        Server: App-webs/
        Connection: close
        Content-Length: 1366
        Content-Type: application/xml

        <?xml version="1.0" encoding="UTF-8"?>
        <StreamingChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
        <id>1</id>
        <channelName>Camera 01</channelName>
        <enabled>true</enabled>
        <Transport>
        <rtspPortNo>554</rtspPortNo>
        <maxPacketSize>1000</maxPacketSize>
        <sourcePortNo>8200</sourcePortNo>
        <ControlProtocolList>
        <ControlProtocol>
        <streamingTransport>RTSP</streamingTransport>
        </ControlProtocol>
        </ControlProtocolList>
        <Unicast>
        <enabled>true</enabled>
        </Unicast>
        <Multicast>
        <enabled>true</enabled>
        <destIPAddress>0.0.0.0</destIPAddress>
        <destPortNo>8600</destPortNo>
        </Multicast>
        </Transport>
        <Video>
        <enabled>true</enabled>
        <videoInputChannelID>1</videoInputChannelID>
        <videoCodecType>H.264</videoCodecType>
        <videoScanType>progressive</videoScanType>
        <videoResolutionWidth>1600</videoResolutionWidth>
        <videoResolutionHeight>1200</videoResolutionHeight>
        <videoQualityControlType>CBR</videoQualityControlType>
        <constantBitRate>8192</constantBitRate>
        <fixedQuality>60</fixedQuality>
        <maxFrameRate>3000</maxFrameRate>
        <keyFrameInterval>25</keyFrameInterval>
        <BPFrameInterval>0</BPFrameInterval>
        <snapShotImageType>JPEG</snapShotImageType>
        </Video>
```

```
<Audio>
<enabled>true</enabled>
<audioInputChannelID>11</audioInputChannelID>
<audioCompressionType>G.711ulaw</audioCompressionType>
</Audio>
</StreamingChannel>
```

Taking the importance of the stream parameters into consideration, we explain the stream parameters in this chapter.

Stream parameters mainly include three parts: <Transport>, <Video>and <Audio>. In addition, please refer to *Section 8.9.3* to check the definition of <StreamingChannel>.

1. <Transport>

The attribute of <Transport> tag is "req" in the <StreamingChannel>, indicating that <Transport> is necessary by GET request and PUT request.

< rtspPortNo> indicates the port number of RTSP. This tag is configurable and the listening port will be changed after the configuration of <rtspPortNo> takes effect.

<Multicast> is configuration information of multicast streaming, you can configure the multicast address by <destIPAddress>and the multicast port by <destPortNo>.

Note:

In general, <Transport > also acts as optional part so as to have a convenient integration for the third party companies.

2. <Video>

<videoCodecType>: encoding type of the video. HIKCGI protocol supports H.264, MPEG4 and MJPEG.

<videoScanType>: image scan type. HIKCGI protocol supports Progressive Scan and Interlaced Scan.

<videoResolutionWidth>: width of video. The unit is pixel.

<videoResolutionHeight>: height of video. The unit is pixel.

<videoQualityControlType>: bit rate type. HIKCGI supports CBR (Constant Bitrate) and VBR (Variable Bitrate). If bit rate type is CBR, <constantBitRate>

indicates the value of the CBR and the unit is kbps. If bit rate type is VBR, <constantBitRate> indicates the maximum bit rate.

<fixedQuality>: image quality. When the value is 100, the image quality is best; otherwise when the value is 1, the image quality is worst.

<maxFrameRate>: frame rate of current stream. The value is an integer. To avoid that the frame rate is less than 1fps, the actual value of <maxFrameRate> is: current frame rate*100.

For example, if the frame rate is 1/4fps, the value of <maxFrameRate> will be 25fps; if the frame rate is 12fps, the value of <maxFrameRate> will be 1200fps.

<keyFrameInterval>: the interval of I frame. There will appear key frame every a few of frames.

3. <Audio>

<Audio> is the audio part of the stream. If the device doesn't support audio channel, <Audio> doesn't exist. (*Section 2.2.3* for reference)

If the device supports audio channel, the stream can contain audio data. You can set <enabled> as "true" to open audio stream, or set <enabled> as false to close audio stream.

<audioCompressionType>: audio encoding format. HIKCGI supports G.711ulaw, G.711alaw, G.726, AAC and G722.

3.2 Putting the parameters of streaming

As *Section 3.1.2* mentioned, stream parameters include three parts :< Transport>, <Video> and <Audio>. Before sending a request, you must know which optional values each parameter supports. For example, if you set the resolution of <Video>, you need to know all the optional resolutions, then send a request to update or reset the resolution, otherwise the device will return error information if you send a wrong request.

3.2.1 Capabilities

"/Streaming/channels/1/capabilities" is used to access to optional values of each parameter. Please note that "Capabilities" only supports GET method. In a word, you can get the capabilities of a parameter but not update and reset the parameter.

The following example is to get capabilities of the first stream:

```
GET /Streaming/channels/1/capabilities HTTP/1.1
Host: 172.8.6.176
Connection: Keep-Alive
Authorization: Basic YWRtaW46MTIzNDU=
```

IP camera responses as follows:

```
HTTP/1.1 200 OK
Date: Sun, 15 Jul 2012 16:06:19 GMT
Server: App-webs/
Connection: close
Content-Length: 1944
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<StreamingChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id opt="1,2">1</id>
<channelName min="0" max="32">Camera 01</channelName>
<enabled opt="true">true</enabled>
<Transport>
<rtspPortNo min="0" max="65535" def="554">554</rtspPortNo>
<maxPacketSize opt="1000">1000</maxPacketSize>
<sourcePortNo min="0" max="65535" def="8200">8200</sourcePortNo>
<ControlProtocolList>
<ControlProtocol>
<streamingTransport opt="RTSP">RTSP</streamingTransport>
</ControlProtocol>
</ControlProtocolList>
<Unicast>
<enabled opt="true" def="true">true</enabled>
</Unicast>
<Multicast>
<enabled opt="true" def="true">true</enabled>
<destIPAddress min="8" max="16">224.1.2.3</destIPAddress>
<destPortNo min="0" max="65535" def="8600">8600</destPortNo>
```

```
    </Multicast>
    </Transport>
    <Video>
    <enabled opt="true">true</enabled>
    <videoInputChannelID opt="1">1</videoInputChannelID>
    <videoCodecType opt="H.264,MPEG4">H.264</videoCodecType>
    <videoScanType opt="progressive">progressive</videoScanType>
    <videoResolutionWidth
opt="640*480,1280*720,1280*960">1280</videoResolutionWidth>
    <videoResolutionHeight
opt="640*480,1280*720,1280*960">960</videoResolutionHeight>
    <videoQualityControlType opt="CBR,VBR">CBR</videoQualityControlType>
    <constantBitRate min="32" max="16384">4096</constantBitRate>
    <fixedQuality opt="1,20,40,60,80,100">60</fixedQuality>
    <maxFrameRate
opt="2500,2200,2000,1800,1600,1500,1200,1000,800,600,400,200,100,50,25,12,6">
600</maxFrameRate>
    <keyFrameInterval min="1" max="400">12</keyFrameInterval>
    <BPFrameInterval opt="0,1,2">0</BPFrameInterval>
    <snapShotImageType opt="JPEG" def="JPEG">JPEG</snapShotImageType>
    </Video>
    <Audio>
    <enabled opt="true,false">true</enabled>
    <audioInputChannelID opt="11">11</audioInputChannelID>
    <audioCompressionType opt="G.711ulaw"
def="G.711ulaw">G.711ulaw</audioCompressionType>
    </Audio>
    </StreamingChannel>
```

Capabilities information is shown in corresponding tag in form of attribute.

If the optional values of a certain parameter is in a continuous interval, we use “min~max” to indicate these values, “min” = minimum value and “max” = maximum value. For example, <constantBitRate min="32" max="16384"> indicates that the minimum baud rate is 32kbps and the maximum baud rate is 16384kbps. Random integral baud rate is valid when the baud rate is between 32kbps and 16384kbps.

If the option value of a certain parameter is discrete, you can list all the optional values by “opt”, such as <fixedQuality opt="1, 20, 40, 60, 80,100">, it indicates that image quality supports 6 levels. The value corresponding to level 1 is 1; the value corresponding to level 2 is 20 and so on. <videoCodecType opt="H.264, MPEG4"> indicates that main stream supports H.264 encoding and MPEG4 encoding.

Both of <videoResolutionWidth> and <videoResolutionHeight> list the optional values of video resolution, which are shown in the form of “width * height”, such as <videoResolutionHeight opt="640*480, 1280*720, 1280*960">, it indicates that main stream supports 3 optional resolutions: 640(W)*480(H), 1280(W)*720(H) and 1280(W)*960(H).

3.2.2 Setting Stream Parameters

After getting the capabilities by GET/Streaming/channels/ID/capabilities, you can update or reset some parameters by PUT /Streaming/channels/ID.

Please pay attention to those definitions in *HIKCGI Protocol Guide*. If the attribute of a tag is “req”, the tag must be contained in HTTP body when you send a PUT request, otherwise the request is illegal. For example, <videoResolutionWidth>, <videoResolutionHeight>,<videoQualityControlType> and <maxFrameRate> are defined as ”req” in the <Video> page, so the four parameters must be contained in the <Video> page when you send a PUT request.

Here is an example. The device is the same one as *Section 3.2.1*, which supports audio channel. We send a PUT request to reset resolution as 640*480, decrease bit rate to 2048kbps and raise frame rate to 25fps. Please refer to the following request:

```
    PUT /Streaming/channels/1 HTTP/1.1
    Authorization: Basic YWRtaW46MTIzNDU=
    Content-Type:text/xml
    Host:172.8.6.176
    Content-Length:535

    <?xml version="1.0" encoding="UTF-8"?>
    <StreamingChannel version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <id>1</id>
    <enabled>true</enabled>
    <Transport>
```

```
<rtspPortNo>554</rtspPortNo>
</Transport>
<Video>
<videoInputChannelID>1</videoInputChannelID>
<videoResolutionWidth>640</videoResolutionWidth>
<videoResolutionHeight>480</videoResolutionHeight>
<videoQualityControlType>CBR</videoQualityControlType>
<constantBitRate>2048</constantBitRate>
<maxFrameRate>2500</maxFrameRate>
</Video>
</StreamingChannel>
```

IP camera responses as follows:

```
HTTP/1.1 200 OK
Date: Sun, 15 Jul 2012 17:09:05 GMT
Server: App-webs/
Connection: close
Content-Length: 243
Content-Type: application/xml
X-Appweb-Seq: 34

<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<requestURL>/streaming/channels/1</requestURL>
<statusCode>1</statusCode>
<statusString>OK</statusString>
</ResponseStatus>
```

The first line of the HTTP header returns 200 and OK, and it indicates that the request is executed successfully.

Please note that only after the device reboots can some parameters take effect. For example, the current encoding type of the device is H.264, if you reset the encoding type as MPEG4, the response from the device will prompt “Reboot Required” in the <statusString> tag. “7” indicates that this request needs to reboot the device.

```
HTTP/1.1 200 OK
Date: Sun, 15 Jul 2012 17:09:05 GMT
```

```
    Server: App-webs/
    Connection: close
    Content-Length: 262
    Content-Type: application/xml
    X-Appweb-Seq: 34

    <?xml version="1.0" encoding="UTF-8"?>
    <ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <requestURL>/streaming/channels/1</requestURL>
    <statusCode>7</statusCode>
    <statusString>Reboot Required</statusString>
    </ResponseStatus>
```

4 System Maintenance

System maintenance includes: Device Information Configuration, Reboot, Restore Factory Configuration, Upgrade and System Time Configuration.

4.1 Device Information Configuration

Device information configuration has been introduced in *Section 2.1*. Most of device information is read-only, such as Model of device, Serial Number and Description Information of the device. Currently, only Device Name and Device ID support to be updated.

Note:

"Device ID" is useful when a device supports remote controller.

Example: Setting Device Name as "NewName"

```
    PUT /System/deviceInfo HTTP/1.1
    Authorization: Basic YWRtaW46MTIzNDU=
    Content-Type:text/xml
    Content-Length:164

    <?xml version="1.0" encoding="UTF-8"?>
    <DeviceInfo version="1.0"
```

```
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <deviceName>NewName</deviceName>
    </DeviceInfo>
```

4.2 Reboot

PUT /System/reboot doesn't need any parameters. Once the device retrieves this request and the request is executed successfully, the device will reboot at once. The rebooting time is related with the device.

Example: Reboot device remotely

```
    PUT /System/reboot HTTP/1.1
    Host:172.8.6.176
    Authorization: Basic YWRtaW46MTIzNDU=
```

The device responses as follows:

```
  HTTP/1.1 200 OK
  Date: Sun, 15 Jul 2012 18:48:55 GMT
  Server: App-webs/
  Connection: close
  Content-Length: 236
  Content-Type: application/xml
  X-Appweb-Seq: 1244


  <?xml version="1.0" encoding="UTF-8"?>
  <ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
  <requestURL>/system/reboot</requestURL>
  <statusCode>1</statusCode>
  <statusString>OK</statusString>
  </ResponseStatus>
```

4.3 Reset Factory Configuration

This function supports two modes: one is to reset factory configuration fully; the other one is reset some basic factory configuration. The first mode will reset all the information of the device to factory configuration; the second mode will reserve some configuration information that the client has finished, such as Video Standard, Network Parameters, focus value of lens and Language and so on, other information will be reset to factory configuration.

Example: reset factory configuration fully

```
PUT /System/factoryDefault?mode=full HTTP/1.1
    Host:172.8.6.176
    Authorization: Basic YWRtaW46MTIzNDU=
```

Example: reset some basic factory configuration

```
    PUT /System/factoryDefault?mode=basic HTTP/1.1
    Host:172.8.6.176
    Authorization: Basic YWRtaW46MTIzNDU=
```

Only after the device reboots will reset factory configuration take effect. The device responses as follows:

```
    HTTP/1.1 200 OK
    Connection: close
    Content-Length: 257
    Content-Type: application/xml

    <?xml version="1.0" encoding="UTF-8"?>
    <ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <requestURL>/system/factorydefault</requestURL>
    <statusCode>7</statusCode>
    <statusString>Reboot Required</statusString>
    </ResponseStatus>
```

4.4 Upgrade

Upgrading operation is used to upgrade the firmware of the device to a newer one. Please note that you need to know the length of upgrading package before sending upgrading command, therefore, other operations are forbidden during the process.

"HTTP body" is an upgrade package when using PUT/System/firmwareUpgrade. The device will check the upgrade package after receiving it and prompt "Reboot

Required”. At this time, you need to send PUT /System/reboot to reboot the device (Section *4.2* for reference).

Example: upgrade IP camera. The length of upgrade package is 13481620 bytes.

```
PUT /System/firmwareUpgrade /PSIA/System/updateFirmware HTTP/1.1
HOST: 172.8.6.176
Authorization: Basic YWRtaW46MTIzNDU=
Content-Length: 13481620
Connection: keep-alive
Content-Type: application/octet-stream

.....y..
...............F\T@4JAEC.)5",EF\T@4...................................................m...7.....`........
............................H...`.....................................2....*.....................................#...............
...............................e................................ ........I................................|<......................
.....................Y.......9.................................0......................................................................
.................p..._...........................................Y..g....................................(.....gK..............
....................jB.=....5
.................................+{......u..................................?z..i...d?..................................B
l.....qM.................................3....t..&..................................I.v.....gy...............................
..........+...'..V...sO.5...7T........Q.......Linux-2.6.18_pro500-davinci_evm-.........................
............(o.....T7...p....... ..............V4... ... ....!.............~0....P.
```

Here is only a part of upgrade information. That is the content of upgrade package, followed by HTTP Header. The format of upgrade package is binary, so “Content-type” of the header fields is assigned as “application/octet-stream” or “application/binary”.

4.5 System Time Configuration

HIKCGI protocol supports two timing modes: NTP Timing and Manual Timing.

4.5.1 NTP Timing

If the device is set as NTP Timing, firstly, the device sends a request to NTP Server; secondly, NTP Server will return UTC time to the device; finally, the device will execute timing according to UTC time.

Before setting NTP Timing, you need to configure a NTP Server and set the timing mode of system as NTP Timing Mode.

1. Setting NTP Server

If the device supports NTP Timing, the response of the device should contain at least one NTP server configuration by GET /System/time/ntpServers.

```
GET /System/time/ntpServers HTTP/1.1
Host: 172.8.6.176
Authorization: Basic YWRtaW46MTIzNDU=
```

The device responses as follows:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 340
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<NTPServerList version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<NTPServer version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>1</id>
<addressingFormatType>hostname</addressingFormatType>
<hostName></hostName>
<portNo>123</portNo>
</NTPServer>
</NTPServerList>
```

The return information is a list of NTP Server configuration and the current device supports one NTP Server. We don’t configure NTP server, so <hostName> tag is empty.

The address of NTP Server supports two modes: Domain or IP address.

If you use Domain to represent NTP Server, the value of <addressingFormatType> should be assigned as “hostname” and <hostName> is necessary. It is used to assign Domain.

If you use IP address to represent NTP Server, <addressingFormatType> should be assigned as “ipaddress” and <ipAddess> is necessary. It is used to assign IP address.

Example: IP Address Mode

```
PUT /System/time/ntpServers/1 HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type: text/xml
Host: 172.8.6.176
Content-Length: 268

<?xml version="1.0" encoding="utf-8"?>
<NTPServer>
<id>1</id>
<addressingFormatType>ipaddress</addressingFormatType>
<ipAddress>10.6.8.238</ipAddress>
<portNo>123</portNo>
<Extensions>
<synchronizeInterval>60</synchronizeInterval>
</Extensions>
</NTPServer>
```

The IP address of NTP Server is 10.6.8.238 and the server port is 123. The default interval time is 24 hours. Some new devices support to set the interval of timing. This function is contained in the <synchronizeInterval> tag. The unit is minute. As above example, the interval of timing is 60 minutes.

4.5.2 Setting System Timing Mode

After setting NTP Timing Server, you can assign system to NTP Timing. You can modify the current timing mode by PUT /System/time.

Example: Set current Timing mode as NTP Timing.

```
PUT /System/time HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type: text/xml; charset=utf-8
Host: 172.8.6.176
Content-Length:117

<?xml version="1.0" encoding="utf-8"?>
<Time>
```

```
    <timeMode>NTP</timeMode>
    <timeZone>CST-8:00:00</timeZone>
    </Time>
```

The above command sets Timing Mode as NTP Timing and current time zone as CST-8:00:00. If the timing takes effect immediately after send a PUT request (it depends on interval time), the time which Timing Server returns will be 12:00, and then the device will add extra 8 hours according to current time zone, so the time will be 20:00 on July 15th, 2012.

4.5.2 Manual Timing

You need to set Timing Mode as Manual Timing and assign local time.

Example: Setting Manual Timing.

```
PUT /System/time HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type: text/xml; charset=utf-8
Host: 172.8.6.176
Content-Length: 164

<?xml version="1.0" encoding="utf-8"?>
<Time>
<timeMode>manual</timeMode>
<localTime>2012-07-15T19:40:04</localTime>
<timeZone>CST-8:00:00</timeZone>
</Time>
```

After setting Manual Timing, system time will refer to <localTime> and take effect immediately.

5 IO Control

IO Control refers to setting alarm input and alarm output. Not all the devices support alarm input and Alarm output, so the first step is to get capabilities. (*Section 2.2.1 and Section 2.2.2* for reference)

5.1 Alarm Input

You can set an effective signal type of alarm input. Currently, HIKCGI supports two types: high level triggering alarm input and low level triggering alarm input. If you set high level triggering alarm input, it means that the alarm input port will be NC when there is no alarm signal. Once the alarm input port turns into NO, it will appear alarm signal.

Note:

NO: normal open; NC: normal closed.

Example: Setting alarm input 1, setting high level triggering alarm input.

```
PUT /IO/inputs/1 HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Host:172.8.6.176
Content-Type:text/xml
Content-Length:175

<?xml version="1.0" encoding="UTF-8"?>
<IOInputPort version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
<id>1</id>
<triggering>high</triggering>
</IOInputPort>
```

5.2 Alarm Output

When a certain condition occurs, the device will determine whether to trigger alarm output. If the device needs to trigger alarm output, the alarm output port will send a pulse whose width can be assigned.

Example: Setting alarm output 1, setting the width of pulse as 0.5s.

```
PUT /IO/outputs/1 HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type:text/xml
Content-Length:284

<?xml version="1.0" encoding="UTF-8"?>

<IOOutputPort version="1.0"
```

```
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <id>1</id>
    <PowerOnState>
    <defaultState>low</defaultState>
    <outputState>pulse</outputState>
    <pulseDuration>5000</pulseDuration>
    </PowerOnState>
    </IOOutputPort>
```

The width of pulse is assigned as 5000ms.

5.3 Manual Trigger Alarm Output

In general, alarm output is regarded as a linkage method. When an event occurs, there will be a logical judging within the device and trigger alarm output.

In addition, alarm output can be triggered manually, too. In this case, the first step is to set alarm output mode as manual trigger, and then send a corresponding request to enable alarm output or disable alarm output.

5.3.1 Setting Alarm Output Mode as manual trigger mode

Use PUT /IO/outputs/ID to set the alarm output mode as manual trigger mode. The example of *Section 5.2* is pulse mode. In this mode, when an event occurs, if the linkage method is set as alarm output, alarm output port will send a pulse;

Setting <outputState> as “high” by PUT /IO/outputs/ID, thus setting alarm output mode as manual trigger mode.

Example: setting alarm output mode as manual trigger mode.

```
    PUT /IO/outputs/1
    Authorization: Basic YWRtaW46MTIzNDU=
    Content-Type:text/xml
    Content-Length:246

    <?xml version="1.0" encoding="UTF-8"?>
    <IOOutputPort version="1.0"
xmlns="http://www.std-cgi.com/ver10/XMLSchema">
```

```
<id>1</id>
<PowerOnState>
<defaultState>low</defaultState>
<outputState>high</outputState>
</PowerOnState>
</IOOutputPort>
```

5.3.2 Trigger Alarm Output Manually.

PUT /IO/outputs/1/trigger can trigger alarm output to send a high level manually.

Example:

```
PUT /IO/outputs/1/trigger
Authorization: Basic YWRtaW46MTIzNDU=
Content-Type:text/xml
Content-Length:246

<IOPortData xmlns="http://www.hikvision.com/ver10/XMLSchema">
<outputState>high</outputState>
</IOPortData>
```

Note:

In this example, a high level is given at the alarm output port. If you want to stop the alarm output, you can set the <outputState> as “low”.

5.4 Alarm Input Event

HIKCGI protocol supports event function. You can assign the linkage method of alarm input, such as e-mail linkage. When the alarm happens, the system will send e-mail to a specific IP address.

Please refer to *“HIKCGI Event Function”*.

6 Network Configuration

A device can support multiple network interfaces. In terms of IP camera, an IP camera can support wire-line network and wireless network. In this case, the first network interface is wire-line network; the second network interface is wireless network. But as for a device without WIFI, there is only one network interface: /Network/interfaces/1.

6.1 Network Configuration

HIKCGI Protocol offers an interface to network configuration, configurable parameters includes IP Address, Subnet mask, Gateway and DNS Server and so on.

Example: Configure wire-line network.

```
    PUT /Network/interfaces/1/ipAddress HTTP/1.1
    Authorization: Basic YWRtaW46MTIzNDU=
    Host:172.8.6.176
    Content-Type:text/xml
    Content-Length:406

   <?xml version="1.0" encoding="UTF-8"?>
    <IPAddress version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <ipVersion>v4</ipVersion>
    <addressingType>static</addressingType>
    <ipAddress>172.8.6.176</ipAddress>
    <subnetMask>255.255.255.0</subnetMask>
    <DefaultGateway>
    <ipAddress>172.8.6.1</ipAddress>
    </DefaultGateway>
    <PrimaryDNS>
    <ipAddress>0.0.0.0</ipAddress>
    </PrimaryDNS>
    </IPAddress>
```

In the above example, ID of the network interface is 1, which indicates that this is a wire-line network. If this is a wireless network, ID of the network interface should be 2.

Only after you assign <addressingType> as “static” will IP address, Gateway and DNS Server take effect, otherwise if you assign <addressingType>as “dynamic”, you need to enable DHCP function and get parameter via DHCP server.

<ipVersion>: the type of IP address. The type of the above example is IPv4;

<subnetMask>: Subnet mask. The subnet mask of the above example is 24 bits;

<DefaultGateway>: default gateway. The default gateway of the above example is 172.8.6.1.

<PrimaryDNS>: DNS Server address. We don't assign DNS Server address, so and the default value is 0.0.0.0.

6.2 Enable DHCP

After enabling DHCP function, the network configuration of the device will be retrieved from DHCP server, such as IP address, Subnet Mask and Gateway of the device. The above-mentioned parameters don't need to be configured extra, but the address of DNS server can be still configured. Actually, to enable DHCP function means to set address type as dynamic address type.

Example:

```
    PUT /Network/interfaces/1/ipAddress HTTP/1.1
    Authorization: Basic YWRtaW46MTIzNDU=
    Host:172.8.6.176
    Content-Type:text/xml
    Content-Length:197

   <?xml version="1.0" encoding="UTF-8"?>
    <IPAddress version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
    <ipVersion>v4</ipVersion>
    <addressingType>dynamic</addressingType>
    </IPAddress>
```

If the previous type of IP address is “static”, now set <addressingType> as “dynamic”, the configuration will take effect after the device reboots.

```
    HTTP/1.1 200 OK
    Connection: close
    Content-Length: 257
    Content-Type: application/xml

    <?xml version="1.0" encoding="UTF-8"?>
    <ResponseStatus version="1.0"
xmlns="http://www.hikvision.com/ver10/XMLSchema">
```

```
<requestURL>/system/factorydefault</requestURL>
<statusCode>7</statusCode>
<statusString>Reboot Required</statusString>
</ResponseStatus>
```

6.3 DDNS Function

In general, DDNS function is applied for a device without a fixed IP address. Every time, the device may get a different IP address after rebooting. For example, if you enable PPPoE function, the device will dial up to network, but the IP address allocated to the device is different every time. In this case, we can apply for a Domain to the device. When the device reboots, the latest network configuration information of the device will be sent to DDNS server. The reasonability of DDNS server is to relate IP address with corresponding Domain and record the information at any time. In process of interaction, you can use Domain to access to the device.

You need to assign a DDNS server provider, IP address of DDNS server, port number and authentication before enabling DDNS function.

Example: setting DDNS server.

```
PUT /Network/interfaces/1/ddns HTTP/1.1
Authorization: Basic YWRtaW46MTIzNDU=
Host:172.8.6.176
Content-Type:text/xml
Content-Length:358

<?xml version="1.0" encoding="UTF-8"?>
<DDNS version="1.0" xmlns="http://www.hikvision.com/ver10/XMLSchema">
<enabled>true</enabled>
<provider>DynDNS</provider>
<serverIPAddress>10.16.2.10</serverIPAddress>
<portNo>8200</portNo>
<domainName>www.hikvision.ipctest</domainName>
<userName>hikvisionipc</userName>
<password>hik1601480</password>
</DDNS>
```

The DDNS service is enabled in the example, <enabled> is set as “true”.

We set the DDNS service provider as DynDNS; IP address of the server is 10.16.2.10 and the port is 8200.

Domain name of the device is "www.hikvision.ipctest" which the dynamic address is bound with.

User name and password is needed in the PUT request and the password is read-only, which is shown only in "PUT" but "GET". The authentication information can be protected in this way.

7 Two-Way Audio

If a device supports audio channel, you can realize two-way audio by HIKCGI protocol interface. (*Section 2.2.3* for reference)

Open corresponding audio channel before starting two-way audio. At this time, the device will start some initialization. You need to close corresponding audio channel after finishing two-way audio. Because two-way audio is bidirectional, not only you can receive audio data from a device, but also send audio data to the device.

The steps of two-way audio configuration are as follows:

(1) Open the channel of two-way audio

Open two-way audio channel by PUT /TwowayAudio/open, the device will start some initialization.

(2) Receive / send audio data

(2.1) Receive audio data from device by GET /TwowayAudio/receiveData.

After receiving GET request, the device will create a task and send encoding audio data to client end. Please note that the task can be created only one time, which indicates that GET/TwowayAudio/receiveData can be called only one time, too. The device will return HTTP OK and there will be only audio data after HTTP Header.

When the connection is disconnected or the client end sends PUT /TwowayAudio/close, the task is over.

(2.2) Send audio data to the device by PUT /TwowayAudio/sendData.

After receiving PUT request, the device will create a task to receive audio data from the client end. The task can be created only one time, which means PUT /TwowayAudio/sendData can be called only one time, too.

After PUT /TwowayAudio/sendData, if everything goes well, the device will return HTTP OK to inform the client end that the task is created successfully. Now the client end can send audio data to the device via TCP directly, there is no need to use

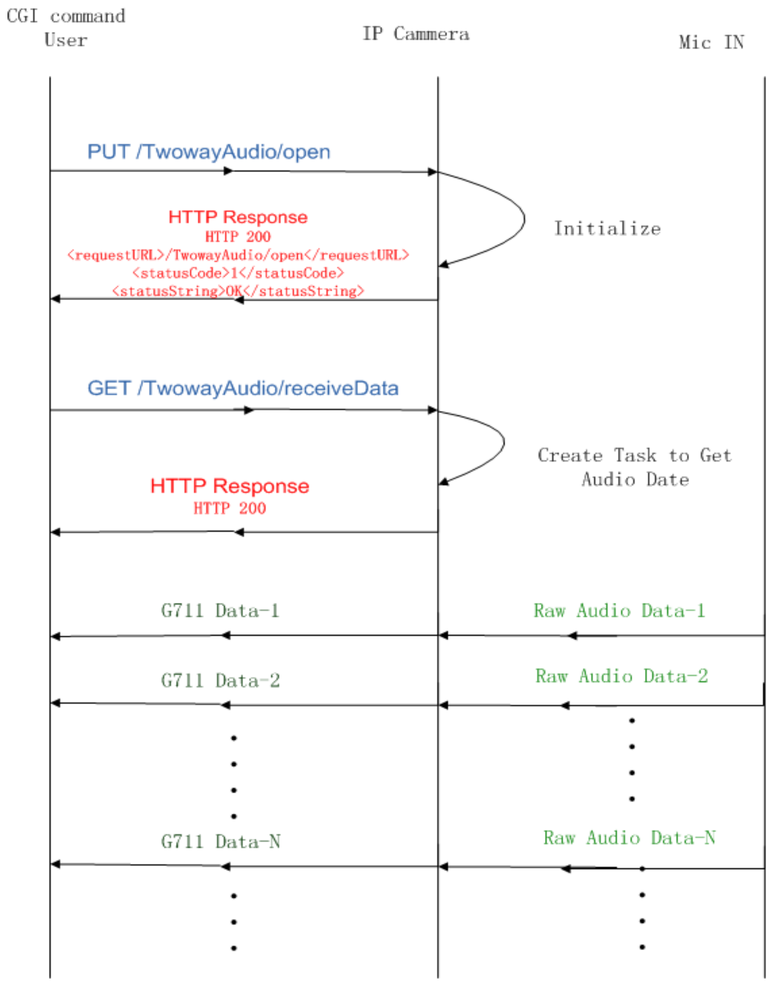
PUT method.

(3) Close two-way audio channel

After closing two-way audio successfully, the device will send HTTP OK message to the client end. After this command, the audio sending task and the audio receiving task will exit synchronously.

Please refer to the following diagram of two-way audio:

(1) The process of the client end receiving audio data



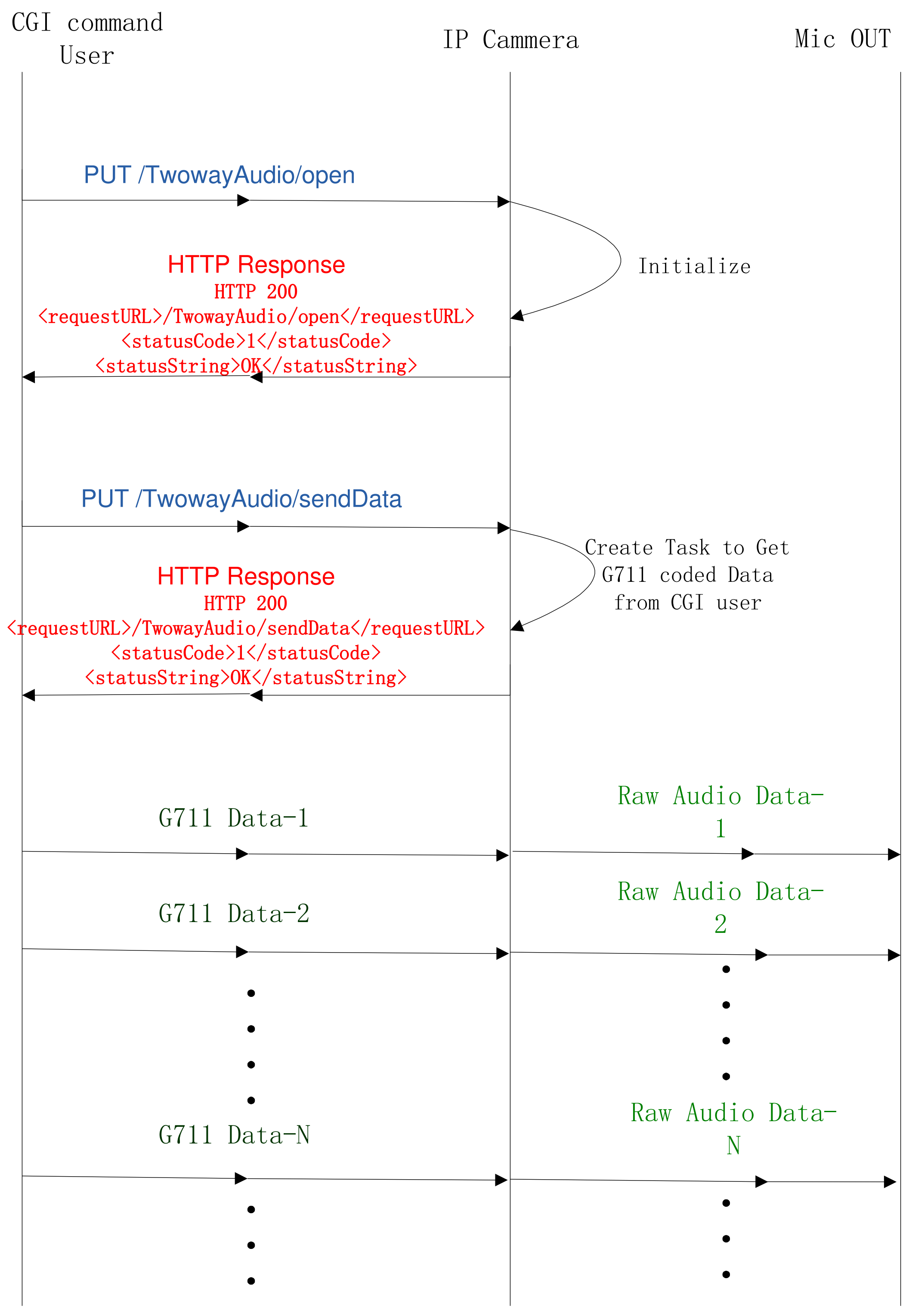
In the above process, the client end uses PUT/TwowayAudio/open to open two-way audio channel, and then sends GET request to the device. The device receives GET request to create a task and sends encoding audio data, and then the device sends HTTP OK message to inform the client end, after that, the task starts.

The device adopts G711 format to encode the audio data from Mic In and sends these data to the client end constantly. The device only response a HTTP OK message to the client end when the task is created. There will be only audio data after HTTP Header.

The client end calls GET /TwowayAudio/receiveData only one time, and then the device sends the audio data constantly until it the task will be over when the connection is disconnected or receiving PUT /TwowayAudio/close.

If you call GET /TwowayAudio/receiveData" request for two times, the device will return failure.

(2) The process of the client end sending audio data


CGI command User
IP Cammera
Mic OUT
PUT /TwowayAudio/open
Initialize
HTTP Response
HTTP 200
<requestURL>/TwowayAudio/open</requestURL>
<statusCode>1</statusCode>
<statusString>OK</statusString>
PUT /TwowayAudio/sendData
Create Task to Get G711 coded Data from CGI user
HTTP Response
HTTP 200
<requestURL>/TwowayAudio/sendData</requestURL>
<statusCode>1</statusCode>
<statusString>OK</statusString>
G711 Data-1
Raw Audio Data-1
G711 Data-2
Raw Audio Data-2
G711 Data-N
Raw Audio Data-N

In the above process of two- way audio, at first, the client end uses PUT /TwowayAudio/open to open two-way audio channel, and then sends PUT/TwowayAudio/sendData to the device. The device receives PUT request to create a task and receives audio data. The device will response HTTP OK to the client end, after that, the task starts. The task waits for the audio data from the client end, and then the device will decode these audio data and send them to Mic Out.

The client end only sends PUT /TwowayAudio/sendData for one time, the following encoding data is sent by TCP connection. PUT /TwowayAudio/sendData is used to create a task. Please note that the task can be created for one time, or return error.

8 Appendix

8.1 Content-Type of HTTP Header：

application/xml : text content of XML format
text/plain: common text type, such as display content of local time
application/binary: binary file format, such as content of upgrade packet
image/jpeg: JPEG picture, such as content of capture command
multipart/mixed: The format indicates that HTTP Body is longer and adopts segment format. The format of the content of each segment may be different. The general statement is as follows:
multipart/mixed; boundary=hikboundary.
In this above example, boundary indicates that the separator of each segment is “hikboundary”. For example, GET /Event/notification/alertStream adopts segment format.
audio/basic: audio data, such as two-way audio data
application/soap+xml： SOAP format

8.2 ResponseStatus Page

<ResponseStatus> is used to indicate the result of PUT request or POST request. HIKCGI protocol defines 7 status codes to indicate the result of the operation. The detailed information is as follows:

```
    <?xml version="1.0" encoding="utf-8" ?> - <ResponseStatus version="1.0"
xmlns=" http://www.hikvision.com/ver10/XMLSchema">
    <requestURL>/Streaming/Channels</requestURL>
    <statusCode>1</statusCode>
    <!-- O=1-OK, 2-Device Busy, 3-Device Error, 4-Invalid Operation, 5-Invalid
XML Format, 6-Invalid XML Content; 7-Reboot Required-->
    <statusString>OK</statusString>
    <ID>1</ID>
    </ResponseStatus>
```