SMaBiT

B-VIEW 2 REST-API SPECIFICATION

SIGLA:	SPT907000SERIES2	**Ente emittente:** BV
TIPO DI DOCUMENTO:	Technical Specification	**Circolazione:**
OGGETTO:	**B-VIEW 2** **REST API SPECIFICATION**	
CODICE:	B-VIEW 2 SERIES	

Riferimento ad altri documenti:

Allegati:

Release Date	Version	Modification	By
14/07/2016	1.0	Initial public version	PH
15/07/2016	1.1	User configuration updated	PH
27/07/2016	1.2	Record/Notify/URL added	PH
25/08/2016	1.3	Updated error codes	PH
04/11/2016	1.4	Commands for WIFi testing	PH
11/11/2016	1.5	Updated stream and PTZ functions	PH
25/04/2019	1.6	Added motion alarm settings	RG
15/01/2020	1.7	Updated the section FirmwareUpgrade	RG
01/09/2021	1.8	Updated for Scanvest Version	RG

INDEX

1 Scope

This specification defines a HTTP-based application programming interface that enables physical security and video management systems to communicate with IP media devices in a particular way.

For further reference about media streaming, please refer to the Develop API of the RTSP protocol.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] RFC2616 Hypertext Transfer Protocol-HTTP/1.1

[2] W3C XML 1.0 specification

[3] W3C Character encodings

[4] RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax and Semantics

[5] RFC 2617 HTTP Authentication:Basic and Digest Access Authentication

[6] International Electrotechnical Commission —ISO/IEC standard on UPnP device architecture makes networking simple and easy‖, 2008-12-09. Retrieved on 2009-05-07.

[7] International Organization for Standardization —ISO/IEC standard on UPnP device architecture makes networking simple and easy‖, 2008-12-10. Retrieved on 2009-05-07.

[8] UPnP Forum —UPnP Specifications Named International Standard for Device Interoperability for IP-based Network Devices‖, 2009-02-05. Retrieved on 2009-05-07.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Services: a set of resources consisting of relevant General Resources.

General Resources: physical resources that supported by the devices.

Node: Services and General Resources.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

REST	REpresentational State Transfer
UPnP	Universal Plug and Play

4 Architecture and Transmission Mechanism

The IP Media Device Management Protocol is based on REST architecture. The management and control interfaces defined in this specification are treated as resources utilizing the REpresentational State Transfer (REST) architecture. This architecture facilitates users by grouping related resources within hierarchical namespaces, and is more flexible for service discovery and future expansion.

REST architecture consists of clients and servers, among which clients initiate request to servers, while servers handle requests and response accordingly. Requests and responses are established via the transmission of ―representations‖ of ―resources‖. REST architecture need to be based on an Application Layer protocol which provides various of standard communication formats for applications based on the transfer of meaningful representational state. HTTP[1] has a very rich vocabulary in terms of verbs(or ―methods‖),URIs, request and response headers, Internet media types, HTTP request and response codes etc. In addition, HTTP also has some features particularly suitable for REST architecture. So HTTP is used as external Application Layer protocol in this specification. In the architecture, clients are physical security and video management systems; servers are IP media devices.

This specification also contains full XML schema for the introduced resources.

4.1 REST and HTTP Methods

The following table shows how HTTP verbs are typically used to implement a web service based on REST architecture.

Resourc	GET	PUT	POST	DELETE
Collection URI, such as http://webServer/resour ces	**List** the members of collection,complete with their member URIs for further navigation.	Meaning defined as ―**replace** the entire collection with another collection‖.	**Create** a new entry in the collection where the ID is assigned automatically by the collection. The ID created is usually included as part of the data returned by this	Meaning defined as ―**delete** the entire collection‖.

Member URI, such as http://webServer/resources/7416	**Retrieve** a representation ofthe addressed member of the collection expressed in an appropriate MIME type.	**Update** the addressed member of the collection or **create** it with the specified ID.	Treat the addressed member as a collection in its own right and **create** a new subordinate of it.	**Delete** the addressed member of the collection.

4.2 XML

A device must support the syntax defined by W3C XML 1.0 specification [2] and UTF-8 character set [3]. All XML files must adopt UTF-8 encoding according to RFC3629. Additionally, all resources share a common XML schema as defined in Annex.

Any resources can specify separate input and output XML Documents. If a specific data structure is defined inside these documents, then they must be specified as XML Schema Documents (xsd) in Annex.

Lists contained in XML blocks will be represented in the format of <XXXList>, and each <XXXList> tag may contain one or more nodes.

4.3 Resources overview

Services	Description	Reference
Network	Configure network interfaces.	8.1
System	Configure and operate the general system functions.	8.2
Streams	Handle video-related configuration.	8.3
Images	Configure the general Images configuration	8.4
Pictures	Configure the general pictures configuration	8.5

4.4 Protocol URL

The URL scheme is used to locate device resources via a specific protocol in the network. This section defines the syntax and semantics for http(s) URLs.

```
<protocol>://<host>[:port][abs_path [?query]]
```

protocol: URL scheme for the particular request. The http and https protocols are allowed in this specification.

host: The host field refer to the hostname, IP address, or the FQDN of an IP device.

port: The port field refer to the port number of that host on which the identified resource is located at the IP device listening for TCP connections. If the port is empty or not given, the default port is assumed. For HTTP, the default port 80. For HTTPS, the default port 443.

abs_path: The Request-URI [1] for the resources is abs_path [4]. The abs_path in this specification is most often of the form ―[/Services][/General Resources][/Special Resources]‖, which is suitable for resources to update or restore device configurations. ―*ID*‖ which appears in the abs_path identifies one resource of a list resource in this specification.

query: The query field is a string of information to be interpreted by the resource. It can include some resource-related parameters. It must be listed in name-value pair syntax (p1=v1&p2=v2&…&pn=vn). Each resource can define a set of parameters. Defining input data which is specific to the resource will be prior than query usage.

4.5 Messages

HTTP messages are used for communication between physical security and video management systems and IP media devices in this specification. In order to configure and control the device, some provisions are specified for these HTTP message.

4.5.1 Connection Header Field

Devices that implement HTTP/1.1 should support persistent connections in order to meet video management systems or client applications' requirements that issue multiple HTTP(s) transactions. HTTP/1.1 is implemented and utilized according to RFC 2616 in the IP devices. For a video management system or client application that uses persistent connection for multiple transactions, it is required to implement ―Connection: Keep-Alive‖ HTTP header field, while also adopt the ―Connection: close‖ HTTP header field for the last transaction of the persistent connection. This process will assume that the application can identify the last request in a sequence of multiple requests.

4.5.2 Authorization and WWW-Authenticate Header Fields

When a video management system or client application sends any request to the device, it must be authenticated by means of Basic Access [5] according to RFC 2617, and thus all the devices are required to support Basic Access. Authorization header field is sent along with each request, and if a user is authenticated, the request will follow the normal execution flow. If client HTTP request is with no authentication credentials, unauthorized HTTP response (401) will be returned with WWW-Authenticate header field.

4.5.3 Entity Body

The Content-Type entity-header field indicates the media type of the entity body. The Content-Type may be designated as —application/xml;charset='UTF-8'‖, —application/octet-stream‖, etc.

For configuration information, the Content-Type is usually—application/xml; charset='UTF-8'‖. For example,

HTTP Request Message

```
GET          /System/status
HTTP/1.1
…
```

HTTP Response Message

```
HTTP/1.1   200
OK
…

Content-Type: application/xml; charset=‖UTF-8‖
…
```

HTTP Request Message

```
PUT /System/configurationData HTTP/1.1
…
Content-Type: application/octet-stream
…
[proprietary configuration file data content ]
```

HTTP Response Message

```
HTTP/1.1 200 OK
…

Content-Type: application/xml; charset=‖UTF-8‖
…
<?xml version=‖1.0‖ encoding=‖UTF-8‖?>
<ResponseStatus   version=—1.0‖   xmlns=—http://www.hikvision.com/ver10/XMLSchema‖>
…
</ResponseStatus>
```

4.5.4 Operations

Different resources will specify different operation.

- The set device configuration‖ resources use PUT operation. If there is an XML block parameter for the request, the inbound XML format is defined according to a resource-special XML schema. Request status will be returned by the XML response information of the device, and can be used for indicating the PUT operation status. The responded XML format is defined by ―XML Response Schema‖ (please refer to section 4.5.5 for detail description). After the device configuration is updated successfully, it will return an XML response with status code ―OK‖; while another status code will be used for indicating unsuccessful operations. In either case, the device only responses after it is ready to continue normal operation, i.e. accepting streaming request, receiving configuration commands, etc.
- The get device configuration‖ resources use GET operation. After a successful GET operation, the result will be returned in XML format according to the resource description. For an unsuccessful request (i.e. users is not authenticated), the result will be returned in XML format according to ―XML Response Schema‖.
- Resources to create device configurations information will use the POST operation. If there is an XML block parameter for the request, the inbound XML format is defined according to a resource-special XML schema. The request status will be indicated by the XML response information returned from the device, and can be used to indicate the status of the POST operation. This XML format is defined according to ―XML Response Schema‖ (see section 4.5.5 for details). After successfully creating the data, the device returns an XML response with status code ―OK‖. A separate status code is used for unsuccessful operations.
- Resources to delete device configurations information will use the Delete operation. If successful, the result will be returned an XML response with status code ―OK‖. A separate status code is used for unsuccessful operations. This XML format is defined according to ―XML Response Schema‖ (see section 4.5.5 for details).
- Data uploading resources (i.e. firmware upgrade, import configuration, etc.) will use PUT operation. The content of the data will be stored in the body of the HTTP request. If successful, the result will be returned an XML response with status code ―OK‖. A separate status code is used for unsuccessful operations. This XML format is defined according to ―XML Response Schema‖ (see section 4.5.5 for details).
- Data receiving resources (i.e. export configuration file) use GET operation. If successful, the result will be returned the data according to the resource description. An XML block is used for unsuccessful operations. This XML format is defined according to ―XML Response Schema‖ (see section 4.5.5 for details).
- For Special Resources, GET operation will be used. For more detailed

description see Section 6.

If there is an XML block for the HTTP request or response, the Content-Type and Content-Length will be set in the headers of the HTTP message.

5 Device discovery

The IP devices support Universal Plug and Play (UPnP) technology to discovery/locate themselves. A UPnP compatible device will automatically announce its network address, supported devices and services types when connected to a network, and therefore becoming ―plug-and-play‖ by allowing clients recognize those information and begin using this device immediately.if you want to get more information about UPnP,please refer to《Discovery version 1.0.0.docx》.

6 Resource Description

6.1.1 Resource Description Outline

Each resource in this specification is defined using the following format.

Resource_URI	***Type Version***
Operation_Name	***User Lever***
Description	*Description of the operation.*
Query	*Indicates the name/value pairs (p1, p2, p3,…,pn) for the resource.*
Inbound Data	*Indicates inbound data for the resources.*
Success Return	*the Type (if present) and the name of XML Data Block*
Notes: *describes any special processing rules for the resource.*	

Type refers to —Special Resource‖, —Service‖ and —General Resource‖.

Version is used to determine the version of the protocol. The version number shall be set to —1.0‖ in this specification.

Operation_Name refers to —GET‖, —PUT‖, —POST‖ and —DELETE‖.

Inbound Data includes three types as follows:

- NONE –no input data
- DataBlock – the name of an XML Data Block. Datablocks used here must be defined according to the specification.
- Mime type – mime type for the input data in the HTTP payload. Remark:—application/ xml‖ is not a valid mime type.

If a device does not support particular XML tags or blocks, then it may not be supported by the resource operations.

Generally, if a field is not provided in the inbound XML, then its current values shall not be modified in the device's repository.

If a required field did not exist in the device's repository, then it must be provided in the applicable resource operations.

Success Return and Error Return detailed description see Section 8.

6.1.2 Annotation

The XML Data Blocks described in this document contains annotations for the field's properties. Please refer to the XML schema definitions for detail description。

The Following annotation content is inserted into the comments to describe the data carried in the field.

Table 6

Annotation	Description
req	Required field.
opt	Optional field. For data uploaded to the device, if the field is present but the device does not support it, it should be ignored.
dep	This field is required depending on the value of another field.
ro	Read-only. For XML data that is both read and written to the device, this field is only present in XML returned from the device. If this field is present in XML uploaded to the device, it should be ignored.
wo	Write-only. This field is only present in XML that can be uploaded to the device. This field should never be present in data returned from the device. [This is used for uploading passwords].
xs:<type>	A type defined in XML Schema Part 2: Datatypes Second Edition, see http://www.w3.org/TR/xmlschema-2

Remark: optional XML structures may contain required fields for the operation, which mean that even if the entire XML block is optional, some of its contained fields may still be necessary if required.

7 Error Handling

As with any other protocol, errors may occur during communications, protocol or message processing, and the specification classifies error handling into categories below:

- Protocol Errors, which are result of an incorrectly formed protocol message. Protocol Errors may contain header value or be received in an not expected or experience a socket timeout. To indicate and interpret protocol error, HTTP protocol has defined a set of standard status codes [e.g., 1xx, 2xx, 3xx, 4xx, 5xx]. According to this specification, the IP devices will use appropriate HTTP protocol defined status codes for error reporting and when received handle accordingly.
- Application Errors, which are generated as a result of REST operations errors. All such application errors must be reported and handled through HTTP messages. The following table indicates the mapping relationship between HTTP status codes and REST operations, and also the information contained in response header and bodies.

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
200	―OK‖-The request has succeeded. Header Notes: None Body Notes: The requested resource will be returned in the body.	√	√		√
201	―Created‖- The request has created a new resource. Header Notes: The Location header contains the URI of the newly created resource. Body Notes: The response returns an entity describing the newly created resource.		√	√	
204	―No Content‖ – The request succeeded, but there is no data to return. Header Notes: None Body Notes: No body is allowed.		√		√

301	―Moved Permanently‖ – The requested resource has moved permanently. Header Notes: The Location Header contains the URI of the new location. Body Notes: The body may contain the new resource	√			
302	―Found‖ – The requested resource should be accessed through this location, but the resource actually lives at another location. This is typically used to set up an alias.	√			

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
	Header Notes: The Location header contains the URI of the resource. Body Notes: The body may contain the new resource				
	―Bad Request‖ – The request was badly formed. This is commonly used for creating or updating a resource, but the data was incomplete or incorrect. Header Notes: The Reason-				
	―Unauthorized‖ – The request requires user authentication to access this resource. If the request contains invalid authentication data, this code is sent.				

403	—Forbiddenǁ – The request is not allowed because the server is refusing to fill the request. A header suggests to the client when to try resubmitting the	√	√	√	√

When setting the configuration, it will return a XML with an error code。Such as：

ResponseStatus XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<ResponseStatus Version="1.0" xmlns="http://www.zwcloud.wang/ver10/XMLSchema">
        <requestURL>/Network/Interfaces/2</requestURL>
        <statusCode>0</statusCode>
</ResponseStatus>
```

<requestURL> indicates URI operation。
<statusCode> represents the return value by the device。

StatusCode	Description
0	Success
-1	Decode error, maybe the XML and URL are not corrisponding
-2	Decode error, XML <req> tag may be missing
-3	Unsupported operation
-4	Decode error, wrong channel identifier
-5	Parameter error
-6	Query log busy
-7	Query log error
-8	SDK call failed
-9	Memory overflow
-10	URL Error
-11	User overflow
-12	Invalid operation
-13	User already exists
-14	Camera is busy with previous event handling

NOTE：For the upgrade: any if return value is greater than zero indicates an update failure.

8 Video Streams and Stream Access

The camera provides 3 parallel RTSP streams (main, first substream, second substram) and 1 Motion JPEG stream which shares the picture settings with the second substream. The Motion JPEG stream is also used for the snapshop picture generation.

Valid parameters for the streams:

Stream	Resolutions	Framerate [fps]	Bitrate [Kbps]
mainstream	1920x1080* 1024x768 640x512	1-25 (25)	512-12288 (2048)
1st substream	704x576 640x360* 320x240	1-25 (25)	128-2048 (512)

Default valure are indicated by * or in brackets.

The URLs for the RTSP streams are:

1. rtsp://ip:port/live/0/main
2. rtsp://ip:port/live/0/sub

9 Services and General Resources

9.1 /Network

/Network	General Resource v1.0
GET	Viewer
Description	Get network configuration
Query	None
Inbound Data	None
Success Return	**Network**
PUT	Viewer
Description	Set network configuration
Query	None
Inbound Data	**Network**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

Network XML Block

```
<Network    version = “1.0” xmlns = “”>
        <PPPOE>    <!-- opt -->
        <DDNS>    <!-- opt-->
        <UPNP>      <!-- opt -->
        <EMail>      <!-- opt-->
        <Port>      <!-- opt-->
        <P2P>    <!-- opt-->
        <Interfaces><!--opt-->
</Network>
```

9.1.1 /Network/DDNS

/Network/DDNS	General Resource v1.0
GET	Viewer
Description	Get DDNS configuration
Query	None
Inbound Data	None
Success Return	**DDNS**

PUT	Viewer
Description	Set DDNS configuration
Query	None
Inbound Data	**DDNS**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

DDNS XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DDNS>
        <ServerList/><!-- opt -->
        <Config/><!--req-->
        <Status/><!--opt-->
</DDNS>
```

9.1.1.1/Network/DDNS/ServerList

/Network/DDNS/ServerList	General Resource v1.0
GET	Viewer
Description	Get DDNS providers list
Query	None
Inbound Data	None
Success Return	**ServerList**
Notes:	

ServerList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
< ServerList>
        <Server><!--opt-->
              <ServerName><!--req,xs:string--></ServerName>
        </Server>
</ ServerList>
```

9.1.1.2/Network/DDNS/Config

/Network/DDNS/Config	General Resource v1.0

GET	Viewer
Description	Get DDNS configuration
Query	None
Inbound Data	None
Success Return	**Config**
PUT	Viewer
Description	Set DDNS configuration
Query	None
Inbound Data	**Config**
Success Return	**ResponseStatus**

Notes:

The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.

When retrieving data, the password field is not displayed. For SET, the password will be only changed if the field contains data.

The value of <ProtocolName> but me in the list of supported providers.

Configuration will be activated only if <Enable> is set to true.

The <ServerPort> field is currently not supported and must be 0.

DDNS Confg XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Config>
        <Enable><!--req,xs:boolean--></Enable>
        <ProtocolName>
                <!--req,xs:string,”the string is the one of <ServerList>”>
        </ProtocolName>
        <ServerPort><!--req,xs:integer--></ServerPort>
        <DomainName><!--req,xs:string--></DomainName>
        <UserName><!--req,xs:string--></UserName>
        <Password><!--wo,req,xs:string--></Password>
        <RefreshIntervals><!--req,xs:integer, in sec--></RefreshInterval>
</Config>
```

9.1.1.3 /Network/DDNS/Status

/Network/DDNS/ Status	General Resource v1.0
GET	Viewer
Description	Get DDNS status information
Query	None
Inbound Data	None
Success Return	**Status**
Notes:	

DDNS Status XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Status>
    <State>
        <!--ro,req,xs:string, “stopped,Registering ,Registered ,stopping”-->
    </State>
</Status>
```

9.1.2 /Network/UPNP

/Network/UPNP	General Resource v1.0
GET	Viewer
Description	Get UPNP port mapping configuration
Query	None
Inbound Data	None
Success Return	**UPNP**
PUT	Viewer
Description	Set UPNP port mapping configuration
Query	None
Inbound Data	**UPNP**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

UPNP XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<UPNP>
        <Config/><!--req-->
        <Status/><!--opt-->
</UPNP>
```

9.1.2.1/Network/UPNP/Config

/Network/UPNP/Config	General Resource v1.0
GET	Viewer
Description	Get UPNP configuration
Query	None
Inbound Data	None
Success Return	**Config**
PUT	Viewer
Description	Set UPNP configuration
Query	None
Inbound Data	**Config**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

UPNP Config XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Config>
    <Enable><!--req,xs:boolean--></Enable>
    <WanIPAddress><!--req,xs:string--></WanIPAddress>
    <LocalHTTPPort><!--req,xs:integer--></LocalHTTPPort>
    <ExportHTTPPort><!--req,xs:integer--></ExportHTTPPort>
    <LocalMobliePort><!--req,xs:integer--></LocalMobliePort>
    <ExportMobliePort><!--req,xs:integer--></ExportMobliePort>
    <LocalCommandPort><!--req,xs:integer--></LocalCommandPort>
    <ExportCommandPort><!--req,xs:integer--></ExportCommandPort>
    <LocalMediaPort><!--req,xs:integer--></LocalMediaPort>
    <ExportMediaPort><!--req,xs:integer--></ExportMediaPort>
</Config>
```

9.1.2.2 /Network/UPNP/Status

/Network/UPNP/Status	General Resource v1.0
GET	Viewer
Description	Get UPNP status information
Query	None
Inbound Data	None
Success Return	**Status**
Notes:	

UPNP Status XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Status>
    <HTTPPortState><!--ro,req,xs:string,”success,failed,mapping” --></HTTPPortState>
    <MobliePortState><!--ro,req,xs:string,”success,failed,mapping” --></HTTPMoblieState>
    <CommandPortState><!--ro,req,xs:string,”success,failed,mapping”--></CommandPortState>
    <MediaPortState><!--ro,req,xs:string,”success,failed,mapping”--></MediaPortState>
</Status>
```

9.1.3 /Network/EMail

/Network/EMail	General Resource v1.0
GET	Viewer
Description	Get email client settings
Query	None
Inbound Data	None
Success Return	**EMail**
PUT	Viewer
Description	Set email client settings
Query	None
Inbound Data	**EMail**
Success Return	**ResponseStatus**

Notes:

The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.

If <AllDay> is true , <TimeBlock> field is invalid, if <AllDay> is false , <TimeBlock> field is valid.

A day is divided into 48 blocks of 30 minutes which can be either active (1) or inactive (0).

The <SenderEmailPassword> content will not be retrieved by the GET command. The value will be only changed if the password field of the PUT comman contains valid data.

EMail XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<EMail>
    <Enable><!--req,xs:boolean--></Enable>
    <SMTPServer><!--req,xs:string--></SMTPServer>
    <SMTPPort><!--req,xs:string--></SMTPPort>
    <SMTPAuthMethod><!--req,xs:string,None,Login--></SMTPAuthMethod>
    <SMTPTLSConfig><!--req-->
        <SMTPTLS><!--req,xs:string,”off,on”--></SMTPTLS>
    </SMTPTLSConfig>
    <SenderEmailAddress><!--req,xs:string--></ SenderEmailAddress>
    <SenderEmailPassword><!--opt,xs:string--></ SenderEmailPassword>
    <ReceiverList><!--req-->
        <Receiver><!--opt-->
            <ID><!--req,xs:integer--></ID>
            <EmailAddress><!--req,xs:string--></EmailAddress>
        </Receiver>
    </ReceiverList>
    <AttachedSnapshot><!--req,xs:boolean--></AttachedSnapshot>
    <RefreshIntervals><!--req,xs:integer, in sec--></RefreshIntervals>
    <Schedule><!--opt-->
        <AllDay><req,xs:boolean></AllDay>
        <TimeBlockList>
            <TimeBlock_0><!req,xs:string></Block_0>
            <TimeBlock_1><!req,xs:string></Block_1>
            <TimeBlock_2><!req,xs:string></Block_2>
            <TimeBlock_3><!req,xs:string></Block_3>
            <TimeBlock_4><!req,xs:string></Block_4>
            <TimeBlock_5><!req,xs:string></Block_5>
            <TimeBlock_6><!req,xs:string></Block_6>
        <TimeBlockList>
    </Schedule>
</EMail>
```

9.1.4 /Network/Port

/Network/Port	General Resource v1.0
GET	Viewer
Description	Get IP ports configuration
Query	None
Inbound Data	None
Success Return	**Port**
PUT	Viewer
Description	Set IP ports configuration
Query	None
Inbound Data	**Port**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests. The device will be restarted after saving the configuration.	

Port XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Port>
      <HTTP><!--req,xs:integer--></HTTPPort>
      <Command><!--req,xs:integer--></CommandPort>
      <Media><!--req,xs:integer--></MediaPort>
      <Mobile><!--req,xs:integer--></MobilePort>
      <RTP><!--req,xs:integer--></RtpPort>
      <Discovery><!--req,xs:integer--></Discovery>
</Port>
```

9.1.5 /Network/P2P

/Network/P2P	General Resource v1.0
GET	Viewer
Description	Get P2P service status and configuration
Query	None
Inbound Data	None
Success Return	
PUT	Viewer
Description	Set P2P service status and configuration
Query	None

Inbound Data	
Success Return	ResponseStaus **ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests. The device will be restarted after saving the configuration.	

P2P XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<P2P>
        <Enable><!--req,xs:boolean--></Enable>
        <UUID><!--req,xs:string--></UUID>
        <Status><!--ro,opt,xs:string,”offline,internet-ready,intranet-ready” --></Status>
</P2P>
```

9.1.6 /Network/FTP

/Network/FTP	General Resource v1.0
GET	Viewer
Description	Get FTP client settings
Query	None
Inbound Data	None
Success Return	
PUT	Viewer
Description	Set FTP client settings
Query	None
Inbound Data	
Success Return	ResponseStaus **ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests. Valid values for <TransferContentTypeMask>: "00" disable "01" only upload picture "10" only upload video "11" both upload picture and video When retrieving data, the password field is not displayed. For SET, the password will be only changed if the field contains data.	

P2P XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<FTP Version="1.0" xmlns="http://www.zwcloud.wang/ver10/XMLSchema">
        <Enable><!—req,xs:boolean--></Enable>
        <UserName><!—req,xs:string--></UserName>
        <Password><!--wo,opt,xs:string--></Password>
        <IPAddress><!—req,xs:string--></IPAddress>
        <Port><!—req,xs:integer--></Port>
        <TransferContentTypeMask><!—req,xs:string--></TransferContentTypeMask>
        <StorageDirectory><!—req,xs:string--></StorageDirectory>
        <Status><!—opt,xs:string,”not connect”,”connected”--></Status>
</FTP>
```

9.1.7 /Network/Interfaces

/Network/Interfaces	General Resource v1.0
GET	Viewer
Description	Get all network interfaces
Query	None
Inbound Data	None
Success Return	**InterfaceList**
PUT	Viewer
Description	Set all network interfaces
Query	None
Inbound Data	**InterfaceList**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

Network interface XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<InterfaceList>
        <Interface><!--opt-->
</InterfaceList>
```

9.1.7.1/Network/Interfaces/NetworkCardID

/Network/Interfaces/NetworkCardID	General Resource v1.0
GET	Viewer
Description	Get configuration of specific network interface
Query	None
Inbound Data	None
Success Return	
PUT	Viewer
Description	Set configuration of specific network interface
Query	None
Inbound Data	
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests. By default, "NetworkCardID" 1 is the wired network interface and 2 is the wireless interface, if applicable.	

Wire

```
<?xml version="1.0" encoding="utf-8"?>
<Interface>
        <ID><!--req,xs:integer--></ID>
        <IPAddress/><!--req-->
</Interface>
```

9.1.7.1.1 /Network/Interfaces/NetworkCardID/IPAddress

/Network/Interfaces/NetworkCardID/IPAddress	General Resource v1.0
GET	Viewer
Description	Get IP address of a specifc interface
Query	None
Inbound Data	None
Success Return	
PUT	Viewer
Description	Set IP address of a specifc interface
Query	None
Inbound Data	
Success Return	**ResponseStatus**
Notes:	

The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.

IPAddress XML BLock

```
<?xml version="1.0" encoding="utf-8"?>
<IPAddress>
        <ipVersion><!--ro,opt,xs:string,V4--></ipVersion>
                <ipAddress><!--req,xs:string--></ipAddress>
        <MacAddress><!--ro,opt,xs:string,readonly--></MacAddress>
        <SubnetMask>
                <!-- req, xs:string, subnet mask for IPv4 address -->
          </SubnetMask>
        <Gateway><!--req,xs:string--></ Gateway>
        <DHCP/><!--req-->
        <DNS/><!--req-->
</IPAddress>
```

9.1.7.1.1.1 /Network/Interfaces/NetworkCardID/IPAddress/DHCP

/Network/Interfaces/NetworkCardID/IPAddress/DHCP	General Resource v1.0
GET	Viewer
Description	Get the DHCP of a particular network interface.
Query	None
Inbound Data	None
Success Return	**DHCP**
PUT	Viewer
Description	Set the DHCP of a particular network interface.
Query	None
Inbound Data	**DHCP**
Success Return	**ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

DHCP XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DHCP>
        <Enable><!--req,xs:boolean--></Enable>
        <Status><!--ro,opt,xs:string,"initializing,obtaining,obtained" --></Status>
</DHCP>
```

9.1.7.1.1.2 /Network/Interfaces/NetworkCardID/IPAddress/DNS

/Network/Interfaces/NetworkCardID/IPAddress/DNS	General Resource v1.0
GET	Viewer
Description	Get DNS configuration of a specific network interface
Query	None
Inbound Data	None
Success Return	**DNS**
PUT	Viewer
Description	Set DNS configuration of a specific network interface
Query	None
Inbound Data	**DNS**
Success Return	ResponseStaus **ResponseStatus**
Notes: The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

DNS XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DNS>
        <Enable><!--req,xs:boolean--></Enable>
        <PrimaryDNS><!--req,xs:string--></PrimaryDNS>
        <SecondaryDNS><!--req,xs:string--></SecondaryDNS>
</DNS>
```

9.1.7.1.2 /Network/Interfaces/ NetworkCardID/WIFIAccessPointList

/Network/Interfaces/NetworkCardID/WIFIAccessPointList	General Resource v1.0
GET	Viewer
Description	Get WiFi access point list
Query	None
Inbound Data	None
Success Return	**WIFIAccessPointList**
Notes: Only applicable on wireless network interfaces. WirelessMode: 1. legacy 11B only 2. legacy 11A only 3. legacy 11a/b/g mixed	

4. legacy 11G only
5. 11ABGN mixed
6. 11N only
7. 11GN mixed
8. 11AN mixed
9. 11BGN mixed

WIFIAccessPointList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<WIFIAccessPointList><!--req-->
  <AccessPoint><!--opt-->
    <ID><!--ro,req,xs:integer--></ID>
    <SSID><!ro,req,xs:string--></SSID>
    <MACAddress><!--ro,req,xs:string--></MACAddress>
    <SignalValue><!--ro,req,xs:integer,0-100--></SignalValue>
    <Authentication><!--ro,req,xs:string,None,shared,WPAPSK,WPA2PSK,
    WPANONE,WPA,WPA2--></Authentication>
    <EncryptType><!--ro,req,xs:string,NONE,WEP,TKIP,AES--></EncryptType>
    <WirelessMode><!--ro,req,xs:integer--></WirelessMode>
   </AccessPoint>
</WIFIAccessPointList>
```

9.1.7.1.3 /Network/Interfaces/NetworkCardID/Wireless

/Network/Interfaces/NetworkCardID/Wireless	General Resource v1.0
GET	Viewer
Description	Get WiFi configuration of a specific network interface
Query	None
Inbound Data	None
Success Return	**ConnectionConfig**
PUT	Viewer
Description	Set WiFi configuration of a specific network interface
Query	None
Inbound Data	**ConnectionConfig**
Success Return	**ResponseStatus**
Notes: Only applicable on wireless network interfaces. The network configuration is stored asynchronously. Please leave at least 5 seconds between 2 configuration requests.	

If <AuthMode> mode is OPEN or SHARED, <WEP> is applied for the key, if <AuthMode> model is WPAPSK or WPA2PSK, <WPA> is applied. Inactive sections will be shown in any case.

If <AuthMode> is WPAPSK or WPA2PSK, <keyvalue> in <WPA> shall not be empty.

The value of <KeyType> in <WPA> must be “ASCII”.

The use of <KeyValue> and <KeyValueB64> is alternative, one of both fields is mandatory.

WIFIConnectionConfig XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<ConnectionConfig> <!--opt-->
    <Enable><!--req,xs:integer,”disconnected,connected” --></Enable>
    <SSID><!--req,xs:string--></SSID>
    <MACAddress><!--ro,req,xs:string--></MACAddress>
    <NetworkType><!--req,xs:string,managed,Ad-hoc--></NetworkType>
    <AuthMode><!--req,xs:string,OPEN,SHRED,WEPAUTO,WPAPSK,WPA2PSK,
    WPANONE,WPA,WPA2,NOPASSWORD--></AuthMode>
    <WEP><!--dep,depends on <AuthMode>-->
        <KeyID><!--req,xs:integer,”1-4”--></KeyID>
        <keyList>
           <Key>
              <ID><!—req,xs:integer--></ID>
              <KeyType><!—req,xs:string,”HEX,ASCII”--></KeyType>
              <KeyValue><!--opt,xs:string--></KeyValue>
              <KeyValueB64><!--req,xs:string,”Base 64 encoded key string”--></KeyValue>

           </Key>
        </keyList>
    </WEP>
    <WPA><!--dep,depends on <AuthMode>-->
        <EncryptType><!--req,xs:string,NONE,WEP,TKIP,AES--></EncryptType>
        <Key>
           <KeyType><!--req,xs:string,”ASCII”--></KeyType>
           <KeyValue><!--req,xs:string,”the length of the string is 8-63”--></KeyValue>
           <KeyValueB64><!--req,xs:string,”Base 64 encoded key string”--></KeyValue>

        </key>
    </WPA>
</ConnectionConfig>
```

9.1.7.1.4 /Network/Interfaces/NetworkCardID/WIFIStatus

/Network/Interfaces/NetworkCardID/WIFIStatus	General Resource v1.0
GET	Viewer
Description	Get WiFi status information for a specific network interface
Query	None
Inbound Data	None
Success Return	**WIFIStatus**
Notes: Only applicable on wireless network interfaces.	

WIFIStatus XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<WIFIStatus><!--req-->
        <State><!--ro,req,xs:string,”open,close,notsupport” --></State>
</WIFIStatus>
```

9.1.8 /Network/WifiParamTest

/Network/WifiParamTest	General Resource v1.0
PUT	Viewer
Description	Check WiFi configuration parameters
Query	None
Inbound Data	**ConnectionConfig**
Success Return	**ResponseStatus**
Notes: It is advisable to use this command before applying the WiFi settings. To check the result, run the WiFiParamTestStatus command 30 seconds after this command.	

Network interface XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<ConnectionConfig><!--opt-->
<SSID><!--req,xs:string--></SSID>
<MACAddress />
<NetworkType><!--req,xs:string,managed,Ad-hoc--></NetworkType>
<AuthMode><!--req,xs:string,OPEN,SHRED,WEPAUTO,WPAPSK,WPA2PSK,
WPANONE,WPA,WPA2,NOPASSWORD--></AuthMode>
<WEP><!--dep,depends on <AuthMode>-->
<KeyID><!--req,xs:integer,”1-4”--></KeyID>
<keyList>
<Key>
<ID><!—req,xs:integer--></ID>
<KeyType><!—req,xs:string,”HEX,ASCII”--></KeyType>
<KeyValue><!--opt,xs:string--></KeyValue>
</Key>
</keyList>
</WEP>
<WPA><!--dep,depends on <AuthMode>-->
<EncryptType><!--req,xs:string,NONE,WEP,TKIP,AES--></EncryptType>
<Key>
<KeyType><!--req,xs:string,”ASCII”--></KeyType>
<KeyValue><!--req,xs:string,”the length of the string is 8-63”--></KeyValue>
</key>
</WPA>
</ConnectionConfig>

Example: <?xml version="1.0" encoding="UTF-8" ?>
<ConnectionConfig Version="1.0">
        <SSID>TestWiFi</SSID>
        <MACAddress />
```

```
    <NetworkType>managed</NetworkType>
    <AuthMode>WPAPSK</AuthMode>
    <WEP Version="1.0">
        <KeyID>0</KeyID>
        <keyList Version="1.0">
            <Key Version="1.0">
                <ID>1</ID>
                <KeyType>HEX</KeyType>
                <KeyValue />
            </Key>
            <Key Version="1.0">
                <ID>2</ID>
                <KeyType>HEX</KeyType>
                <KeyValue />
            </Key>
            <Key Version="1.0">
                <ID>3</ID>
                <KeyType>HEX</KeyType>
                <KeyValue />
            </Key>
            <Key Version="1.0">
                <ID>4</ID>
                <KeyType>HEX</KeyType>
                <KeyValue />
            </Key>
        </keyList>
    </WEP>
    <WPA Version="1.0">
        <EncryptType>TKIP</EncryptType>
        <Key Version="1.0">
            <ID>1</ID>
            <KeyType>ASCII</KeyType>
            <KeyValue>3008993007</KeyValue>
        </Key>
    </WPA>
</ConnectionConfig>
```

9.1.9 /Network/WifiParamTestStatus

/Network/WifiParamTestStatus	General Resource v1.0
GET	Viewer
Description	Get the result of the WiFi settings check
Query	None
Inbound Data	None
Success Return	**WiFi test status**
Notes: Returns the result of the WiFiParamTest command.	

Network interface XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<WIFIStatusForTest Version="1.0">
<Status><!--req,xs:string,”disconnect,connected”--></Status>
</WIFIStatusForTest>
```

9.2 /System

/System	General Resource v1.0

9.2.1 /System/Time

/System/Time	General Resource v1.0
GET	Viewer
Description	Get time configuration
Query	None
Inbound Data	None
Success Return	**Time**
PUT	Viewer
Description	Set time configuration
Query	None
Inbound Data	**Time**
Success Return	ResponseStaus **ResponseStatus**
Notes: The date and time string has ISO 8601 format.	

Time XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Time>
    <DateTimeFormat>
          <!--
          req,string,YYYYMMDDWhhmmss,YYYYMMDDhhmmss,MMDDYYYYWhhmmss,MMDDYY
          YYhhmmss,DDMMYYYYWhhmmss,DDMMYYYYhhmmss-->
    </DateTimeFormat>
    <TimeFormat><!--req,xs:string,12hour,24hour--></TimeFormat>
    <SystemTime><!--req,xs:datetime,” 20040503T173008+08”--></SystemTime>
</Time>
```

9.2.2 /System/NTP

/System/NTP	General Resource v1.0
GET	Viewer
Description	Get NTP service status and configuration
Query	None
Inbound Data	None

Success Return	**NTP**
PUT	Viewer
Description	Set NTP service status and configuration
Query	None
Inbound Data	**NTP**
Success Return	**ResponseStatus**
Notes: TimeZone: time zones are defined by POSIX 1003.1 section 8.3 time zone notations. the +/- value indicates the difference from UTC. Example: EST+5EDT01:00:00,M3.2.0/02:00:00,M11.1.0/02:00:00 Defines eastern standard time as with a GMT-5 offset. Daylight savings time is called, is one hour later and begins on the second Sunday of March at 2 am and ends on the first Sunday of November at 2 am. CET-1CEST01:00:00,M3.5.0/02:00:00,M10.5.0/03:00:00 Defines central European time as GMT+1 with a one-hour daylight savings time (—C) that starts on the last Sunday in March at 2 am and ends on the last Sunday in October at 3 am.	

NTP XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<NTP>
    <Enable><!—req,xs:boolean--></Enable>
    <ServerName><!—req,xs:string,"example: time.windows.com"--></ServerName>
    <TimeZone><!--req, xs:string, POSIX time zone string,"CST-9:00:00"--></TimeZone>
</NTP>
```

9.2.3 /System/DST

/System/DST	General Resource v1.0
GET	Viewer
Description	Get DST configuration
Query	None
Inbound Data	None
Success Return	**DST**
PUT	Viewer
Description	Set DST configuration
Query	None
Inbound Data	**DST**
Success Return	**ResponseStatus**
Notes:	

If <Mode> is “date”, use <DateModeStart> and <DateModeStop>
If -<Mode> is “week”, use < WeekModeStart > and < WeekModeStop >
< TimeDifference > in seconds
<DateModeStart> and <DateModeStop>: ISO 8601 date/time string is accepted and returned
<WeekModeStart> and <WeekModeStop>: the format is “month-week-day-hour-minute-second” in numbers

NTP XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<DST Version="1.0" xmlns="http://www.zwcloud.wang/ver10/XMLSchema">
      <Enable><!—req,xs:boolean--></Enable>
      <Mode><!—req,xs:string,”date,week”--></Mode>
      <TimeDifference>req,xs:string, </TimeDifference>
      <DateModeStart><!--req,xs:datetime,” 20040503T173008+08”--></DateModeStart>
      <DateModeStop><!--req,xs:datetime,” 20040503T173008+08”--></DateModeStop>
      <WeekModeStart><!--req,xs:string,” 3-1-1-2-3-6”--></WeekModeStart>
      <WeekModeStop><!--req,xs:string,” 2-4-5-7-8-8”--></WeekModeStop>
</DST>
```

9.2.4 /System/DeviceInfo

/System/DeviceInfo	General Resource v1.0
GET	Viewer
Description	Get device information.
Query	None
Inbound Data	None
Success Return	**DeviceInfo**
PUT	Viewer
Description	Set device name
Query	None
Inbound Data	**DeviceInfo**
Success Return	**ResponseStatus**
Notes: Only <DeviceName> can be set, all other parameters are read-only.	

DeviceInfo XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DeviceInfo>
     <DeviceName><!--rw,req,xs:string--></DeviceName>
     <BoardType><!--ro,req,xs:integer--></BoardType>
     <DeviceTypeName><!--ro,req,xs:string--></DeviceTypeName>
```

```
    <SerialNum><!--ro,req,xs:string--></SerialNum>
    <HardWareVersion><!--ro,req,xs:string--></HardWareVersion>
    <SoftWareVersion><!--ro,req,xs:string--></SoftWareVersion>
    <VideoInputChannels><!-ro,-req,xs:integer--></VideoChannels>
    <AudioChannels><!--ro,req,xs:integer--></AudioChannels>
    <AlarmInputChannels><!--ro,req,xs:integer--></AlarmChannel>
    <AlarmOutputChannels><!--ro,req,xs:integer--></AlarmOutputChannel>
    <MonitorCount><!--ro,req,xs:integer--></MonitorCount>
    <SpotOutChannels><!--ro,req,xs:integer--></SpotOutChannels>
    <TwoWayAudio><!--ro,req,xs:boolean--></TwoWayAudio>
</DeviceInfo>
```

9.2.5 /System/DeviceStatus

/System/DeviceStatus	General Resource v1.0
GET	Viewer
Description	Get channel status
Query	None
Inbound Data	None
Success Return	**DevicestatusList**
Notes:	

DevicestatusList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DeviceStatusList>
    <DeviceStatus/><!--opt-->
</DeviceStateList>
```

9.2.5.1 /System/DeviceStatus/ChannelID

/System/DeviceStatus/ID	General Resource v1.0
GET	Viewer
Description	Get status of a specific channel
Query	None
Inbound Data	None
Success Return	**DeviceStatus**
Notes: <FirstSubStreamEnable>, <SecondSubStreamEnable> status of 1st, 2nd stream.	

The return value “false” means that the feature is not supported by the camera.

DeviceStatus XML Block

```
<DeviceStatus>
        <ID><!--req,xs:integer--></ID>
        <MainStreamVideoBitrate><!--ro,req,xs:integer --></ MainStreamVideoBitrate>
        <FirstSubStreamEnable><!--ro,req,xs:boolean--></ FirstSubStreamEnable>
        <FirstSubStreamVideoBitrate><!--ro,req,xs:integer--></ FirstSubStreamVideoBitrate>
        <SecondSubStreamEnable><!--ro,req,xs:boolean--></ SecondSubStreamEnable>
        <SecondSubStreamVideoBitrate><!--ro,req,xs:integer--></SecondStreamVideoBitrate>
        <RecordState><!--ro,req,xs:string,”none,alarm,normal,manual,motion”--></RecordState>
        <VideoLostState><!--ro,req,xs:boolean--></VideoLostState>
        <MotionState><!--ro,req,xs:boolean--></MotionState>
        <AlarmInState><!--ro,req,xs:boolean--></AlarmInState>
        <AlarmOutState><!--ro,req,xs:boolean--></AlarmOutState>
</DeviceStatus>
```

9.2.6 /System/FirmwareUpgrade

/System/FirmwareUpgrade	General Resource v1.0
PUT	Viewer
Description	Upgrade device firmware
Query	None
Inbound Data	**Firmware as multipart/form-data**
Success Return	**ResponseStatus**

Notes:

```
HEADER:
        Authorization: Basic user-password hash-base64
HTML FORM DATA
        name=”setting_file”       Content:
                                          filename=”[name of firmware file].pkg”
                                          Firmware file content as blob
```

Example:

```
Content-Disposition: form-data; name="setting_file"; filename="bv-b-20200503.pkg"
Content-Type: application/octet-stream
```

9.2.7 /System/Reboot

/System/Reboot	General Resource v1.0

PUT	Viewer
Description	Reboot the camera
Query	None
Inbound Data	**Opaque Data**
Success Return	**ResponseStatus**
Notes:	

9.2.8 /System/FactoryDefault

/System/FactoryDefault	General Resource v1.0
PUT	Viewer
Description	Restore factory settings
Query	None
Inbound Data	**Opaque Data**
Success Return	**ResponseStatus**
Notes:	

9.2.9 /System/SendTestEmail

/System/SendTestEmail	General Resource v1.0
PUT	Viewer
Description	Send a test email message
Query	None
Inbound Data	**Opaque Data**
Success Return	**ResponseStatus**
Notes: Configure email client first using /Network/Email	

9.2.10 /System/Logging

/System/Logging	General Resource v1.0
GET	Viewer
Description	Retrieve device log
Query	majorType minorType

	startTime stopTime
Inbound Data	None
Success Return	**LogList**

Notes:

“majorType”:

1	System Operation
2	Configuration Operation
3	Record Operation
4	User Managerment
5	Alarm Event
6	Exception Event
7	Storage Management
8	Any Operation

If “majorType” is 1, “minorType” can be:

0	Any System Operation
1	System upgrade
2	PTZ control
3	Illegal shutdown
4	Normal shutdown
5	Startup
6	Remote reboot
7	Clear alarm

If “majorType” is 2, “minorType” can be:

100	Any Configuration Operation
101	Record time table settings
102	Normal settings
103	Encoding settings
104	Network settings
105	Server settings
106	Motion detection settings
107	PTZ decoder settings
108	Restore default settings
109	Set display device parameters
110	Display mode settings
111	Image color settings
112	Auto-maintenance settings
113	Alarm output settings
114	Alarm input settings
115	Function customize settings
116	Import PTZ protocol
117	Export PTZ protocol

118 Import parameters settings
119 Export parameters settings
120 Abnormal parameters settings

If “majorType” is 3, “minorType” can be:
200 Any Record Operation
201 Start manual record
202 Start auto record
203 Stop

If “majorType” is 4, “minorType” can be:
300 Any User Managerment
301 Add user
302 Delete user
303 Modify user
304 User login
305 User logout

If “majorType” is 5, “minorType” can be:
400 Any Alarm Event
401 Motion detection begin
402 Motion detection end
403 Input alarm begin

If “majorType” is 6, “minorType” can be:
500 Any Exception Event
501 IP conflict
502 HDD full
503 HDD error
504 Illegal visit
505 Signal loss
506 Signal restore
507 Network disconnect

If “majorType” is 7, “minorType” can be:
600 All Storage Management
601 Format partition
602 Create partition
603 Delete partition
604 Uninstall HDD
605 HDD group management

The format of “startTime” and “stopTime” is “YYYY-MM-DDThh:mm:ss”. The device supports up to 1000 log entries.

Example:
http://192.168.1.171/System/Logging?majorType=8&minorType=0&startTime=2015-07-28T00:00:00&stopTime=2015-07-29T23:59:00.

LogList XML Block

```
<LogList>
    <Log> <!-- opt -->
        <ID> <!-- req, xs:integer --> </ID>
        <time> <!--req, xs:datetime --> </time>
        <majorType> <!--req, xs:integer --> </majorType>
        <minorType> <!--req, xs:integer --> </minorType>
        <netUser> <!--req, xs:string --> </netUser>
        <hostIPAddress> <!--req, xs:string --> </hostIPAddress>
        <channel> <!--req, xs:integer --> </channel>
    </Log>
</LogList>
```

9.2.11 /System/Users

/System/Users	General Resource v1.0
GET	Viewer
Description	Get user list and parameters
Query	None
Inbound Data	None
Success Return	**UserList**
Notes:	

NTP XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<UserList>
     <User> <!-- opt -->
</UserList>
```

9.2.12 /System/Users/N

/System/Users/N	General Resource v1.0

GET	Viewer
Description	Get configuration of a specific user
Query	None
Inbound Data	None
Success Return	**User**
Notes: “N” corresponds to the user <ID> +1, so to get the data for user 6, N is 7.	

NTP XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<User>
    <ID><!--req,xs:integer--></ID>
    <UserName><!--req,xs:string--></UserName>
    <UserLevel><!--req,xs:string,”Default,Adminstrator,Normal”> </UserLevel>
    <IPAddress><!--ro,opt,xs:string></IPAddress>
    <MACAddress><!--ro,opt,xs:string></MACAddress>
</User>
```

9.2.1 /System/Users/1

/System/Users/1	General Resource v1.0
PUT	**Viewer**
Description	Update user data
Query	None
Inbound Data	**User**
Success Return	**ResponseStatus**
POST	**Viewer**
Description	Add new user
Query	None
Inbound Data	**User**
Success Return	**ResponseStatus**
Delete	**Viewer**
Description	Delete user
Query	None
Inbound Data	**User**
Success Return	**ResponseStatus**
Notes: There is only one admin user with ID 0. Do only add “Normal” users. To add a new user, <ID> is required but it needs to be empty. In the POST request always use /1. <ID>, <IPAddress>, and <MACAddress> must be empty but included in the body.	

For PUT: the only parameter which can be changed is the password. In any case and in addition to the fields mentioned above, also <UserLevel> must be included in the body and it must be empty.

NTP XML Block

```
<?xml version="1.0" encoding="UTF-8" ?>
<User>
      <ID />
      <UserName><!--req,xs:string--></UserName>
      <Password><!--wo,req,xs:string--></ Password>
      <UserLevel><!--req,xs:string,”Normal”> </UserLevel>
      <IPAddress />
      <MACAddress />
</User>
```

9.2.2 /System/ChannelID/RemoteForceIFrame

/System/ChannelID/RemoteForceIFrame	General Resource v1.0
PUT	Viewer
Description	Force i-frame remotely
Query	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

9.2.3 /System/DeviceConfigFile

/System/DeviceConfigFile	General Resource v1.0
GET	Viewer
Description	Retrieve camera’s configuration file
Query	None
Inbound Data	None
Success Return	**Config file**
PUT	Viewer
Description	Upload device configration
Query	None
Inbound Data	**Config file**

Success Return	**ResponseStatus**
Notes:	

9.2.4 /System/DeviceAutoMaintenance

/System/DeviceAutoMaintenance	General Resource v1.0
GET	Viewer
Description	Get automatic camera restart function status
Query	None
Inbound Data	
Success Return	**ResponseStatus**
PUT	Operator
Description	Set automatic camera restart function
Query	None
Inbound Data	
Success Return	**ResponseStatus**
Notes: <Enable>: enable automatic restart function. <Mode>: restart interval whereby <WeekDayMask> applies only if “everyweek” is selected. <WeekDayMask>: mask format “0000000” for the week starting on Sunday.	

AlarmInState XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<AutoMaintenance><!--req-->
        <Enable><!--req,xs:boolean--></Enable>
        <Mode><!--req,xs:string,”everyweek,everyday,once”--></Mode>
        <WeekDayMask><!—req,xs:string,”0000000”--></WeekDayMask>
        <MaintenanceTime><!--req,xs:datetime,” 20040503T173008”-->
        </ MaintenanceTime>
</AutoMaintenance>
```

9.3 /Security

9.3.1 /Security/UserAuth

/Security/UserAuth	General Resource v1.0
PUT	Viewer
Description	Check user credntials
Query	None
Inbound Data	**None**
Success Return	**ResponseStatus**
Notes: Example URL：http://192.168.1.46/Security/UserAuth Data format: PUT /Security/UserAuth HTTP/1.1\r\n Connection:close\r\n Host: 192.168.0.46\r\n Content-Length:\r\n \r\n Username=admin&Password=admin	

9.4 /Streams

/Streams	General Resource v1.0
GET	Viewer
Description	Get stream properties
Query	None
Inbound Data	None
Success Return	**StreamList**
PUT	Viewer
Description	Set stream properties
Query	None
Inbound Data	**StreamList**
Success Return	**ResponseStatus**
Notes:	

Streams XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<StreamList>
        <Stream><!-- opt -->
</StreamList>
```

9.4.1 /Streams/ChannelID

/Streams/ChannelID	General Resource v1.0
GET	Viewer
Description	Get channel properties
Query	None
Inbound Data	None
Success Return	**Stream**
PUT	Viewer
Description	Set channel properties
Query	None
Inbound Data	**Stream**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Stream XML Block

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

```
        <ID><!--opt,xs:integer--></ID>
        <StreamCapability><!--opt-->
        <StreamConfigList>
                <StreamConfig><!--opt-->
        </StreamConfigList>
</Stream>
```

9.4.1.1/Streams/ChannelID/CapabilityV2

/Streams/ChannelID/CapabilityV2	General Resource v1.0
GET	Viewer
Description	Get streaming capabilities
Query	None
Inbound Data	None
Success Return	**StreamCapability**

Notes:

The <ChannelID> in the URI shows the channel number, <ChannelID> starts at 1.

The <ID> under the <StreamCapability> represents the channel ID.

The <ID> under the <MainStreamCapability> indicates the main stream of ID the device supported

The <ID> under the <MainStreamCapability>, <SecondStreamCapabilityList>,

indicates the sub stream of ID the device supported Resolution and maximum frame rate supported by the main stream is not the same, but different main stream supports different sub-stream: Understanding the structure of the flow capacity, it is not the same, that is to say the resolution and maximum frame rate of the sub-stream by the resolution of the main stream to the decision. Therefore, the design follows the structure to represent the encoding capabilities of the device.

StreamCapability XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<StreamCapability>
          <ID><!--req,xs:integer, --></ID>
        <MainStreamCapabilityList><!--ro,req-->
                <MainStreamCapability><!---ro,opt->
                        <ID><!--req,xs:integer, --></ID>
                        <ResolutionWidth><!--ro,opt,xs:integer--></ResolutionWidth>
                        <ResolutionHeigth><!--ro,opt,xs:integer--></ResolutionHeigth>
                        <MaxFrameRate><!--ro,req,xs:integer,--></MaxFrameRate>
                        <SubStreamCapabilityList><!--ro,req-->
                                <StreamCapability><!--ro,opt-->
                                        <ID><!--ro,req,xs:integer--></ID>
                                        <ResolutionWidth><!--ro,opt,xs:integer--></ResolutionWidth>
```

```
                    <ResolutionHeigth><!--ro,opt,xs:integer--></ResolutionHeigth>
                    <MaxFrameRate><!--ro,req,xs:integer,--></MaxFrameRate>
                </StreamCapability>
            </SubStreamCapabilityList>
            <SecondStreamCapabilityList><!--ro,req-->
                 <StreamCapability><!--ro,opt-->
                    <ID><!--ro,req,xs:integer--></ID>
                    <ResolutionWidth><!--ro,opt,xs:integer--></ResolutionWidth>
                    <ResolutionHeigth><!--ro,opt,xs:integer--></ResolutionHeigth>
                    <MaxFrameRate><!--ro,req,xs:integer,--></MaxFrameRate>
                </StreamCapability>
            </SecondStreamCapabilityList>
        </MainStreamCapability>
    </MainStreamCapabilityList>
</StreamCapability>
```

9.4.1.2 /Streams/ChannelID/StreamID

/Streams/ChannelID/StreamID	General Resource v1.0
GET	Viewer
Description	Get streaming properties for StreamID
Query	None
Inbound Data	None
Success Return	**StreamConfig**
PUT	Viewer
Description	Set streaming properties for StreamID
Query	None
Inbound Data	**StreamConfig**
Success Return	ResponseStaus **ResponseStatus**
Notes: The camera only supports <ChannelID> 1. <StreamID> represents stream type, generally "1" represents the main stream, “2” means "the first sub-Stream", “3” means “the second sub-stream”. <FrameRate> field indicates the frame rate, if the value is 25500, which indicates that the frame rate is a full frame rate. <FixedQuality> field represents the quality, the smaller the value, the higher the quality. If <StreamID> corresponds to a sub-stream, then <StreamType>, <VideoCodecType>, <FixedQuality> is invalid, <BitrateType> is always a CBR (constant bit rate). <keyFrameInterval> is not supported. Resolution and frame rate ranges refer to the <StreamCapability>	

If <StreamID> corresponds to a snapshot stream, then <StreamType>, <VideoCodecType>, <FixedQuality> is invalid, <BitrateType> is always a CBR (constant bit rate).

StreamConfig XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<StreamConfig><!--opt-->
    <ID><!--req,xs:integer--><ID>
    <StreamType><!--req,xs:string,”Video,VideoAndAudio”--></StreamType>
    <VideoCodecType><!—ro,opt,xs:string,”H.264”--></VideoCodecType>
    <ResolutionWidth><!--req,xs:integer--></ResolutionWidth>
    <ResolutionHeigth><!--req,xs:integer--></ResolutionHeigth>
    <FrameRate><!--req,xs:integer, maximum frame rate x100--></FrameRate>
    <BitrateType><!--req,xs:string,"CBR,VBR,SmartAVC"--></BitrateType>
    <Bitrate><!--opt, xs:integer, in kbps--></Bitrate>
    <FixedQuality><!--req,xs:integer,"0-5"--></FixedQuality>
    <keyFrameInterval> <!-- opt, xs:integer--> </keyFrameInterval>
    <Transport><!--opt-->
        <RTSPURI><!--ro,req,xs:string--></RTSPURI>
    </Transport>
</StreamConfig>
```

9.4.1.2.1 /Streams/ChannelID/StreamID/Transport

/Streams/ChannelID/StreamID/Transport	General Resource v1.0
GET	Viewer
Description	Get the stream RTSP address
Query	None
Inbound Data	None
Success Return	**Transport**
Notes: The camera only supports <ChannelID> 1. The <StreamID> represents stream type, generally "1" represents the main stream, “2” means "the first sub-Stream". The <RTSPURI> field indicates the address which is access to the stream by the RTSP Protocol.	

Stream Transport XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Transport>
    <RTSPURI><!--ro,req,xs:string--></RTSPURI>
</Transport>
```

9.4.1.2.2 /Streams/ChannelID/StreamID/ReceiveData

/Streams/ChannelID/StreamID/ReceiveData	General Resource v1.0
GET	Viewer
Description	Get the MJPEG stream for the device.
Query	None
Inbound Data	None
Success Return	**Transport**
Notes: M-JPEG stream URL. Currently only the 4th stream of the 1st channel supports this format, so http://[IP]/Streams/1/4/ReceiveData	

9.5 /Snapshot

9.5.1 /Snapshot/ChannelID/RemoteImageCapture

/Snapshot/ChannelID/RemoteImageCapture	General Resource v1.0
GET	Viewer
Description	Get a jpeg picture.
Query	None
Inbound Data	None
Success Return	**ResponseStatus or the JPEG picture stream**
Notes: The camera only supports <ChannelID> 1. “ImageFormat”: 2: jpg (currently only suppoted format) The resolution corresponds to the settings of the 4th stream (Motion JPEG).	

9.6 /Images

/Images	General Resource v1.0
GET	Viewer
Description	Get image configuration
Query	None
Inbound Data	None
Success Return	**ResponseStatus or ImageList**
PUT	Viewer
Description	Set image configuration
Query	None
Inbound Data	**ImageList**
Success Return	**ResponseStatus**
Notes:	

ImageList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<ImageList><!--req-->
        <Image><!--opt-->
</ImageList>
```

9.6.1 /Images/ChannelID

/Images/ ChannelID	General Resource v1.0
GET	Viewer
Description	Get image configuration for a specific channel
Query	None
Inbound Data	None
Success Return	**ResponseStatus or Image**
PUT	Viewer
Description	Set image configuration for a specific channel
Query	None
Inbound Data	**Image**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Image XML Block

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

```
        <ID><!--req,xs:integer--></ID>
        <Scene><!--opt-->
        <Basic/><!--opt-->
        <Color/><!--opt-->
        <Exposure/><!--opt-->
        <WhiteBlance/><!--opt-->
        <IrCutFillter/><!--opt-->
</Image>
```

9.6.1.1/Images/ChannelID/Scene

/Images/ChannelID/Scene	General Resource v1.0
GET	Viewer
Description	Get scene configuration
Query	None
Inbound Data	None
Success Return	**Scene**
PUT	Viewer
Description	Set scene configuration
Query	None
Inbound Data	**Scene**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Scene XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Scene>
    <Support><!—ro,req,xs:boolean></Support>
    <Mode><!—req,xs:string,"standard,soft,vivid,custom"></Mode>
</Scene>
```

9.6.1.2 /Images/ChannelID/Basic

/Images/ChannelID/Basic	General Resource v1.0
GET	Viewer
Description	Get basic image configuration
Query	None
Inbound Data	None
Success Return	**Basic**
PUT	Viewer
Description	Set basic image configuration
Query	None
Inbound Data	**Basic**
Success Return	**ResponseStatus**
Notes:	

Basic XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Basic>
    <powerLineFrequencyMode><!--req,xs:string,50HZ,60HZ --></ powerLineFrequencyMode>
    <NoiseReduce><!--req,xs:string,"close,weak,normal,strong"--></NoiseReduce>
    <BLC><!--req,xs:string,"close,open,notsupport"--></BLC>
    <WDR><!--req,xs:string, "close,weak,normal,strong"--></WDR>
    <FlipMode>
        <!--req,xs:string,"close,leftright,updown,center,notsupport"-->
    </FlipMode>
</Basic>
```

9.6.1.3 /Images/ChannelID/Color

/Images/ChannelID/Color	General Resource v1.0
GET	Viewer
Description	Get image color configuration
Query	None
Inbound Data	None
Success Return	**Color**
PUT	Viewer
Description	Set image color configuration
Query	None
Inbound Data	**Color**

Success Return	ResponseStatus
Notes: The camera only supports <ChannelID> 1. <Mode> shows automatically adjust the brightness.If mode is “auto”, setting the value of the <Brightness> field are unsuccessed, If mode is “manual”, setting the value of the <Brightness> field is valid.For unsupported devices, setting the value of the <Brightness> field is always valid.	

Color XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Color>
    <Mode><!--req,xs:string,”auto,manual,notsupport”--></Mode>
    <Brightness><!--req,xs:integer,0-100--></Bright>
    <Contrast><!--req,xs:integer,0-100--></Contrast>
    <Saturation><!--req,xs:integer,0-100--></Saturation>
    <Hue><!--req,xs:integer,0-100--></Hue>
    <Sharpness><!--req,xs:integer,0-100--></Sharpness>
</Color>
```

9.6.1.4/Images/ChannelID/Exposure

/Images/ChannelID/Exposure	General Resource v1.0
GET	Viewer
Description	Get image exposure configuration
Query	None
Inbound Data	None
Success Return	**Exposure**
PUT	Viewer
Description	Set image exposure configuration
Query	None
Inbound Data	**Exposure**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Exposure XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Exposure>
    <IrisType><!--req,xs:string,”manual, DC-Iris”--></IrisType>
    <shutter>
        <!—req,xs:string,”auto, 1/1, 1/3, 1/8, 1/12, 1/15,1/25,1/30,1/50,1/60,1/100, 1/120,
        1/240, 1/480, 1/960, 1/1000, 1/2000, 1/4000, 1/8000, 1/10000,”-->
        </shutter>
        <Gain><!--req,xs:string,”auto,6dB,12dB,18dB,24dB,30dB,36dB,42dB” --></Gain>
</Exposure>
```

9.6.1.5/Images/ChannelID/WhiteBalance

/Images/ChannelID/WhiteBalance	General Resource v1.0
GET	Viewer
Description	Get image white balance configuration
Query	None
Inbound Data	None
Success Return	**WhiteBlance**
PUT	Viewer
Description	Set image white balance configuration
Query	None
Inbound Data	**WhiteBlance**
Success Return	ResponseStaus **ResponseStatus**
Notes: The camera only supports <ChannelID> 1. <GainRed> and <GainBlue> are applicable only if If <Mode> is “manual”.	

WhiteBlance XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<WhiteBlance>
        <Mode>
                <!--req,xs,string,”auto,incandescent,warm,fluorescent,natural,manual,
                locked”-->
        </ Mode>
        <GainRed><!--dep,xs:integer,0-100--></GainRed>
        <GainBlue><!--dep,xs:integer,0-100--></GainBlue>
</WhiteBlance>
```

9.6.1.6/Images/ChannelID/IrCutFilter

/Images/ChannelID/IrCutFilter	General Resource v1.0
GET	Viewer
Description	Get image IrCutFilter configuration
Query	None
Inbound Data	None
Success Return	**IrCutFilter**
PUT	Viewer
Description	Set image IrCutFilter configuration
Query	None
Inbound Data	**IrCutFilter**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1. If <Mode> is “time”, <StartTime> and <EndTime> are effective, other is not effective.If <Mode> is “initiative” or “passivity”, <Sensitivity> and <SwitchTime> are not effective ,other is not effective. The greater the value of the <Sensitivity>, the more sensitive.	

IrCutFilter XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<IrCutFilter>
        <Mode><!--req,xs:string,”initiative,passivity,day,night,time”--></Mode>
        <DayStartTime>
              <!--dep,xs:time, ”17:30:08”-->
        </ DayStartTime >
        <DayEndTime>
              <!--dep,xs:time, ” 17:30:08”-->
        </ DayStartTime >
        <Sensitivity><!—dep,xs:integer,0-100--></Sensitivity>
        <SwitchTime><!—dep,xs:integer,3s-15s--></SwitchTime>
</IrCutFilter>
```

9.7 /Pictures

/Pictures	General Resource v1.0
GET	Viewer
Description	Get all picture configuration for the device
Query	None
Inbound Data	None
Success Return	**PictureList**
PUT	Viewer
Description	Set all picture configuration for the device
Query	None
Inbound Data	**PictureList**
Success Return	**ResponseStatus**
Notes:	

PictureList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<PictureList>
    <Picture><!—opt-->
</PictureList>
```

9.7.1 /Pictures/ChannelID

/Pictures/ChannelID	General Resource v1.0
GET	Viewer
Description	Get picture configuration
Query	None
Inbound Data	None
Success Return	**Picture**
PUT	Viewer
Description	Set picture configuration
Query	None
Inbound Data	**Picture**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Picture XML Block

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

```
        <ID><!—req,xs:integer--></ID>
        <OSD/><!--req-->
        <VideoLost/><!--req-->
        <Motion/><!--req-->
        <Mosaic/><!--req-->
</Picture>
```

9.7.1.1/Pictures/ChannelID/OSD

/Pictures/ChannelID/OSD	General Resource v1.0
GET	Viewer
Description	Get OSD configuration
Query	None
Inbound Data	None
Success Return	**OSD**
PUT	Viewer
Description	Set OSD configuration
Query	None
Inbound Data	**OSD**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1. Make sure time and name do not overlap.	

OSD XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<OSD>
        <DisplayTime><!--req-->
                <Enable><!--req,xs:boolean--></Enable>
                <PosX><!--req,xs:integer,0-703--></PosX>
                <PosY><!--req,xs:integer,0-575--></PosY>
        </DisplayTime>
        <DisplayName><!--req-->
                <Enable><!--req,xs:boolean--></Enable>
                <PosX><!--req,xs:integer,0-703--></PosX>
                <PosY><!--req,xs:integer,0-575--></PosY>
                <Name><!--req,xs:string,”in Unicode”--></Name>
        </DisplayName>
</OSD>
```

9.7.1.2 /Pictures/ChannelID/Motion

/Pictures/ChannelID/Motion	General Resource v1.0
GET	Viewer
Description	Get motion detection configuration
Query	None
Inbound Data	None
Success Return	**Motion**
PUT	Viewer
Description	Set motion detection configuration
Query	None
Inbound Data	**Motion**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Motion XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Motion>
    <Enable><!--req,xs:boolean--></Enable>
    <Senstive><!--req,xs:string,”low,middle,high”,--></Senstive>
    <Trigger><!--opt-->
    <Schedule><!--opt-->
    <MotionRegionList><!--opt-->
</Motion>
```

9.7.1.3 /Pictures/ChannelID/Motion/RegionsV2

/Pictures/ChannelID/Motion/RegionsV2	General Resource v1.0
GET	Viewer
Description	It is used to get the motion regions configuration of a particular video input channel for the device.
Query	None
Inbound Data	None
Success Return	**ResponseStatusor MotionRegionListV2**
PUT	Viewer
Description	It is used to get the motion regions configuration of a particular video input channel for the device.

Query	None
Inbound Data	**MotionRegionListV2**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1.	

Motion RegionsV2XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<MotionRegionListV2>
    <RegionX_0><!--xs:string,”00000000000000”--></RegionX_0>
    <RegionX_1><!--xs:string,”00000000000000”--></RegionX_1>
    <RegionX_2><!--xs:string,”00000000000000”--></RegionX_2>
    <RegionX_3><!--xs:string,”00000000000000”--></RegionX_3>
    <RegionX_4><!--xs:string,”00000000000000”--></RegionX_4>
    <RegionX_5><!--xs:string,”00000000000000”--></RegionX_5>
    <RegionX_6><!--xs:string,”00000000000000”--></RegionX_6>
    <RegionX_7><!--xs:string,”00000000000000”--></RegionX_7>
    <RegionX_8><!--xs:string,”00000000000000”--></RegionX_8>
    <RegionX_9><!--xs:string,”00000000000000”--></RegionX_9>
    <RegionX_10><!--xs:string,”00000000000000”--></RegionX_10>
    <RegionX_11><!--xs:string,”00000000000000”--></RegionX_11>
</MotionRegionListV2>
```

9.7.1.3.1 /Pictures/ChannelID/Motion/Trigger

/Pictures/ChannelID/Motion/Trigger	General Resource v1.0
GET	Viewer
Description	Get motion trigger configuration
Query	None
Inbound Data	None
Success Return	**Trigger**
PUT	Viewer
Description	Set motion trigger configuration
Query	None
Inbound Data	**Trigger**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1. <AlarmOutMask>, <SnapshotMask>, <RecordMask> and<PTZActionList> are not applicable.	

Motion Trigger XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Trigger>
    <NotifyAMS>
        <Enable><!--req,xs:boolean--></Enable>
    </NotifyAMS>
    <BeepAlert>
        <Enable><!--req,xs:boolean--></Enable>
    </BeepAlert>
    <FullScreen>
        <Enable><!--req,xs:boolean--></Enable>
    </FullScreen>
    <Mail>
        <Enable><!--req,xs:boolean--></Enable>
    </Mail>
    <AlarmOut>
        <Enable><!--req,xs:boolean--></Enable>
        <AlarmOutMask><!--req,xs:string,” 1111111111111111”--></AlarmOutMask>
    </AlarmOut>
    <Snapshot>
        <Enable><!--req,xs:boolean--></Enable>
        <SnapshotMask><!--req,xs:string,” 1111111111111111”--></SnapshotMask>
    </Snapshot>
    <Record>
        <Enable><!--req,xs:boolean--></Enable>
        <RecordMask><!--req,xs:string,” 1111111111111111”--></RecordMask>
    </Record>
    <PTZ>
        <Enable><!--req,xs:boolean--></Enable>
        <ptzChannelID> <!--req, xs:string --></ptzChannelID>
        <ActionName><!-- req, xs:string,”none,preset, pattern, patrol” --></ActionName>
        <ActionNum><!--req,xs:integer,depend on <ActionName>--></ActionNum>
    </PTZ>
</Trigger>
```

9.7.1.3.2 /Pictures/ChannelID/Motion/Schedule

/Pictures/ChannelID/Motion/Schedule	General Resource v1.0
GET	Viewer
Description	Get motion detection schedule
Query	None
Inbound Data	None
Success Return	**Schedule**
PUT	Viewer
Description	Set motion detection schedule
Query	None
Inbound Data	**Schedule**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1. If set to <AllDay> , <TimeBlock> is not applicable. A day is divided into 48 blocks of 30 minutes each. Block 0 stands for Sunday… Block 6 for Saturday.	

Motion Schedule XML Block

```
<Schedule>
        <AllDay><!--req,xs:boolean--></AllDay>
        <TimeBlock>
                <Block_0><!req,xs:string></Block_0>
                <Block_1><!req,xs:string></Block_1>
                <Block_2><!req,xs:string></Block_2>
                <Block_3><!req,xs:string></Block_3>
                <Block_4><!req,xs:string></Block_4>
                <Block_5><!req,xs:string></Block_5>
                <Block_6><!req,xs:string></Block_6>
        <TimeBlock>
</Schedule>
```

9.7.1.4/Pictures/ChannelID/Mosaic

/Pictures/ChannelID/Mosaic	General Resource v1.0
GET	Viewer
Description	Get privacy mask configuration
Query	None
Inbound Data	None

Success Return	**Mosaic**
PUT	Viewer
Description	Set privacy mask configuration
Query	None
Inbound Data	**Mosaic**
Success Return	**ResponseStatus**
Notes: The camera only supports <ChannelID> 1. 4 masks with ID 1..4 are supported.	

Mosaic XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Mosaic>
    <Enable><!--req,xs:boolean--></Enable>
    <MosaicRegionList>
      <Region><!--opt-->
        <ID><!--req,xs:integer--></ID>
        <TopLeftX><!--req,xs:integer,0-703--></TopLeftX>
        <TopLeftY><!--req,xs:integer,0-575--></TopLeftY>
        <BottomRightX><!--req,xs:integer,0-703--></BottomRightX>
        <BottomRightY><!--req,xs:integer,0-575--></BottomRightY>
     </Region>
   </MosaicRegionList>
</Mosaic>
```

9.8 /TwowayAudio

9.8.1 /TwowayAudio/Open

/TwowayAudio/Open	General Resource v1.0
PUT	Viewer
Description	Open 2-way audio
Query	None
Inbound Data	**None**
Success Return	**ResponseStatus**
Notes:	

```
<TwoWayAudio>
      <VoiceMode><!—req,xs:string,”G726,G711U,G711A” --></VoiceMode>
</TwoWayAudio>
```

9.8.2 /TwowayAudio/Close

/TwowayAudio/Open	General Resource v1.0
PUT	Viewer
Description	Close 2-way audio
Query	None
Inbound Data	**None**
Success Return	**ResponseStatus**
Notes:	

9.8.3 /TwowayAudio/SendData

/TwowayAudio/SendData	General Resource v1.0
PUT	Viewer
Description	Send audio data
Query	None
Inbound Data	**Audio Data**
Success Return	**ResponseStatus**
Notes:	

Example:

```
PUT /TwowayAudio/sendData HTTP/1.1
…
Content-Type: audio/basic
Content-Length: xxx
\r\n
TwowayAudio Data
```

9.8.4 /TwowayAudio/ReceiveData

/TwowayAudio/ReceiveData	General Resource v1.0
GET	Viewer
Description	Receive audio data.
Query	None
Inbound Data	**None**
Success Return	**Audio Data**
Notes:	

Example:

```
GET /TwowayAudio/receiveData HTTP/1.1
…
HTTP/1.1 200 OK
…
Content-Type: audio/basic
Content-Length: xxx
\r\n
TwowayAudio Data…
```

9.9 /EventsNotification

/EventsNotification	General Resource v1.0
Notes:	

9.9.1 /EventsNotification/SubscribedEventTypeList

/EventsNotification/SubscribedEventTypeList	General Resource v1.0
GET	Viewer
Description	Get subscribed event type list.
Query	None
Inbound Data	**None**
Success Return	**SubscribedEventTypeList**
PUT	Operator
Description	Subscribe to event type
Query	None
Inbound Data	EventList
Success Return	**ResponseStatus**
DELETE	Operator
Description	Delete event type subscription
Query	None
Inbound Data	EventList
Success Return	**ResponseStatus**
Notes: Use “PUT” to subscribe to an event type by setting <EventType>	

EventList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<SubscribedEventTypeList>
    <SubscribedEventType>
        <ID><!--req,xs:integer--></ID>
        <EventType><!—req,xs:string，“VMD,videolost,AlarmIn” --></EventType>
        <EventTypeDescription><!—req,xs:string--></EventTypeDescription>
    </SubscribedEventType>
</SubscribedEventTypeList>
```

9.9.2 /EventsNotification/EventList

/EventsNotification/EventList	General Resource v1.0
GET	Viewer
Description	Retrieve events list
Query	None
Inbound Data	**None**
Success Return	EventNotify
Notes: You need to subscribe to at least one event type first	

EventList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<EventList>
        <Event>
              <ID><!--req,xs:string --></ID>
              <IPAddress><!—req,xs:string--></IPAddress>
              <Port><!—req,xs:integer--></Port>
              <Protocol><!—req,xs:string--></Protocol>
              <ChannelID><!—req,xs:string--></ChannelID>
              <DateTime><!—req,xs:datetime--></DateTime>
              <EventType><!—req,xs:string--></EventType>
              <EventDescription><!—req,xs:string--></EventDescription>
        </Event>
</EventList>
```

9.10 /Record

/Record	General Resource v1.0
GET	Viewer
Description	Get recording configuration
Query	None
Inbound Data	**None**
Success Return	Record
PUT	Operator
Description	Set recording configuration
Query	None
Inbound Data	Record
Success Return	**ResponseStatus**
Notes:	

Record XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<Record>
  <RecordWriteCircleType><!—rw,req,xs:string,”fullnotwrite,writecircle”-
  -></RecordWriteCircleType>
    <RecordDeleteType><!—rw,req,xs:string,”24,48,none”--></RecordDeleteType>
    <RecordStreamType><!—rw,req,xs:string,”main,sub”--></RecordStreamType>
    <PreRecord><!—rw,opt,xs:integer,1-120,the unit is second--></PreRecord>
    <DelayRecord><!—rw,opt,xs:integer,1-120,the unit is second--></DelayRecord>
</Record>
```

9.10.1 /Record/HttpEvent/Config

/Record/HttpEvent/Config	General Resource v1.0
GET	Viewer
Description	Get recording status
Query	None
Inbound Data	**None**
Success Return	HttpEventConfig
PUT	Operator
Description	Set recording status
Query	None
Inbound Data	HttpEventConfig
Success Return	**ResponseStatus**
Notes: Enable or disable recording function	

AlarmOut XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<HttpEventConfig>
    <Enable><!--req,xs:boolean--></Enable>
</HttpEventConfig>
```

9.10.2 /Record/HttpEvent/Call

/Record/HttpEvent/Call	General Resource v1.0
PUT	**Viewer**
Description	Call recording event
Query	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes: Starts event recording. The result of combined pre-recording and post event recording will be saved into a file located in /download/video/.	

9.10.3 /Record/Notify/URL

/Record/Notify/URL	**General Resource v1.0**
GET	**Viewer**
Description	**Get notify request settings**
Query	**None**
Inbound Data	**None**
Success Return	**URL**
PUT	**Viewer**
Description	**Set notify request parameters**
Query	**None**
Inbound Data	**URL**
Success Return	**ResponseStatus**
Notes: Request URL example: http://192.168.50.50/eventurl/myevent.php?eventfile=H_20160712192233_00006.avi&camip=192.168.50.51 whereby 192.168.50.50 is the notification target “<Host>” camip is the IP address of the camera eventfile is the video file name in the /download/video/ directory	

/myevent.php is the request path on the target "<Path>"

URL XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<URL>
    <Protocol><!--req,xs:string,"HTTP"--></Protocol>
    <Host><!--req,xs:string,"192.168.1.11"--></Host>
    <Port><!--req,xs:integer,"80"--></Port>
    <Path><!--req,xs:string,"/myevent.php"--></Path>
</URL>
```

9.10.4 /Record/Format/Call

/Record/Format/Call	General Resource v1.0
PUT	Viewer
Description	Format the storage device
Query	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes:	

9.11 /Disk

/Disk	General Resource v1.0
GET	Viewer
Description	Get the disk configuration.
Query	None
Inbound Data	**None**
Success Return	DiskList
PUT	Operator
Description	Set the disk usage configuration.
Query	None
Inbound Data	DiskList
Success Return	**ResponseStatus**
Notes: The PUT method is currently not supported.	

DiskList XML Block

```
<?xml version="1.0" encoding="utf-8"?>
<DiskList>
    <Disk>
        <DiskID><!—ro,req,xs:integer,”0,1,2….”--></DiskID>
        <TotalCapacity><!—ro,req,xs:integer,the unit is MB--></TotalCapacity>
        <AvailableCapacity><!—ro,req,xs:integer,the unit is MB--></AvailableCapacity>
        <DiskStorageType><!—ro,req,xs:string,”SD,samba”--></DiskStorageType>
        <DiskStorageAttribute><!ro,req,xs:string,”idle,writting,reading,diskexception,diskerror,read
        only,readwrite” --></DiskStorageAttribute>
        <PictureCapacityPercent><!—rw,req,xs:integer,0--100--></PictureCapacityPercent>
        <RecordCapacityPercent><!—rw,req,xs:integer,0--100--></RecordCapacityPercent>
        <DiskFormatStatus><!—ro,req,xs:string,”unformatted,formatting,formatted,format
failure”--></DiskFormatStatus>
    </Disk>
</DiskList>
```

9.12 /PTZ

/PTZ	General Resource v1.0

Notes:

Functions to control the focus and the zoom function, if supported by the camera model.

9.12.1 /PTZ/ChannelID/Presets/Goto

/PTZ/ChannelID/Presets/Goto	General Resource v1.0
PUT	Viewer
Description	Initalize autofocus
Query	None
Inbound Data	**Param1=103**
Success Return	**ResponseStatus**

Notes:

1. Param1=103

   Full reinitialization of the autofocus function. Calibration takes 3 to 5 minutes. Use this function only if the camera is already mounted and if the standard settings do not provide the best result.
2. Param1=85

   Override a manual focus setting with the automatic parameter.

Use this function for only for first use or if autofocus is not working well. After sending the command, the camera will process recalibration for 3 to 5 minutes. In this time, do not send other image or PTZ setting commands.

The camera only supports <ChannelID> 1.

Example:

URL：http://192.168.1.46/PTZ/1/Presets/Goto

Data format:

```
PUT /PTZ/1/Presets/Goto HTTP/1.1\r\n
Connection:close\r\n
Host: 192.168.0.46\r\n
Content-Length:\r\n
\r\n
Param1=103
```

9.12.2 /PTZ/ChannelID/FocusFar

/PTZ/ChannelID/FocusFar	General Resource v1.0
PUT	Viewer
Description	Adjust focus far
Query	None
Inbound Data	**See notes**
Success Return	**ResponseStatus**

Notes:

The camera only supports <ChannelID> 1.

1. Use Param1=1 to start and Param1=2 stop focus operation. Usually, 1 will be used for the ButtonPressed event and 2 for the ButtonReleased event. Once the maxium value has been reached, the focus operation will stop. There is not error message.
2. Use Param2 to set the speed. Valid values are from 1 to 10. Default is 1 or 2.

Example:

URL：http://192.168.1.46/PTZ/1/FocusFar

Data format:

```
PUT /PTZ/1/FocusFar HTTP/1.1\r\n
Connection:close\r\n
Host: 192.168.0.46\r\n
Content-Length:\r\n
\r\n
Param1=1&Param2=3
```

9.12.3 /PTZ/ChannelID/FocusNear

/PTZ/ChannelID/FocusNear	General Resource v1.0
PUT	Viewer
Description	Adjust focus near
Query	None
Inbound Data	**See notes**
Success Return	**ResponseStatus**

Notes:

The camera only supports <ChannelID> 1.

1. Use Param1=1 to start and Param1=2 stop focus operation. Usually, 1 will be used for the ButtonPressed event and 2 for the ButtonReleased event. Once the maxium value has been reached, the focus operation will stop. No exception will be thrown..
2. Use Param2 to set the speed. Valid values are from 1 to 10. Default is 1 or 2.

Example:

URL：http://192.168.1.46/PTZ/1/FocusNear

Data format:

```
    PUT /PTZ/1/FocusFar HTTP/1.1\r\n
    Connection:close\r\n
    Host: 192.168.0.46\r\n
    Content-Length:\r\n
    \r\n
Param1=1&Param2=3
```

9.12.4 /PTZ/ChannelID/ZoomIn

/PTZ/ChannelID/ZoomIn	General Resource v1.0
PUT	Viewer
Description	Zoom in
Query	None
Inbound Data	**See notes**
Success Return	**ResponseStatus**

Notes:

The camera only supports <ChannelID> 1.

1. Use Param1=1 to start and Param1=2 stop zoom operation. Usually, 1 will be used for the ButtonPressed event and 2 for the ButtonReleased event. Once the maxium value has been reached, the focus operation will stop. No exception will be thrown.
2. Use Param2 to set the speed. Valid values are from 1 to 10. Default is 1 or 2.

Example:

URL：http://192.168.1.46/PTZ/1/ZoomIn

Data format:

```
PUT /PTZ/1/ZoomIn HTTP/1.1\r\n
Connection:close\r\n
Host: 192.168.0.46\r\n
Content-Length:\r\n
\r\n
Param1=1&Param2=3
```

9.12.5 /PTZ/ChannelID/ZoomOut

/PTZ/ChannelID/ZoomOut	General Resource v1.0
PUT	Viewer
Description	Zoom out
Query	None
Inbound Data	**See notes**
Success Return	**ResponseStatus**

Notes:

The camera only supports <ChannelID> 1.

1. Use Param1=1 to start and Param1=2 stop zoom operation. Usually, 1 will be used for the ButtonPressed event and 2 for the ButtonReleased event. Once the maxium value has been reached, the focus operation will stop. No exception will be thrown.
2. Use Param2 to set the speed. Valid values are from 1 to 10. Default is 1 or 2.

Example:

URL：http://192.168.1.46/PTZ/1/ZoomIn

Data format:

```
PUT /PTZ/1/ZoomIn HTTP/1.1\r\n
Connection:close\r\n
Host: 192.168.0.46\r\n
Content-Length:\r\n
\r\n
Param1=1&Param2=3
```

9.13 /Alarm

/Alarm	General Resource v1.0
Notes: Functions to control the motion detection of the camera.	

9.13.1 /Alarm/AMS/2

/Alarm/AMS/2	General Resource v1.0
PUT	Viewer
Description	Set the server address
Query	None
Inbound Data	**AMS**
Success Return	**ResponseStatus**
Notes: UserName, Password: optional values for HTTP authentication with alarm server ServerURL: Server address coded to base64 **Example:** Text: 192.168.31.1 Base64: MTkyLjE2OC4zMS4x	

AMS Block

```
<?xml version="1.0" encoding="UTF-8"?>
<AMS Version="1.0">
   <Enable><!-- rw,req:string,”true,false” --></Enable>
   <ServerName><!-- rw,req:string,”motionEvent --></ServerName>
   <UserName><!-- rw,opt:string --></UserName>
   <Password><!--rw,opt:string --></Password>
   <ServerURL><!--req,xs:string,”Base 64 encoded string with server IP address”--></ServerURL>
</AMS>
```

9.13.2 /Alarm/EventCfgList/2

/Alarm/EventCfgList/2	General Resource v1.0
PUT	Viewer
Description	Set the response message
Query	None
Inbound Data	**EventCfgList**
Success Return	**ResponseStatus**
Notes: Message: Base64 encoded Server URL	

Alarm interval: event repetition time in case of continuos motion in seconds (default 300)

All other parameters are fixed. Do not change.

Example:

Text: motion/test.asp Base64: bW90aW9uL3Rlc3QuYXNw

EventCfgList Block

```
<?xml version="1.0" encoding="UTF-8"?>
<EventCfgList Version="1.0">
   <motion Version="1.0">
      <PreRecordTime></PreRecordTime>
      <AlarmInterval></AlarmInterval>
      <Message></Message>
   </motion>
   <http Version="1.0">
      <PreRecordTime></PreRecordTime>
      <AlarmInterval><!--rw,req:integer --></AlarmInterval>
      <Message><!--req,xs:string,”Base 64 encoded string with server path”--></Message>
   </http>
</EventCfgList>
```

10 Device discovery V2

In addition to the standard device discovery using UPnP/SSDP there is also a proprietary way to discover and to configure the IP settings of the camera.

This is achieved using muticast messages to transmit device information. The multicast address is 239.239.239.52 at port 5566. Please refer to the following code samples for further reference.

```
//Detect Request
Client:
        <?xml version=”1.0” encoding=”UTF-8”>
        <Request>
              <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”10” />
              <CfgInfo MainCommand=”100” AssistCommand=”-1” DeviceMac=”xx” />
        </Request>

Server:
        <?xml version=”1.0” encoding=”UTF-8”>
        <Response>
              <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”11” ReplyMsg=”xx”>
              <CfgInfo MainCommand=”100” AssistCommand=”-1” DeviceMac=”xx” />
              <DeviceProbeConfig>
                    <DeviceType><!--ro,req,xs:string--></DeviceType>
                    <DeviceVersion><!--ro,req,xs:string--></DeviceVersion>
                    <DeviceChannelCount><!--ro,req,xs:integer--></DeviceChannelCount>
                    <DeviceEnableRegister><!--req,xs:boolean--></DeviceEnableRegister>
                    <DeviceRegisterInterval>
                          <!--req,xs:integer,”second”-->
                    </DeviceRegisterInterval>
                    <DeviceMac><!--ro,req,xs:string--></DeviceMac>
                    <DeviceIP><!--req,xs:string--></DeviceIP>
                    <DeviceMask><!--req,xs:string--><DeviceMask>
                    <DeviceGateway><!--req,xs:string--></DeviceGateway>
                    <DeviceManagerHost><!--req,xs:string--></DeviceManagerHost>
                    <DeviceDetectPort><!--req,xs:integer--></DeviceDetectPort>
                    <ManagerHostPort><!--req,xs:integer--></ManagerHostPort>
                    <CommandPort><!--req,xs:integer--></CommandPort>
                    <MediaPort><!--req,xs:integer--></MediaPort>
                    <HttpPort><!--req,xs:integer--></HttpPort>
                    <RtspPort><!--req,xs:integer--></RtspPort>
                    <TutkUID><!—ro,req,xs:integer--><TutkUID>
```

```
                <UlkUID><!—ro,req,xs:integer--><UlkUID>
            </DeviceProbeConfig>
        </Response>

//Detect Set Device Configuration
Client:
        <?xml version="1.0" encoding="UTF-8">
        <Request>
            <Head Version="1.0.0" ConfigLength="xxx" CommandID="10">
            <CfgInfo MainCommand="101" AssistCommand="-1" DeviceMac="xx" />
            <DeviceProbeConfig>
                <DeviceType><!--ro,req,xs:string--></DeviceType>
                <DeviceVersion><!--ro,req,xs:string--></DeviceVersion>
                <DeviceChannelCount><!--ro,req,xs:integer--></DeviceChannelCount>
                <DeviceEnableRegister><!--req,xs:boolean--></DeviceEnableRegister>
                <DeviceRegisterInterval>
                    <!--req,xs:integer,"second"-->
                </DeviceEnableRegInterval>
                <DeviceMac><!--ro,req,xs:string--></DeviceMac>
                <DeviceIP><!--req,xs:string--></DeviceIP>
                <DeviceNetmask><!--req,xs:string--><DeviceMask>
                <DeviceGateway><!--req,xs:string--></DeviceGateway>
                <DeviceManagerHost><!--req,xs:string--></DeviceManagerHost>
                <DetectPort><!--req,xs:integer--></DeviceDetectPort>
                <ManagerHostPort><!--req,xs:integer--></ManagerHostPort>
                <CommandPort><!--req,xs:integer--></CommandPort>
                <MediaPort><!--req,xs:integer--></MediaPort>
                <HttpPort><!--req,xs:integer--></HttpPort>
                <RtspPort><!--req,xs:integer--></RtspPort>
                <TutkUID><!—ro,req,xs:integer--><TutkUID>
                <UlkUID><!—ro,req,xs:integer--><UlkUID>
            </DeviceProbeConfig>
        </Request >

Server:
        <?xml version="1.0" encoding="UTF-8">
        <Response>
            <Head Version="x.x.x" ConfigLength="xxx" CommandID="11" ReplyMsg="xx" />
            <CfgInfo MainCommand="101" AssistCommand="-1" DeviceMac="xx" />
        </Response>
```

```
//Detect Network Configuration , requires authentication
Client:
    <?xml version=”1.0” encoding=”UTF-8”>
    <Request>
        <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”10” />
        <CfgInfo MainCommand=”102” AssistCommand=”-1” DeviceMac=”xx” />
        <Auth UserName=”xx” UserPassword=”xx” />
    </Request>

Server:
    <?xml version=”1.0” encoding=”UTF-8”>
    <Response>
        <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”11” ReplyMsg=”xx”>
        <CfgInfo MainCommand=”102” AssistCommand=”-1” DeviceMac=”xx” />
        <DeviceProbeNetConfig>
            <DeviceType><!--ro,req,xs:string--></DeviceType>
            <DeviceVersion><!--ro,req,xs:string--></DeviceVersion>
            <DeviceChannelCount><!--ro,req,xs:integer--></DeviceChannelCount>
            <DeviceEnableRegister><!--req,xs:boolean--></DeviceEnableRegister>
            <DeviceEnableRegInterval>
                <!--req,xs:integer,second in unit-->
            </DeviceEnableRegInterval>
            <NetworkInterfaceList>
                <NetworkInterface>
                    <ID><!--req,xs:string--></ID>
                    <Enable><!--ro,req,xs:string,”open,close,notsupport”--></Enable>
                    <DhcpEnable><!--req,xs:boolean--></DhcpEnable>
                    <MAC><!--req,xs:string--></MAC>
                    <IPAddress><!--req,xs:string--></IPAddress>
                    <Netmask><!--req,xs:string--></Netmask>
                    <Gateway><!--req,xs:string--></Gateway>
                    <EnableAutoGenerateIP>
                        <!--req,xs:boolean-->
                    </EnableAutoGenerateIP>
                    <NetworkState>
                      <!--ro,opt,”get ip success,close,getting ip,get ip failed”-->
                    </NetworkState>
                </NetworkInterface>
            </NetworkInterfaceList>
            <DeviceManagerHost><!--req,xs:string--></DeviceManagerHost>
            <DeviceDetectPort><!--req,xs:integer--></DeviceDetectPort>
            <ManagerHostPort><!--req,xs:integer--></ManagerHostPort>
            <CommandPort><!--req,xs:integer--></CommandPort>
            <MediaPort><!--req,xs:integer--></MediaPort>
```

```
                <HttpPort><!--req,xs:integer--></HttpPort>
                <RtspPort><!--req,xs:integer--></RtspPort>
                <TutkUID><!—ro,req,xs:integer--><TutkUID>
                <UlkUID><!—ro,req,xs:integer--><UlkUID>
            </DeviceProbeNetConfig>
        </Response>

//Detect Set Network Configuration , requires authentication
Client:
        <?xml version=”1.0” encoding=”UTF-8”>
        <Request>
             <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”10”>
            <CfgInfo MainCommand=”103” AssistCommand=”-1” DeviceMac=”xx” />
            <Auth UserName=”xx” UserPassword=”xx” />
            <DeviceProbeNetConfig>
                <DeviceType><!--ro,req,xs:string--></DeviceType>
                <DeviceVersion><!--ro,req,xs:string--></DeviceVersion>
                <DeviceChannelCount><!--ro,req,xs:integer--></DeviceChannelCount>
                <DeviceEnableRegister><!--req,xs:boolean--></DeviceEnableRegister>
                <DeviceEnableRegInterval>
                    <!--req,xs:integer,second in unit-->
                </DeviceEnableRegInterval>
                <DeviceMac><!--ro,req,xs:string--></DeviceMac>
                <NetworkInterfaceList>
                    <NetworkInterface>
                        <ID><!--req,xs:string--></ID>
                        <Enable><!--req,xs:string,”open,close,notsupport”--></Enable>
                        <DhcpEnable><!--req,xs:boolean--></DhcpEnable>
                        <MAC><!--req,xs:string--></MAC>
                        <IPAddress><!--req,xs:string--></IPAddress>
                        <Netmask><!--req,xs:string--></Netmask>
                        <Gateway><!--req,xs:string--></Gateway>
                        <EnableAutoGenerateIP>
                            <!--req,xs:boolean-->
                        </EnableAutoGenerateIP>
                        <NetworkState>
                          <!--opt,”get ip success,close,getting ip,get ip failed”-->
                        </NetworkState>
                    </NetworkInterface>
                </NetworkInterfaceList>
                <DeviceManagerHost><!--req,xs:string--></DeviceManagerHost>
                <DeviceDetectPort><!--req,xs:integer--></DeviceDetectPort>
                <ManagerHostPort><!--req,xs:integer--></ManagerHostPort>
                <CommandPort><!--req,xs:integer--></CommandPort>
```

```
                        <MediaPort><!--req,xs:integer--></MediaPort>
                        <HttpPort><!--req,xs:integer--></HttpPort>
                        <RtspPort><!--req,xs:integer--></RtspPort>
                        <TutkUID><!—ro,req,xs:integer--><TutkUID>
                        <UlkUID><!—ro,req,xs:integer--><UlkUID>
                </DeviceProbeNetConfig>
        </Request>

Server:
        <?xml version=”1.0” encoding=”UTF-8” />
        <Response>
                <Head Version=”1.0.0” ConfigLength=”xxx” CommandID=”11” ReplyMsg=”xx” />
                <CfgInfo MainCommand=”103” AssistCommand=”-1” DeviceMac=”xx” />
        </Response>
```

Note:

<Version> is show this interface version, now, we only support “1.0.0”.

<ConfigLenght> is show the struct length that you send.

<CommandID> is the Command ID,if it is the data that the client send ,the <CommandID> is 10, If it is the data that the server send,the <CommandID> is 11.

<Replymsg> is the error code that the server send, if the <ReplyMsg> is 0,then reply success,otherwise, reply failed.

<MainCommand> and <AssistCommand> is show the instruction that you execute. Different request has different instruction.

<DeviceMac> is the Device mac that you want to set.