Camera HTTP API
User Guide

Version 1.0

August 2015

Contents

1 Scope 1
2 Camera HTTP API 2
2.1 HTTP Methods 2
2.2 XML 2
2.3 HTTP Header 2
3 Services and General Resources 4
3.1 System 4
3.1.1 Device Information 4
3.1.2 Factory Default 5
3.1.3 Auto Reboot 6
3.1.4 NTP 7
3.1.5 Reboot 8
3.1.6 Shutdown 8
3.1.7 Configuration File 9
3.1.8 System General 9
3.1.9 DST 11
3.1.10 UID 12
3.1.11 Log 13
3.1.12 Version Information 13
3.1.13 Performance Information 14
3.1.14 Password On Boot 15
3.1.15 Firmware Upgrade 15
3.2 Network 16
3.2.1 Local Link 16
3.2.2 Net Port 17
3.2.3 DDNS 19
3.2.4 3G 20
3.2.5 Net Status 20
3.2.6 Upnp 21
3.2.7 Link Type 22
3.2.8 PPPOE 22
3.2.9 Ftp 23
3.2.10 Email 24
3.3 Video 25
3.3.1 Input Configuration 26
3.3.2 Input OsdChannelName 26

3.3.3 Input OsdDatetime 27
3.3.4 Input Shelter 28
3.4 Streaming 30
3.4.1 Compression 30
3.4.2 Rtsp Url 32
3.5 Security 33
3.5.1 Users 33
3.5.2 Online Users 34
3.6 MD 35
3.6.1 MD Configuration 35
3.7 PTZ 38
3.7.1 Serial 38
3.7.2 Preset 39
3.7.3 Patrol 40
3.7.5 Control 42
3.8 IO 44
3.8.1 Input 45
3.8.2 Output 46
3.8.3 Output Status 47
3.8.4 Output Trigger 48
3.9 Image 49
3.9.1 Power Line Frequency 49
3.9.2 Exposure 50
3.9.3 Shutter 50
3.9.4 Gain 51
3.9.5 Scene 52
3.9.6 IR 53
3.9.7 BLC 54
3.9.8 Mirror 55
3.9.9 Flip 56
3.10 Event 57
3.10.1 Start Alarm 57
3.10.2 Stop Alarm 57
3.10.3 Alarm Event 58
3.11 Record 58
3.11.1 Recording Schedule 58
3.12 Two Way Audio 60
3.12.1 Open 60

3.12.2 Close..... 61
3.12.3 Send Data..... 61
Notes 62

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 the Media Streaming, please refer to "RTSP protocol".

2 Camera HTTP API

2.1 HTTP Methods

GET and POST methods are supported in this specification.

GET /cgi-bin/config.cgi?name=<cmdName>&parm1=<val1>& parm2=<val2> HTTP/1.1

POST /cgi-bin/config.cgi?name=<cmdName>&parm1=<val1>& parm2=<val2> HTTP/1.1

name: The name field refer to the parameter name for the HTTP Command to GET or POST. (example: name=/Video/input/shelter)

parm: The parm field refer to the query in the command. (example: channel=0)

2.2 XML

A device must support the syntax defined by W3C XML 1.0 specification and UTF-8 character set. All XML files must adopt UTF-8 encoding according to RFC3629.
Lists contained in XML blocks will be represented in the format of <XXXList>, and each <XXXList> tag may contain zero, one or more nodes.

2.3 HTTP Header

The HTTP Message should include the custom Seq header and the Authorization header. The encryption use base64 and the source token should be “user:password”.

Example:

Authorization: Basic YWRtaW46MTExMTEx

Seq: 20

HTTP Message Example:

GET /cgi-bin/config.cgi?name=/Video/input/osdDatetime&channel=0 HTTP/1.1

Authorization: Basic YWRtaW46MTExMTEx

Seq: 20

User-Agent: session

Host: 192.168.2.178

Cache-Control: no-cache

HTTP/1.1 200 OK

Server: GoAhead-http

Date: Sat Aug 29 10:48:53 2015

Transfer-Encoding: chunked

Connection: keep-alive

Pragma: no-cache

Cache-Control: no-cache

Seq: 20

Content-Type: text/xml;charset=gb2312

```
<?xml version="1.0" encoding="UTF-8" ?>
<body>
	<OsdDatetime version="1.1">
		<channelId>0</channelId>
		<enable>1</enable>
		<topLeftX>1</topLeftX>
		<topLeftY>0</topLeftY>
	</OsdDatetime>
</body>
```

3 Services and General Resources

3.1 System

/System
Notes:

3.1.1 Device Information

name=/System/deviceInfo V1.1	
GET	**Viewer**
Description	It is used to get device information.
Parameter	None
Inbound Data	None
Success Return	**DeviceInfo**
Notes: All fields are read-only.	

DeviceInfo XML Block
<DeviceInfo version="1.1"> <firmVersion> <!-- ro, xs:string --> </firmVersion> <IOInputPortNum> <!-- ro, xs:integer, “” --> </IOInputPortNum> <IOOutputPortNum> <!-- ro, xs:integer, “” --> </IOOutputPortNum> <diskNum> <!-- ro, xs: integer, “” --> </diskNum> <type> <!-- ro, xs:string, “dvr, ipc” --> </type> <channelNum> <!-- ro, xs:integer, “” --> </channelNum> <audio> <!-- ro, xs: integer, “0:no audio; 1:have audio” --> </audio>

```
    < resolution>
        <width>  <!-- ro, xs: integer -->  </width>
        <high>  <!-- ro, xs: integer -->  </high>
    </resolution>
    <language>  <!-- ro, xs:integer, “see the /System/SystemGeneral's Notes” -->
</language>
   <sdCard>  <!-- ro, xs: integer, “0:disable SdCard;1:enable SdCard” --> </sdCard>
    <PTZ>  <!-- ro, xs:integer, “0:disable ptz;1:enable ptz” -->  </PTZ>
    <typeInfo>  <!-- ro, xs:string, “eg: H18C01-B” -->  </typeInfo>
    <softVer>  <!-- ro, xs:integer, “soft version” -->  </softVer>
    <hardVer>  <!-- ro, xs:integer, “hardware version” -->  </hardVer>
    <panelVer>  <!-- ro, xs:integer, “panel version” -->  </panelVer>
    <norm>  <!-- ro, xs:string, “PAL, NTSC” -->  </Norm>
    <osdFormat>  <!-- ro, xs:string, “OSD Time format: YMD, MDY, DMY, YMD_CN,
MDY_CN, DMY_CN” -->  </osdFormat>
    <B485>  <!-- ro, xs:integer, “0:disable 485;1:enable 485” -->  </B485>
</DeviceInfo>
```

3.1.2 Factory Default

/System/factoryDefault	**V1.1**
POST	
Description	It is used to reset some configurations selected for the device to the factory default.
Parameter	None
Inbound Data	**Restore**
Success Return	**ResponseStatus**
Notes: Different modes are provided:	

“all cfgs” “display cfg” “display and network cfg” …
Select what you want to restore.
After apply, the device should be rebooted.

Restore XML Block
<Restore version="1.1">
<all> <!-- req, xs:integer, “0:no;1:restore all cfgs” --> </all>
<display> <!-- req, xs:integer, “0:no;1:restore display cfg” --> </display>
<recording> <!-- req, xs:integer, “0:no;1:restore recording cfg” -->
</recording>
<network> <!-- req, xs:integer, “0:no;1:restore network cfg” --> </network>
<alarm> <!-- req, xs:integer, “0:no;1:restore alarm cfg” --> </alarm>
<device> <!-- req, xs:integer, “0:no;1:restore device cfg” --> </device>
<system> <!-- req, xs:integer, “0:no;1:restore system cfg” --> </system>
<wifi> <!-- req, xs:integer, “0:no;1:restore wifi cfg” --> </wifi>
<IPC> <!-- req, xs:integer, “only used for DVR,0:no;1:restore all IPC” -->
</IPC>
</Restore>

3.1.3 Auto Reboot

/System/autoReboot	**V1.1**
GET	
Description	It is used to get configuration of AutoReboot.
Parameter	None
Inbound Data	None
Success Return	**AutoReboot**
POST	
Description	It is used to set configuration of AutoReboot.

Parameter	None
Inbound Data	**AutoReboot**
Success Return	**ResponseStatus**
Notes:	

AutoReboot XML Block
<AutoReboot version="1.1"> <enable> <!-- req, xs:integer, “0:disable auto reboot;1:enable auto reboot” --> </enable> <weekDay> <!-- req, xs:string, “Everyday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday” --> </weekDay> <hour> <!-- req, xs:integer, “0~23” --> </hour> <minute> <!-- req, xs:integer, “0~59” --> </minute> <second> <!-- req, xs:integer, “0~59” --> </second> </AutoReboot>

3.1.4 NTP

/System/ntp	**V1.1**
GET	
Description	It is used to get configuration of NTP.
Parameter	None
Inbound Data	None
Success Return	**Ntp**
POST	
Description	It is used to set configuration of NTP.
Parameter	None
Inbound Data	**Ntp**

Success Return	ResponseStatus
Notes: This POST command will restart some services, next command should be 2 senconds later.	

Ntp XML Block
<Ntp version="1.1"> <enable> <!-- req, xs:integer, “0:disable NTP;1:enable NTP” --> </enable> <server> <!-- req, xs:string, “NTP server address” --> </server> <synchronizeInterval> <!-- req, xs:integer, “NTP synchronize time every synchronizeInterval minutes.” --> </ synchronizeInterval> <port> <!-- req, xs:integer, “default:123” --> </port> </Ntp>

3.1.5 Reboot

/System/reboot	V1.1
POST	
Description	It is used to reboot the device.
Parameter	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes: ResponseStatus is returned before the device proceeds to reboot.	

3.1.6 Shutdown

/System/shutdown	V1.1

POST	
Description	It is used to shutdown the device.
Parameter	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes: ResponseStatus is returned before shutdown system. This operation will be invalid if the device do not support shutdown.	

3.1.7 Configuration File

/System/configurationFile	V1.1
GET	
Description	It is used to get device's configuration files.
Parameter	None
Inbound Data	None
Success Return	**Opaque Data**
POST	
Description	It is used to update device's configuration files.
Parameter	None
Inbound Data	**Opaque Data**
Success Return	**ResponseStatus**
Notes: Configuration file is device-dependant. It may be binary or any other format. The device will be reboot after configuration file is applied.	

3.1.8 System General

/System/systemGeneral	V1.1
GET	
Description	It is used to get configuration of System General.
Parameter	None
Inbound Data	None
Success Return	**SystemGeneral/Norm**
POST	
Description	It is used to set configuration of System General.
Parameter	None
Inbound Data	**SystemGeneral/Norm**
Success Return	**ResponseStatus**

Notes:

Language list:

Chinese, English, Japanese, Russian, Italian, Czech, Romania, Hungary, France, Spanish, Portugal, Korean, BigCHN, Polish, Bulgarian

If the device is IPC, OSD's language is Chinese or English.

This POST command will restart some services, next command should be 2 senconds later.

timeZone example: UTC+08:00：-28800 = (+8)*(-3600)

SystemGeneral XML Block

```
<SystemGeneral version="1.1">
    <timeZone> <!-- req, xs:integer, “seconds offset of utc” --> </timeZone>
    <osdFormat> <!-- req, xs:string, “MDY, YMD, DMY, MDY_CN, YMD_CN, DMY_CN” --> </osdFormat>
    <year> <!-- req, xs:integer, “” --> </year>
    <month> <!-- req, xs:integer, “1~12” --> </month>
    <day> <!-- req, xs:integer, “1~31” --> </day>
    <hour> <!-- req, xs:integer, “0~23” --> </hour>
```

```
    <minute>    <!-- req, xs:integer, “0~59” -->    </minute>
    <second>    <!-- req, xs:integer, “0~59” -->    </second>
    <deviceId>    <!-- ro, xs:integer, “default:000” -->    </deviceId>
    <language>    <!-- req, xs:string, “see the notes above” -->    </language>
</SystemGeneral>
```

Norm XML Block
<Norm version="1.1">   <norm>   <!-- req, xs:string, "PAL, NTSC” -->   </norm> </Norm>

3.1.9 DST

/System/dst	V1.1
GET	
Description	It is used to get configuration of DST.
Parameter	None
Inbound Data	None
Success Return	**Dst**
POST	
Description	It is used to set configuration of DST.
Parameter	None
Inbound Data	**Dst**
Success Return	**ResponseStatus**
Notes:	

Dst XML Block
<Dst version="1.1">   <enable>   <!-- req, xs:integer, “0:disable DST;1:enable DST” -->   </enable>

```
    <offset>  <!-- req, xs:integer, “0~1, 0:1 hour;1:2 hours” -->  </offset>
    <startMonth>  <!-- req, xs:integer, “1~12” -->  </startMonth>
    <startWeekIndex>  <!-- req, xs:integer, “1~5, weekday index of month” -->
</startWeekIndex>
    <startWeekday>  <!-- req, xs:string, “Sunday, Monday, ...” -->
</startWeekday>
    <startHour>  <!-- req, xs:integer, “0~23” -->  </startHour>
    <startMinute>  <!-- req, xs:integer, “0~59” -->  </startMinute>
    <startSecond>  <!-- req, xs:integer, “0~59” -->  </startSecond>
    <endMonth>  <!-- req, xs:integer, “1~12” -->  </endMonth>
    <endWeekIndex>  <!-- req, xs:integer, “1~5, weekday index of month” -->
</endWeekIndex>
    <endWeekday>  <!-- req, xs:string, “Sunday, Monday, ...” -->  </endWeekday>
    <endHour>  <!-- req, xs:integer, “0~23” -->  </endHour>
    <endMinute>  <!-- req, xs:integer, “0~59” -->  </endMinute>
    <endSecond>  <!-- req, xs:integer, “0~59” -->  </endSecond>
</Dst>
```

3.1.10 UID

/System/uid	**V1.1**
GET	**Viewer**
Description	It is used to get configuration of UID.
Parameter	None
Inbound Data	None
Success Return	**Uid**
Notes:	

Uid XML Block
<Uid version="1.1">   <uid> <!-- ro, xs:string, “” --> </uid> </Uid>

3.1.11 Log

/System/log	V1.1
GET	**Viewer**
Description	It is used to get system log file.
Parameter	None
Inbound Data	None
Success Return	**Opaque Data**
Notes:	

3.1.12 Version Information

/System/versionInfo	V1.1
GET	**Viewer**
Description	It is used to get version information.
Parameter	None
Inbound Data	None
Success Return	**VersionInfo**
Notes:	

VersionInfo XML Block

```
<VersionInfo version="1.1">
    <name>  <!-- ro, xs:string, “” -->  </name>
    <type>  <!-- ro, xs:string, “like: H18C01-B” -->  </type>
    <serialNumber>  <!-- ro, xs:string, “” -->  </serialNumber>
    <buildDay>  <!-- ro, xs:string, “” -->  </buildDay>
    <hardwareVersion>  <!-- ro, xs:string, “” -->  </hardwareVersion>
    <cfgVersion>  <!-- ro, xs:string, “” -->  </cfgVersion>
    <firmwareVersion>  <!-- ro, xs:string, “” -->  </firmwareVersion>
    <detail>  <!-- ro, xs:string, “” -->  </detail>
</VersionInfo>
```

3.1.13 Performance Information

/System/performanceInfo	V1.1
GET	**Viewer**
Description	It is used to get information of system current performance.
Parameter	None
Inbound Data	None
Success Return	**PerformanceInfo**
Notes:	

PerformanceInfo XML Block

```
<PerformanceInfo version="1.1">
    <cpuUseRate>  <!-- ro, xs:integer, “use %” -->  </ cpuUseRate >
    <codeRate>  <!-- ro, xs:integer, “” -->  </codeRate>
    <netDataRate>  <!-- ro, xs:integer, “internet access data rate,use Kb/s” -->
</netDataRate>
</PerformanceInfo>
```

3.1.14 Password On Boot

/System/PasswordOnBoot	V1.1
GET	
Description	It is used to get state if password on boot or not.
Parameter	None
Inbound Data	None
Success Return	**PasswordOnBoot**
POST	
Description	It is used to set state if password on boot or not.
Parameter	None
Inbound Data	**PasswordOnBoot**
Success Return	**ResponseStatus**
Notes:	

PasswordOnBoot XML Block
<PasswordOnBoot version="1.1"> <enable> <!-- req, xs:integer, “0:disable password on boot; 1:enable password on boot” --> </enable> </PasswordOnBoot>

3.1.15 Firmware Upgrade

/System/firmwareUpgrade	V1.1
POST	
Description	It is used to upgrade the firmware of the device.

Parameter	**upgradeFileName=%s&updateParameter=%d**
Inbound Data	**Opaque Data**
Success Return	**ResponseStatus**
Notes: You should input the Parameter string which contain upgrades file name and update parameter flag. updateParameter should be 0 or 1. The device should be rebooted after the upgrade is completed.	

3.2 Network

/Network
Notes:

3.2.1 Local Link

/Network/localLink	**V1.1**
GET	
Description	It is used to get local link information.
Parameter	None
Inbound Data	None
Success Return	**Dhcp/AutoDns/Ip/Dns**
POST	
Description	It is used to set local link information.
Parameter	None
Inbound Data	**Dhcp/AutoDns/Ip/Dns**
Success Return	**ResponseStatus**

Notes:

Some fields may be invalid. Such as in POST, if dhcp is enable, the Ip will be ignored.

Dhcp XML Block

```
<Dhcp version="1.1">
    <enable>  <!-- req, xs:integer, “0:disable dhcp;1:enable dhcp” -->  </enable>
</Dhcp>
```

AutoDns XML Block

```
<AutoDns version="1.1">
    <enable>  <!-- req, xs:integer, “0:disable autodns;1:enable autodns” -->
</enable>
</AutoDns>
```

Ip XML Block

```
<Ip version="1.1">
    <ip>  <!-- req, xs:string, “” -->  </ip>
    <mask>  <!-- req, xs:string, “” -->  </mask>
    <mac>  <!-- req, xs:string, “eg: EC:71:DB:DC:5D:93” -->  </mac>
    <gateway>  <!-- req, xs:string, “” -->  </gateway>
</Ip>
```

Dns XML Block

```
<Dns version="1.1">
    <dns1>  <!-- req, xs:string, “” -->  </dns1>
    <dns2>  <!-- req, xs:string, “” -->  </dns2>
</Dns>
```

3.2.2 Net Port

/Network/netPort	V1.1
GET	
Description	It is used to get net ports.
Parameter	None
Inbound Data	None
Success Return	**ServerPort/HttpPort/RtspPort/OnvifPort**
POST	
Description	It is used to set net ports.
Parameter	None
Inbound Data	**ServerPort/HttpPort/RtspPort/OnvifPort**
Success Return	**ResponseStatus**
Notes: The services should be restarted after the configurations are applied.	

ServerPort XML Block

```
<ServerPort version="1.1">
    <serverPort>  <!-- req, xs:integer, “default:9000” -->  </serverPort>
</ServerPort>
```

HttpPort XML Block

```
<HttpPort version="1.1">
    <httpPort>  <!-- req, xs:integer, “default:80” -->  </httpPort>
</HttpPort>
```

RtspPort XML Block

```
<RtspPort version="1.1">
    <rtspPort>  <!-- req, xs:integer, “default:554” -->  </rtspPort>
</RtspPort>
```

OnvifPort XML Block
<OnvifPort version="1.1">   <onvifPort> <!-- req, xs:integer, “default:8000” --> </onvifPort> </OnvifPort>

3.2.3 DDNS

/Network/ddns	V1.1
GET	
Description	It is used to get configuration of DDNS.
Parameter	None
Inbound Data	None
Success Return	**Ddns**
POST	
Description	It is used to set configuration of DDNS.
Parameter	None
Inbound Data	**Ddns**
Success Return	**ResponseStatus**
Notes: This POST command will restart some services, next command should be 2 senconds later.	

Ddns XML Block
<Ddns version="1.1">   <enable> <!-- req, xs:integer, “0: disable ddns;1:enable ddns” --> </enable>   <ddnsName> <!-- req, xs:string, “” --> </ddnsName>   <userName> <!-- req, xs:string, “” --> </userName>   <password> <!-- req, xs:string, “” --> </password> </Ddns>

3.2.4 3G

/Network/3g	V1.1
GET	
Description	It is used to get configuration of 3G.
Parameter	None
Inbound Data	None
Success Return	**Dhcp/AutoDns/Ip/Dns/Net3g**
POST	
Description	It is used to set configuration of 3G.
Parameter	None
Inbound Data	**Dhcp/AutoDns/Ip/Dns/ Net3g**
Success Return	**ResponseStatus**
Notes: Dhcp/AutoDns/Ip/Dns are shown in /Network/localLink above. Some fields may be invalid. Such as in POST, if dhcp is enable, the Ip will be ignored.	

Net3g XML Block

```
<Net3g version="1.1">
    <autoApn>  <!-- req, xs:integer, “0:disable auto apn;1:enable auto apn” -->
</autoApn>
    <apn>  <!-- req, xs:string, “” -->  </apn>
    <phone>  <!-- req, xs:string, “dial phone number” -->  </phone>
    <userName>  <!-- req, xs:string, “” -->  </userName>
    <password>  <!-- req, xs:string, “” -->  </password>
</Net3g>
```

3.2.5 Net Status

/Network/netStatus	V1.1
GET	**Viewer**
Description	It is used to get net status.
Parameter	None
Inbound Data	None
Success Return	**Ip/Dns**
Notes: Ip/Dns are shown in /Network/localLink above. But all the configurations are read only.	

3.2.6 Upnp

/Network/upnp	V1.1
GET	
Description	It is used to get configuration of upnp.
Parameter	None
Inbound Data	None
Success Return	**Upnp**
POST	
Description	It is used to set configuration of upnp.
Parameter	None
Inbound Data	**Upnp**
Success Return	**ResponseStatus**
Notes: This POST command will restart some services, next command should be 2 senconds later.	

Upnp XML Block
<Upnp version="1.1">   <enable> <!-- req, xs:integer, “0:disable upnp;1:enable upnp” --> </enable> </Upnp>

3.2.7 Link Type

/Network/linkType	V1.1
GET	
Description	It is used to get link type.
Parameter	None
Inbound Data	None
Success Return	**LinkType**
POST	
Description	It is used to set link type.
Parameter	None
Inbound Data	**LinkType**
Success Return	**ResponseStatus**
Notes:	

LinkType XML Block
<LinkType version="1.1">   <type> <!-- req, xs:string, “LAN, PPPOE, sCDMA” --> </type> </LinkType>

3.2.8 PPPOE

/Network/pppoe	V1.1
GET	
Description	It is used to get configuration of PPPOE.
Parameter	None
Inbound Data	None
Success Return	**Pppoe**
POST	
Description	It is used to set configuration of PPPOE.
Parameter	None
Inbound Data	**Pppoe**
Success Return	**ResponseStatus**
Notes:	

Pppoe XML Block
<Pppoe version="1.1"> <userName> <!-- req, xs:string, “” --> </userName> <password> <!-- req, xs:string, “” --> </password> </Pppoe>

3.2.9 Ftp

/Network/ftp	V1.1
GET	
Description	It is used to get configuration of Ftp.
Parameter	None
Inbound Data	None

Success Return	**Ftp**
POST	
Description	It is used to set configuration of Ftp.
Parameter	None
Inbound Data	**Ftp**
Success Return	**ResponseStatus**
Notes:	

Ftp XML Block
<Ftp version="1.1"> <server> <!-- req, xs:string, “” --> </server> <port> <!-- req, xs:integer, “default:21” --> </port> <nonymous> <!-- req, xs:integer, “0:disable nonymous;1:enable nonymous” --> </nonymous> <remoteDir> <!-- req, xs:string, “remote file dir” --> </remoteDir> <userName> <!-- req, xs:string, “” --> </userName> <password> <!-- req, xs:string, “” --> </password> </Ftp>

3.2.10 Email

/Network/email	**V1.1**
GET	
Description	It is used to get configuration of Email.
Parameter	None
Inbound Data	None
Success Return	**Email**
POST	

Description	It is used to set configuration of Email.
Parameter	None
Inbound Data	**Email**
Success Return	**ResponseStatus**
Notes: This POST command will restart some services, next command should be 2 senconds later.	

Email XML Block
<Email version="1.1"> <smtpServer> <!-- req, xs:string, “SMTP server name” --> </smtpServer> <userName> <!-- req, xs:string, “” --> </userName> <password> <!-- req, xs:string, “” --> </password> <address1> <!-- req, xs:string, “receivers mail address” --> </address1> <address2> <!-- req, xs:string, “receivers mail address” --> </address2> <address3> <!-- req, xs:string, “receivers mail address” --> </address3> <smtpPort> <!-- req, xs:integer, “smtp port” --> </smtpPort> <attachment> <!-- req, xs:integer, “0:disable attachment;1:enable attachment” --> </attachment> <ssl> <!-- req, xs:integer, “0:disable ssl;1:enable ssl” --> </ssl> <interval> <!-- req, xs:integer, “send another mail should be at least interval seconds later” --> </interval> </Email>

3.3 Video

/Video

Notes:

3.3.1 Input Configuration

/Video/input	V1.1
GET	
Description	It is used to get configuration of video input which channel is ID.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**VideoInput**
POST	
Description	It is used to set configuration of video input which channel is ID.
Parameter	channel=<ID>
Inbound Data	**VideoInput**
Success Return	**ResponseStatus**
Notes:	

VideoInput XML Block

```
<VideoInput version="1.1">
    <channelId>  <!-- ro, xs:integer, “the channel id” -->  </channelId>
    <bright>  <!-- req, xs:integer, “0~255,default:128” -->  </bright>
    <contrast>  <!-- req, xs:integer, “0~255,default:128” -->  </contrast>
    <saturation>  <!-- req, xs:integer, “0~255,default:128” -->  </saturation>
    <hue>  <!-- req, xs:integer, “0~255,default:128” -->  </hue>
    <sharpen>  <!-- req, xs:integer, “0~255,default:128” -->  </sharpen>
</VideoInput>
```

3.3.2 Input osdChannelName

/Video/input/osdChannelName	V1.1
GET	
Description	It is used to get a particular channel name configuration for a video input channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**OsdChannelName**
POST	
Description	It is used to set a particular channel name configuration for a video input channel.
Parameter	channel=<ID>
Inbound Data	**OsdChannelName**
Success Return	**ResponseStatus**
Notes:	

OsdChannelName XML Block

```
<OsdChannelName version="1.1">
    <channelId>  <!-- ro, xs:integer, “the channel id” -->  </channelId>
    <enable>  <!-- req, xs:integer, “0:disable;1:enable” -->  </enable>
    <topLeftX>  <!-- req, xs:integer, “0:left; 1: middle; 2:right” -->  </topLeftX>
    <topLeftY>  <!-- req, xs:integer, “0:top; 1: middle; 2: bottom” -->  </topLeftY>
</OsdChannelName>
```

3.3.3 Input OsdDatetime

/Video/input/osdDatetime	V1.1

GET	
Description	It is used to get a particular osd datetime configuration for a video input channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**OsdDatetime**
POST	
Description	It is used to set a particular osd datetime configuration for a video input channel.
Parameter	channel=<ID>
Inbound Data	**OsdDatetime**
Success Return	**ResponseStatus**
Notes:	

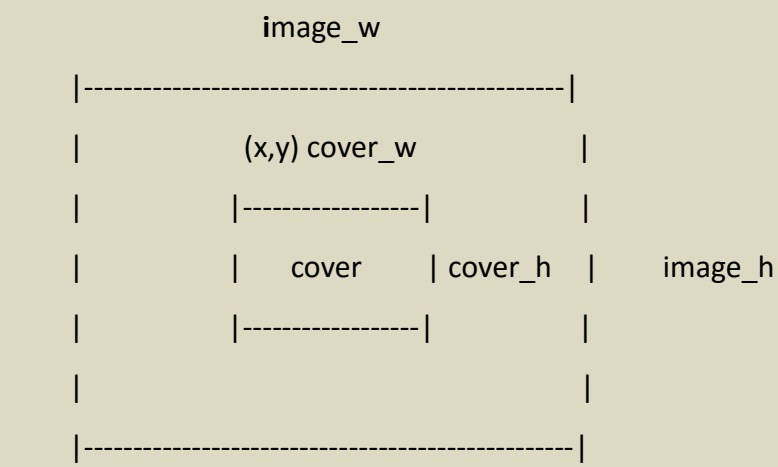
OsdDatetime XML Block
<OsdDateTime version="1.1"> <channelId> <!-- ro, xs:integer, “the channel id” --> </channelId> <enable> <!-- req, xs:integer, “0:disable;1:enable” --> </enable> <topLeftX> <!-- req, xs:integer, “0:left; 1: middle; 2:right” --> </topLeftX> <topLeftY> <!-- req, xs:integer, “0:top; 1: middle; 2: bottom” --> </topLeftY> </OsdDateTime>

3.3.4 Input Shelter

/Video/input/shelter V1.1	
GET	
Description	It is used to get configuration of one channel's shelter.
Parameter	channel=<ID>

Inbound Data	None
Success Return	**Shelter**
POST	
Description	It is used to set configuration of one channel's shelter.
Parameter	channel=<ID>
Inbound Data	**Shelter**
Success Return	**ResponseStatus**

Notes:



x: cover left_top coordinate x

y: cover left_top coordinate y

topLeftX = ((x<<16)&0xffff0000)|(image_w&0xffff);

topLeftY = ((y<<16)&0xffff0000)|(image_h&0xffff);

width = ((cover_w<<16)&0xffff0000)|(image_w&0xffff);

height = ((cover_h<<16)&0xffff0000)|(image_h&0xffff);

Shelter XML Block

```
<Shelter version="1.1">
    <channelId>    <!-- ro, xs:integer, “the channel id” -->    </channelId>
    <enable>    <!-- req, xs:integer, “0:disable;1:enable” -->    </enable>
    <shelterList>
```

```
        <shelter>
            <id>    <!-- req, xs:integer, “0~3” -->    </id>
            <enable>    <!-- req, xs:integer, “0:disable;1:enable” -->    </enable>
            <topLeftX>    <!-- req, xs:integer, “see the notes” -->    </topLeftX>
            <topLeftY>    <!-- req, xs:integer, “see the notes” -->    </topLeftY>
            <width>    <!-- req, xs:integer, “see the notes” -->    </width>
            <height>    <!-- req, xs:integer, “see the notes” -->    </height>
        </shelter>    <!-- opt -->
        ...
    </shelterList>
</Shelter>
```

3.4 Streaming

/Streaming
Notes:

3.4.1 Compression

/Streaming/compression	V1.1
GET	
Description	It is used to get configuration of compression.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Compression**

POST	
Description	It is used to set configuration of compression.
Parameter	channel=<ID>
Inbound Data	**Compression**
Success Return	**ResponseStatus**
Notes:	

Compression XML Block

```
<Compression version="1.1">
    <channelId>  <!-- ro, xs:integer, “” -->  </channelId>
    <mainStream>
        <audio>  <!-- req, xs:integer, “0:disable audio;1:enable audio” -->
</audio>
        <width>  <!-- req, xs:integer, “resolution” -->  </width>
        <height>  <!-- req, xs:integer, “resolution” -->  </height>
        <encoderType>  <!-- req, xs:string, “vbr, cbr” -->  </encoderType>
        <frame>  <!-- req, xs:integer, “” -->  </frame>
        <bitRate>  <!-- req, xs:integer, “” -->  </bitRate>
    </mainStream>
    <subStream>
        <audio>  <!-- req, xs:integer, “0:disable audio;1:enable audio” -->
</audio>
        <width>  <!-- req, xs:integer, “resolution” -->  </width>
        <height>  <!-- req, xs:integer, “resolution” -->  </height>
        <encoderType>  <!-- req, xs:string, “vbr, cbr” -->  </encoderType>
        <frame>  <!-- req, xs:integer, “” -->  </frame>
        <bitRate>  <!-- req, xs:integer, “” -->  </bitRate>
    </subStream>
    <thirdStream>
```

```
        <audio>    <!-- req, xs:integer, “0:disable audio;1:enable audio” -->
</audio>
        <width>    <!-- req, xs:integer, “resolution” -->    </width>
        <height>    <!-- req, xs:integer, “resolution” -->    </height>
        <encoderType>    <!-- req, xs:string, “vbr, cbr” -->    </encoderType>
        <frame>    <!-- req, xs:integer, “” -->    </frame>
        <bitRate>    <!-- req, xs:integer, “” -->    </bitRate>
    </thridStream>
</Compression>
```

3.4.2 Rtsp Url

/Streaming/RtspUrl	V1.1
GET	**Viewer**
Description	It is used to get RTSP Url of a specified channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**RtspUrl**
Notes: url name like: rtsp://192.168.1.11:554/h264Preview_01_main	

RtspUrl XML Block

```
<RtspUrl version="1.1">
    <channelId>    <!-- ro, xs:integer, “” -->    </channelId>
    <UrlList>
        <Url>
            <streamType>    <!-- ro, xs:string, “mainStream, subStream” -->
</streamType>
```

```
            <urlToken>    <!-- ro, xs:string, “” -->    </urlToken>
        </Url>
        ...
    </UrlList>
</RtspUrl>
```

3.5 Security

/Security
Notes:

3.5.1 Users

/Security/users	V1.1
GET	
Description	It is used to get users' information.
Parameter	None
Inbound Data	None
Success Return	**UserList**
POST	
Description	It is used to set users' information.
Parameter	None
Inbound Data	**UserList**
Success Return	**ResponseStatus**
Notes:	

UserList XML Block
<UserList version="1.1">   <User>   <userId> <!-- ro, xs:integer, “account id” --> </userId>   <userName> <!-- req, xs:string, “” --> </userName>   <password> <!-- req, xs:string, “” --> </password>   <userLevel> <!-- req, xs:integer, “0:normal user;1:administrator” --> </userLevel>   </User> <!-- opt-->   ... </UserList>

3.5.2 Online Users

/Security/onlineUsers	V1.1
GET	
Description	It is used to get online users' information.
Parameter	None
Inbound Data	None
Success Return	**OnlineUserList**
Notes:	

OnlineUserList XML Block
<OnlineUserList version="1.1">   <OnlineUser>   <userId> <!-- ro, xs:integer, “account id” --> </userId>   <sessionId> <!-- ro, xs:integer, “online user id” --> </sessionId>

```
        <userName>    <!-- ro, xs:string, “” -->    </userName>
        <password>    <!-- ro, xs:string, “” -->    </password>
        <userLevel>    <!-- ro, xs:integer, “0:normal user;1:administrator” -->
</userLevel>
        <ipAddress>    <!-- ro, xs:string, “” -->    </ipAddress>
        <macAddress>    <!-- ro, xs:string, “” -->    </macAddress>
    </OnlineUser>    <!-- opt-->
    ...
</OnlineUserList>
```

3.6 MD

/MD
Notes: Motion Detection

3.6.1 MD Configuration

/MD	V1.1
GET	
Description	It is used to get configurations of motion detection.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**MD**
POST	
Description	It is used to set configurations of motion detection.
Parameter	channel=<ID>

Inbound Data	MD
Success Return	ResponseStatus
Notes:	

MD XML Block

```
<MD version="1.1">
      <channelId>    <!-- ro, xs:integer, “” -->    </channelId>
      <enable>    <!-- req, xs:integer, “0:disable motion detection;1:enable motion detection” -->    </enable>
      <scope>    <!-- opt, “motion detection area , encode binary array data to base64 string”    -->
            <columns>    <!-- req, xs:integer, “96” -->    </columns>
            <rows>    <!-- req, xs:integer, “64” -->    </rows>
            <valueTable>    <!—req, xs:string, “base64 string by scope array[rows][columns]( {0,1,0,1,0,1, …})” -->    </valueTable>
      </scope>
      <width>    <!-- ro, xs:integer, “0~96” -->    </width>
      <height>    <!-- ro, xs:integer, “0~64     ” -->    </height>
      <sensitivityInfoList>    <!—opt, sensitivityInfoList must be 4 parts(id=0, id=1, id=2, id=3), part1’s begin time should be 00:00, part2’s begin time should be part1’s end time, part3’s begin time should be part2’s end time, part4’s begin time should be part3’s end time, part4’s end time should be 23:59 -->
            <sensitivityInfo>
                  <id>    <!-- req, xs:integer, “0” -->    </id>
                  <sensitivity>    <!-- req, xs:integer, “0~50,default:9” -->    </sensitivity>
                  <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
                  <beginMinute>    <!-- req, xs:integer, “0~59” -->    </beginMinute>
                  <endHour>    <!-- req, xs:integer, “0~23” -->    </endHour>
                  <endMinute>    <!-- req, xs:integer, “0~59” -->    </endMinute>
```

```
        </sensitivityInfo>
        <sensitivityInfo>
            <id>    <!-- req, xs:integer, “1” -->    </id>
            <sensitivity>    <!-- req, xs:integer, “0~50,default:9” -->    </sensitivity>
            <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
            <beginMinute>    <!-- req, xs:integer, “0~59” -->    </beginMinute>
            <endHour>    <!-- req, xs:integer, “0~23” -->    </endHour>
            <endMinute>    <!-- req, xs:integer, “0~59” -->    </endMinute>
        </sensitivityInfo>
        <sensitivityInfo>
            <id>    <!-- req, xs:integer, “2” -->    </id>
            <sensitivity>    <!-- req, xs:integer, “0~50,default:9” -->    </sensitivity>
            <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
            <beginMinute>    <!-- req, xs:integer, “0~59” -->    </beginMinute>
            <endHour>    <!-- req, xs:integer, “0~23” -->    </endHour>
            <endMinute>    <!-- req, xs:integer, “0~59” -->    </endMinute>
        </sensitivityInfo>
        <sensitivityInfo>
            <id>    <!-- req, xs:integer, “3” -->    </id>
            <sensitivity>    <!-- req, xs:integer, “0~50,default:9” -->    </sensitivity>
            <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
            <beginMinute>    <!-- req, xs:integer, “0~59” -->    </beginMinute>
            <endHour>    <!-- req, xs:integer, “0~23” -->    </endHour>
            <endMinute>    <!-- req, xs:integer, “0~59” -->    </endMinute>
        </sensitivityInfo>
    </sensitivityInfoList>
    <timeBlockList>
        <timeBlock>
            <enable>    <!-- req, xs:integer, “0:disable;1:enable” -->    </enable>
            <weekDay>    <!-- req, xs:string, “Sunday, Monday,...” -->    </weekDay>
```

```
            <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
            <endHour>    <!-- req, xs:, “0~23,bigger or equal beginHour” -->
</endHour>
        </timeBlock>    <!-- opt -->
        ...
    </timeBlockList>
</MD>
```

3.7 PTZ

/PTZ
Notes:

3.7.1 Serial

/PTZ /serial	V1.1
GET	
Description	It is used to get one channel's configuration of PTZ.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Serial**
POST	
Description	It is used to set one channel's configuration of PTZ.
Parameter	channel=<ID>
Inbound Data	**Serial**
Success Return	**ResponseStatus**

Notes:

Serial XML Block
<Serial version="1.1">   <channelId> <!-- ro, xs:integer, “” --> </channelId>   <baudRate> <!-- req, xs:integer, “1200, 2400, 4800, 9600” --> </baudRate>   <dataBit> <!-- req, xs:string, “CS8, CS7, CS6, CS5” --> </dataBit>   <stopBit> <!-- req, xs:integer, “1 or 2” --> </stopBit>   <parity> <!-- req, xs:string, “none, odd, even” --> </parity>   <flowControl> <!-- req, xs:string, “none, hard, xon, xoff” --> </flowControl>   <controlProtocol> <!-- req, xs:string, “PELCO_D, PELCO_P” --> </controlProtocol>   <controlAddress> <!-- req, xs:integer, “default: channelId+1” --> </controlAddress> </Serial>

3.7.2 Preset

/PTZ/preset	V1.1
GET	
Description	It is used to get the information of one preset.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Preset**
POST	
Description	It is used to set the information of one preset.
Parameter	channel=<ID>
Inbound Data	**Preset**

Success Return	ResponseStatus
Notes: if the <command> is include in <preset>, the preset will be set or deleted or called to position. Otherwise, just configure the preset.	

Preset XML Block
<Preset version="1.1"> <channelId> <!-- ro, xs:integer, “” --> </channelId> <presetList> <preset> <id> <!-- req, xs:integer, “0~255” --> </id> <command> <!-- opt, xs:string, “setPos; delPos; toPos” --> </command> <name> <!-- opt, xs:string, “preset name” --> </name> </preset> <!-- opt --> ... </presetList> </Preset>

3.7.3 Patrol

/PTZ/patrol	V1.1
GET	
Description	It is used to get the information of one patrol.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Patrol**
POST	
Description	It is used to set the information of one patrol.

Parameter	channel=<ID>
Inbound Data	**Patrol**
Success Return	**ResponseStatus**
Notes:	

Patrol XML Block

```
<Patrol version="1.1">
    <channelId>  <!-- ro, xs:integer, “” -->  </channelId>
    <patrolList>
        <patrol>
            <id>  <!-- req, xs:integer, “0~5” -->  </id>
            <enable>  <!-- req, xs:integer, “0:disable;1:enable” -->  </enable>
            <starting>  <!-- opt, xs:integer, “0:not start;1:starting” -->  </starting>
            <name>  <!-- opt, xs:string, “patrol name” -->  </name>
            <keyPosList>
                <keyPos>
                    <id>  <!-- req, xs:integer, “0~15” -->  </id>
                    <presetId>  <!-- req, xs:integer, “-1:delete this keyPoint;
0~255” -->  </presetId>
                    <speed>  <!-- req, xs:integer, “1~64” -->  </speed>
                    <dwellTime>  <!-- req, xs:integer, “ptz stay time in one point.
1~30 seconds” -->  </dwellTime>
                </keyPos>  <!-- opt -->
                ...
            </keyPosList>
        </patrol>  <!-- opt -->
        ...
    </patrolList>
</Patrol>
```

3.7.5 Control

/PTZ/control	V1.1
POST	
Description	It is used to control the PTZ.
Parameter	None
Inbound Data	**PtzControl**
Success Return	**ResponseStatus**

Notes:

connect to the ptz command, some parameters are unneeded. you just set it "0".

the value of command is:

"stop": PTZ stop turning.

"left" : PTZ turn left in the specified speed.

"right" : PTZ turn right in the specified speed.

"up" : PTZ turn up in the specified speed.

"down" : PTZ turn down in the specified speed.

"leftUp" : PTZ turn left-up in the specified speed.

"leftDown": PTZ turn left-down in the specified speed.

"rightUp" : PTZ turn right-up in the specified speed.

"rightDown" : PTZ turn right-down in the specified speed.

"irisDec"

"irisInc"

"zoomDec"

"zoomInc"

"focusDec"

"focusInc"

"auto"

"highSpeed"

/*patrol*/

"startPatrol"

"stopPatrol"

"clearPatrol"

PtzControl XML Block
<PtzControl version="1.1"> <channelId> <!-- req, xs:integer, “” --> </channelId> <command> <!-- req, xs:string, “ptz command name, define in the notes above” --> </command> <patrolId> <!-- opt, xs:integer, “0~5” --> </patrolId> <speed> <!-- opt, xs:integer, “1~64” --> </speed> </PtzControl>

3.8 IO

/IO	V1.1
GET	
Description	It is used to get the I/O ports number.
Parameter	None
Inbound Data	None
Success Return	**IOPortInfo**

Notes: If IOInputPortNum is 3, the inputs id is {0,1,2}; If IOOutputPortNum is 3, the outputs id is {0,1,2};

IOPortInfo XML Block
<IOPortInfo version="1.1"> <IOInputPortNum> <!-- ro, xs:integer, “” --> </IOInputPortNum> <IOOutputPortNum> <!-- ro, xs:integer, “” --> </IOOutputPortNum> </IOPortInfo>

3.8.1 Input

/IO/input	**V1.1**
GET	
Description	It is used to get configuration of one particular input port.
Parameter	input=<ID>
Inbound Data	None
Success Return	**IOInputPort**
POST	
Description	It is used to get configuration of one particular input port.
Parameter	input=<ID>
Inbound Data	**IOInputPort**
Success Return	**ResponseStatus**
Notes:	

IOInputPort XML Block
<IOInputPort version="1.1"> <id> <!-- ro, xs:integer, “base on IOInputPortNum” --> </id>

```
    <enable>    <!-- ro, xs:integer, “0:disable IO input alarm;1:enable IO input alarm”
-->    </enable>
    <triggering>    <!-- req, xs:string, “high or low” -->    </triggering>
    <alarmAction>    <!-- req, xs:string, “IOOutputTrigger, buzz, snap or none” -->
</alarmAction>
    <ScheduleTimeBlockList>
        <timeBlock>
            <enable>    <!-- req, xs:integer, “0:disable;1:enable” -->    </enable>
            <weekDay>    <!-- req, xs:string, “Sunday, Monday,...” -->    </weekDay>
            <beginHour>    <!-- req, xs:integer, “0~23” -->    </beginHour>
            <endHour>    <!-- req, xs:, “0~23,bigger or equal beginHour” -->
</endHour>
        </timeBlock>    <!-- opt -->
        ...
    </ScheduleTimeBlockList>
</IOInputPort>
```

3.8.2 Output

/IO/output	**V1.1**
GET	
Description	It is used to get configuration of one particular output port.
Parameter	output=<ID>
Inbound Data	None
Success Return	**IOOutputPort**
POST	
Description	It is used to set configuration of one particular output port.
Parameter	output=<ID>
Inbound Data	**IOOutputPort**

Success Return	ResponseStatus
Notes:	

IOOutputPort XML Block

```
<IOOutputPort version="1.1">
    <id>  <!-- ro, xs:integer, “base on IOOutputPortNum” -->  </id>
    <pulseDuration>  <!-- req, xs:integer, “seconds” -->  </pulseDuration>
    <ScheduleList>
        <Schedule>
            <alarmType>  <!-- req, xs:string, “IO,MD,videoloss or none” -->
</alarmType>
            <timeBlockList>
                <timeBlock>
                    <enable>  <!-- req, xs:integer, “0:disable;1:enable” -->
</enable>
                    <weekDay>  <!-- req, xs:integer, “Sunday, Monday,...” -->
</weekDay>
                    <beginHour>  <!-- req, xs:integer, “0~23” -->  </beginHour>
                    <endHour>  <!-- req, xs:, “0~23,bigger or equal beginHour”
-->  </endHour>
                </timeBlock>  <!-- opt -->
                ...
            </timeBlockList>
        </Schedule>  <!-- opt -->
        ...
    </ScheduleList>
</IOOutputPort>
```

3.8.3 Output Status

/IO/output/status	V1.1
GET	
Description	It is used to get status of one particular output port.
Parameter	output=<ID>
Inbound Data	None
Success Return	**IOOutputPortStatus**
Notes:	

IOOutputPortStatus XML Block
<IOOutputPortStatus version="1.1"> <id> <!-- ro, xs:integer, “base on IOOutputPortNum” --> </id> <status> <!-- req, xs: string, “inactive; active ” --> </status> </IOOutputPortStatus>

3.8.4 Output Trigger

/IO/output/trigger	V1.1
POST	
Description	It is used to manually trigger a particular output port.
Parameter	output=<ID>
Inbound Data	**IOOutputPortStatus**
Success Return	**ResponseStatus**
Notes: IOOutputPortStatus is shown in /IO/output/outputs/ID/status above.	

3.9 Image

/Image
Notes:

3.9.1 Power Line Frequency

/Image/powerLineFrequency	V1.1
GET	
Description	It is used to get the powerLineFrequency value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**PowerLineFrequency**
POST	
Description	It is used to set the powerLineFrequency value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**PowerLineFrequency**
Success Return	**ResponseStatus**
Notes: Configure the powerlineFrequency requires to reboot the camera.	

PowerLineFrequency XML Block
<PowerLineFrequency version="1.1"> <mode> <!-- req, xs:string, “50hz, 60hz” --> </mode> <enable> <!-- req, xs:integer, “0:disable; 1:enable” --> </enable>

</PowerLineFrequency>

3.9.2 Exposure

/Image/exposure	V1.1
GET	
Description	It is used to get exposure configuration of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**ExposureCfg**
POST	
Description	It is used to set exposure configuration of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**ExposureCfg**
Success Return	**ResponseStatus**
Notes:	

ExposureCfg XML Block
<ExposureCfg version="1.1"> <exposureType> <!-- req, xs:string, “auto, shutterFirst, gainFist, manual” --> </exposureType> </ExposureCfg>

3.9.3 Shutter

/Image/shutter	V1.1
GET	
Description	It is used to get shutter value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Shutter**
POST	
Description	It is used to configure shutter value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**Shutter**
Success Return	**ResponseStatus**
Notes: Shutter's POST operate is enabled only when <exportType> is shutterFirst.	

Shutter XML Block
<Shutter version="1.1"> <shutterLevel> <!-- req, xs:string, “1/3, 1/4, 1/5, 1/6, 1/8, 1/12, 1/15, 1/25, 1/30, 1/50, 1/60, 1/100, 1/120, 1/250, 1/500, 1/1000, 1/2000, 1/4000, 1/10000” --> </shutterLevel> </Shutter>

3.9.4 Gain

/Image/gain	V1.1
GET	
Description	It is used to get gain value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Gain**

POST	
Description	It is used to configure gain value of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**Gain**
Success Return	**ResponseStatus**
Notes:	

Gain XML Block
<Gain version="1.1">   <gainLevel> <!-- req, xs:integer, “0~100, default:10” --> </gainLevel> </Gain>

3.9.5 Scene

/Image/scene	V1.1
GET	
Description	It is used to get scene configuration of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Scene**
POST	
Description	It is used to set scene configuration of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**Scene**
Success Return	**ResponseStatus**
Notes:	

Scene XML Block
<Scene version="1.1">   <sceneMode> <!-- req, xs:string, “indoor, outdoor” --> </sceneMode> </Scene>

3.9.6 IR

/Image/IR	V1.1
GET	
Description	It is used to get ircut mode of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**IR**
POST	
Description	It is used to configure ircut mode of a specified Image channel.
Parameter	channel=<ID>
Inbound Data	**IR**
Success Return	**ResponseStatus**
Notes:	

IR XML Block
<IR version="1.1">   <mode> <!-- req, xs:string, “auto, on, off” --> </mode> </IR>

3.9.7 BLC

/Image/BLC	V1.1
GET	
Description	It is used to get the configuration of backlight compensation for a specified image channel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**BLC**
POST	
Description	It is used to set the configuration of backlight compensation for a specified image channel.
Parameter	channel=<ID>
Inbound Data	**BLC**
Success Return	**ResponseStatus**
Notes:	

BLC XML Block

```
<BLC version="1.1">
    <enable>  <!-- req, xs:integer, “0:disable;1:enable” -->  </enable>
    <BLCMode>  <!-- req, xs:string, “backLight, dynamicRange” -->  </ >
    <level>  <!-- req, xs:integer, “0~255, default:128” -->  </level>
</BLC>
```

3.9.8 Mirror

/Image/mirror	V1.1
GET	

Description	It is used to get mirror status of a specified image channnel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Mirror**
POST	
Description	It is used to set mirror status of a specified image channnel.
Parameter	channel=<ID>
Inbound Data	**Mirror**
Success Return	**ResponseStatus**
Notes:	

Mirror XML Block
<Mirror version="1.1"> <status> <!-- req, xs:integer, “0:no mirror;1:use mirror” --> </status> </Mirror>

3.9.9 Flip

/Image/flip	**V1.1**
GET	
Description	It is used to get flip status of a specified image channnel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**Flip**
POST	
Description	It is used to set flip status of a specified image channnel.
Parameter	channel=<ID>
Inbound Data	**Flip**

Success Return	ResponseStatus
Notes:	

Flip XML Block
<Flip version="1.1"> <status> <!-- req, xs:integer, “0:no flip;1:use flip” --> </status> </Flip>

3.10 Event

/Event
Notes:

3.10.1 StartAlarm

/Event/startAlarm	V1.1
POST	
Description	It is used to start alarm.
Parameter	None
Inbound Data	None
Success Return	ResponseStatus
Notes:	

3.10.2 Stop Alarm

/Event/stopAlarm	V1.1
POST	
Description	It is used to stop alarm.
Parameter	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes:	

3.10.3 Alarm Event

/Event/alarmEvent	V1.1
GET	**Viewer**
Description	It is used to get current alarm event status of a specified image channnel.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**AlarmEvent**
Notes:	

AlarmEvent XML Block
<AlarmEvent version="1.1"> <channelId> <!-- ro, xs:integer, “” --> </channelId> <status> <!-- ro, xs:string, “MD, videoLoss, blind, IOAlarm or none” --> </status> </AlarmEvent>

3.11 Record

/Record
Notes:

3.11.1 Recording Schedule

/Record/Schedule	V1.1
GET	
Description	It is used to get recording configuration and time range.
Parameter	channel=<ID>
Inbound Data	None
Success Return	**RecordSchedule**
POST	
Description	It is used to get recording configuration and time range.
Parameter	channel=<ID>
Inbound Data	**RecordSchedule**
Success Return	**ResponseStatus**
Notes:	

RecordSchedule XML Block
<RecordSchedule version="1.1"> <channelId> <!-- ro, xs:integer, “” --> </channelId> <enable> <!-- ro, xs:integer, “0:disable record;1:enable record” --> </enable> <recordDelayTime> <!-- req, xs:integer, “seconds” --> </recordDelayTime>

```
    <preRecordTime>   <!-- req, xs:integer, “seconds” -->   </preRecordTime>
    <packageTime>   <!-- req, xs:integer, “minutes” -->   </packageTime>
    <ScheduleList>
        <Schedule>
            <alarmType>   <!-- req, xs:string, “IO,MD,videoloss or none” -->
</alarmType>
            <timeBlockList>
                <timeBlock>
                    <enable>   <!-- req, xs:integer, “0:disable;1:enable” -->
</enable>
                    <weekDay>   <!-- req, xs:string, “Sunday, Monday,...” -->
</weekDay>
                    <beginHour>   <!-- req, xs:integer, “0~23” -->   </beginHour>
                    <endHour>   <!-- req, xs:, “0~23,bigger or equal beginHour”
-->   </endHour>
                </timeBlock>   <!-- opt -->
                ...
            </timeBlockList>
        </Schedule>   <!-- opt -->
        ...
    </ScheduleList>
</RecordShedule>
```

3.12 Two Way Audio

/TwoWayAudio
Notes:

You would get intercom data in RTSP Preview, and send intercom data using /TwoWayAudio/sendData.

3.12.1 Open

/TwoWayAudio/open	V1.1
POST	
Description	It is used to open intercom.
Parameter	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes:	

3.12.2 Close

/TwoWayAudio/close	V1.1
POST	
Description	It is used to close intercom.
Parameter	None
Inbound Data	None
Success Return	**ResponseStatus**
Notes: If the intercom is opened, but do not receive any data or "close" command in 20 seconds, the intercom will be closed by itself.	

3.12.3 Send Data

/TwoWayAudio/sendData	V1.1
POST	
Description	It is used to send the intercom data.
Parameter	None
Inbound Data	**Opaque data (pcm format)**
Success Return	**ResponseStatus**
Notes:	

NOTES:

1. For IP Camera, now only support one input channel. <id> associated with the input channel can only be 0.

2. XML Data has two modes. For example:

Response data

```
<?xml version="1.0" encoding="UTF-8"?>
<body>
    <Upnp>
        <enable>1</enable>
    </Upnp>
</body>
```

Response status

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseStatus>
    <statusCode>200</statusCode>
```

```
    <statusString>OK</statusString>
</ResponseStatus>
```

<body> is used for the Response data. It contains all the parameters which defined above.

If the request return ok, the statusCode is 200. And the statusString will show the detail.