IP Media Device Management Protocol
User Guide
Interlogix Open Standards Devices

Version 1.0

Revision 5.8

2011-01

Contents

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.

With regard to Media Streaming, please refer to “develop API of 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:

Special Resources: “index”, “indexr”, “description” and “capabilities” resources, that are contained in all Services and General Resources, and provide a special description for these resources.
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:

FQDN	Fully Qualified Domain Name
REST	REpresentational State Transfer
IO	Input/Output
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.

Table 1

Resource	GET	PUT	POST	DELETE
Collection URI, such as http://webServer/resources	**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 operation.	Meaning defined as "**delete** the entire collection".
Member URI, such as http://webServer/resources/7416	**Retrieve** a representation of the 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

Three kinds of resources are defined in this specification. They are “Special Resources”, “Services” and “General Resources”. Related General Resources are grouped by Services. Services and General Resources contain Special Resources. Figure 1 shows their relationship.

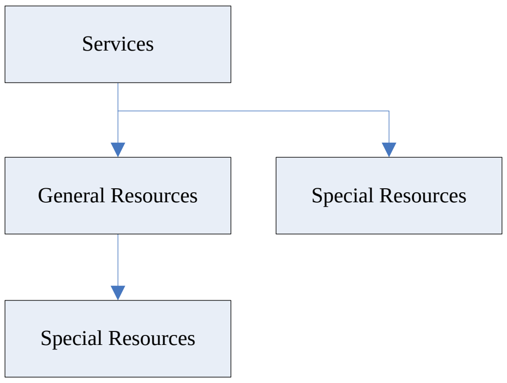


Figure 1

The “index”, “indexr”, “description” and “capabilities” are defined as Special Resources in this specification. Both “index” and “description” will be mandatorily included by each node, and both “indexr” and “capabilities” will be optionally included by each node. For more detailed description see Section 6.

Services defined in this specification are divided into different services categories. Each category has its own name spaces (see Section 4.6 for the name space definitions). The following services are defined:

Table 2

Services	Description	Reference
System	Configure and operate the general system functions.	8.1
Network	Configure network interfaces.	8.2
IO	Configure the Input/Output (IO).	8.3
Video	Handle video-related configuration.	8.4
Audio	Configure the Audio.	8.5
Two way audio	Control two ways audio.	8.6
Serial	Configure and control the Serial ports.	8.7
Security	Provide Security functions.	8.8
Streaming	Configure and control the streaming media content.	8.9
Motion Detection	Configure and control the motion detection of the device	8.10
Event	Provide event notification functions.	8.11
PTZ	Control the device pan tilt and zoom.	8.12

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.

4.5.1Connection 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”
…
<?xml version=”1.0” encoding=”UTF-8”?>
<DeviceStatus version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
…
</DeviceStatus>
```

For data(i.e. firmware, configuration file, etc.), the Content-Type may be “application/octet-stream”. For example,

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.std-cgi.com/ver10/XMLSchema”>
…
</ResponseStatus>
```

4.5.4Operations

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.

4.5.5Error 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.

Table 3

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
200	“OK”-The request has succeeded.	√	√		√

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
	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 location.	√			
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. Header Notes: The Location header contains the URI of the resource. Body Notes: The body may contain the new resource location.	√			
400	“Bad Request” – The request was badly formed. This is commonly used for creating or		√	√	

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
	updating a resource, but the data was incomplete or incorrect. Header Notes: The Reason-Phrase sent with the HTTP status header may contain information on the error. Body Notes: The response may contain more information of the underlying error that occurred in addition to the Reason-Phrase.				
401	"Unauthorized" – The request requires user authentication to access this resource. If the request contains invalid authentication data, this code is sent. Header Notes: At least one authentication mechanism must be specified in the WWW-Authenticate header. The Reason-Phrase sent with the HTTP status header may contain information on the error. Body Notes: The response may contain more information of the underlying error that occurred in addition to the Reason-Phrase.	√	√	√	√
403	"Forbidden" – The request is not allowed because the server is refusing to fill the request. A common reason for this is that the device does not support the requested functionality. Header Notes: The Reason-Phrase sent with the HTTP status header may contain information on the error. Body Notes: The response may contain more information of the underlying error that occurred in	√	√	√	√

HTTP Status Codes	REST Meaning	GET	PUT	POST	DELETE
	addition to the Reason-Phrase.				
404	"Not Found" – The requested resource does not exist. Header Notes: None Body Notes: None	√	√	√	√
405	"Method Not Allowed" – The request used an HTTP method that is not supported for the resource because the specification does not allow this method. If the device does support the functionality but it is a valid operation (that has been defined in this specification), then 403 is returned. Header Notes: The Allow header lists the supported HTTP methods for this resource. Body Notes: None	√	√	√	√
500	"Internal Server Error" - An internal server error has occurred. Header Notes: None Body Notes: None	√	√	√	√
503	"Service Unavailable" – The HTTP Server is up, but the REST service is not available. Typically this is caused by too many client requests. Header Notes: The Retry-After header suggests to the client when to try resubmitting the request. Body Notes: None	√	√	√	√

Responses to many resources calls contain data in XML format. XML Response Schema is defined in Annex. XML Response Schema consists of the following sections:

- requestURI - the URI of the corresponding HTTP request message
- statusCode - indicating the status of the REST operations.

Table 4

statusCode	Description
1	“OK” - indicate a successful operation is done (remark: if the request contains some parameters that are not supported, the device will ignore those parameters and return OK as statusCode)
2	“Device Busy” - for a command which cannot be processed at that time (i.e. if the device receives a reboot command during upgrading process)
3	“Device Error” - if the device can not perform the request for a hardware error. An error message in statusString format to indicate operation failure
4	“Invalid Operation” - either if the operation is not supported by the device, or if the user has not passed the authentication, or if the user does not have enough privilege for this operation
5	“Invalid XML Format” - if the XML format is not recognized by the system. There will be statusString returned to represent different errors
6	“Invalid XML Content” - an incomplete message or a message containing an out-of-range parameter. Relative statusString will be return.
7	“Reboot Required” - If a reboot is required before the operation taking effect

- statusString – error type for the not completed operation.
- id – Return the ID created by the device in POST operation

4.6 Namespaces

The namespace xmlns:std-cgi=“http://www.std-cgi.com/ver10/XMLSchema” is used in this specification.

The following namespaces are referenced by this specification:

- xmlns:xs=”http://www.w3.org/2001/XMLSchema”
- xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
- xmlns:xlink=”http://www.w3.org/1999/xlink”

4.7 Security

User-based access control is adopted in this specification. Security policy configuration in this specification based on three different user levels.

- Administrator – the privilege can access all supported resources on IP device.
- Operator – the privilege can access some general-level and higher-level resources. See the Resource Description of each resource for details.
- Viewer – the privilege can only access some general-level resources. See the Resource Description of each resource for details.

In order to access all supported resources, one account with Administrator privilege level must be active at all times. A default user account “admin” is provided by all IP devices. It has an Administrator user level, and must not be deleted. Its default password is “12345”.

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.

The UPnP architecture supports zero-configuration networking, and the device can dynamically join a network, obtain IP address, announce its name, convey its capabilities upon request, and gets the on-line status and capabilities of other devices. DHCP and DNS servers are optional and are only used if they are available on the network. Devices can leave the network automatically without leaving any unwanted status information behind. UPnP was published as a 73-part International Standard, ISO/IEC 29341, in December, 2008 [6][7][8].

The foundation for UPnP networking is IP addressing. When a device is connected to the network for the first time, its Dynamic Host Configuration Protocol (DHCP) client will search for a DHCP server. If the device successfully get its domain name via DNS server or DNS forwarding, then it should use this domain name for the following network operations; if the network is unmanaged and no DHCP server is found, the device must assign an address for itself, which is known as “AutoIP” of the UPnP Device Architecture [9][10], and use this IP address for the following network operations.

Once given an IP address, the Discovery process will be executed in UPnP networking. The UPnP discovery protocol is also knows as Simple Service Discovery Protocol (SSDP). When a device is added to the network, SSDP allow that device to announce its services to the control points on the network. Similarly, when a control point is added to the network, SSDP allows that control point to search for relative devices on the network. During the above searching or announcing process, a a discovery message which contains essential device specifics or one of its services will be transfered, for example, device type, identifier, and a pointer to more detailed information.

After a control point has discovered a device, the control point still needs more operations to request more information about the device or to interact with it. An HTTP GET request for mandatory index Special Resource will return a list of the resources supported by the device.
Remark: the index resource will only return the first level resources of a node, while the indexr Special Resource will return a complete folder list in tree structure with the current

resource as root folder.

6 Resource Description

6.1 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: d*escribes 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 4.5.5.

6.2 Built-in Types

Table 5

Type	Description
BaudRate	A positive numerical value indicating the data transmission rate in symbols per second.

	Value is>=0. Example: 9600
Color	RGB triplet in hexadecimal format (3 bytes) without the preceding "0x". Example: “FF00FF”
Coordinate	A positive numerical value in pixels. A coordinate pair of 0,0 (x,y) indicates the bottom-left corner of the video image. Value is>=0. Maximum value is dependent on video resolution.
FPS	Frame rate multiplied by 100. Example: 2500 [PAL]
IPv4 Address	Notation is xxx.xxx.xxx.xxx Example: 3.137.217.220
MAC	MAC Address Notation is aa:bb:cc:dd:ee:ff with 6 hex bytes.

6.3 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 Special Resources

7.1 index

index	**Special Resource v1.0**
GET	**Viewer**
Description	Enumerate child resources of a resource.
Query	None
Inbound Data	None
Success Return	ResourceList **ResourceList**
Notes: Returns a non-recursive resource listing of all child resources.	

7.2 indexr

indexr	**Special Resource v1.0**
GET	**Viewer**
Description	Enumerate child resources of a resource.
Query	None
Inbound Data	None
Success Return	ResourceList **ResourceList**
Notes: Returns a recursive resource listing of all child resources.	

7.3 description

description	**Special Resource v1.0**
GET	**Viewer**
Description	Describe the corresponding resource
Query	None
Inbound Data	None
Success Return	ResourceDescription **ResourceDescription**
Notes: <version> set the version of resource. In this specification, its value is “1.0”.	

A version attribute is included in the description. This means resources with different versions may exist within the same Services. In that case, the version of Services is the version of the contained resource with the lowest version, and all resources in the Services container must be backward compatible. If any resource of a Service container can not maintain backward compatibility with previous versions, a new Services version should be introduced.”

7.4 capabilities

capabilities	**Special Resource v1.0**
GET	**Viewer**
Description	Describe the capabilities of the corresponding resource
Query	None
Inbound Data	None
Success Return	the XML Data Block resource-specified
Notes:	

For the General Resource, which inbound data is specified as an XML payload, the Special Resource (capabilities) is provided for video management systems or client applications to query an IP device and understand what XML tags are supported.

“Capabilities” is essentially an XML instance of the corresponding General Resource XML Data Block. “Capabilities” must contain the acceptable values for each attribute.

While XML Schema Document are also required of any XML data defined by this specification and xsd documents are capable of defining the acceptable range of values for any attribute, using a global xsd to define capacities would imply that all devices support the same options for any parameter. By allowing devices to respond to the capabilities request, each device can support different values for any attribute, within the constraints of the schema.

Table 7

Capabilities Attribute	**Description**	**Syntax**	**Applicable XML Data Types**

min	The minimum character length for a string, or the minimum numerical value of a number	Examples:
		min=”0”
		min=”19”
		min=”-74”(numerical only)
		min=”1.6”
			All except fixed data types$^{1)}$

max	The maximum character length for a string, or the maximum numerical value of a number	Examples:
		max=”4”
		max=”37”
		max=”8192”
		max=”14.61”
			All except fixed data types$^{1)}$

range	Indicates the possible range of numerical values within the “min” and “max”	Ranges are listed in numerical order separated by a “,”	All numerical data types

Capabilities Attribute	Description	Syntax	Applicable XML Data Types
	attributes of an element. This attribute should only be used if the possible value for an XML element does not include the entire numerical range between “min” and “max” attributes	character. A range has the form “x~y” where x is the range floor and y is the range ceiling. Single numbers may also be used. Example: if an XML element supports values 0, 456, 1674 to 2009 and 2012, the syntax would be: range=”0, 456, 1674~2009, 2012”	

opt	Lists the supported options for a CodeID data type. Required for XML elements with a CodeID data type. This attribute should not be used for any other data type	If all options are supported, the syntax is “all”. Otherwise, supported options are listed separated by a “, ” character.
		
		Examples:
		opt=”all”
		opt=”1, 4, 6, 7”
			CodeID

def	Indicates the default value of the XML element. If the element has not default value, this attribute should not be used	Examples:
		def=”7416”
		def=”ace”
			All data types

reqReboot	Indicates if configuration of this XML element requires a device reboot before taking effect. If an element does not require a boot, this attribute should not be used	reqReboot=”true”	All data types
dynamic	Indicates if an XML element has dynamic capabilities dependent on other XML configuration. For example, if an element’s data range changes based on another element’s configured value,	dynamic=”true”	All data types

Capabilities Attribute	Description	Syntax	Applicable XML Data Types
	this attribute must be used. In this case, the element's capability attributes must always reflect the current device configuration		
Size	Indicates the maximum number of entries in an XML List. This attribute is only applicable to XML list elements. This attribute should not be used for any other type of element	Example: If a device supports 16 users the example would be <UserList size="16"> <User> … </UserList>	Only supported for list elements

1) Fixed, pre-defined data types do not need certain capability attributes because their formats/data ranges are already defined.

Special Resources do not contain themselves.

The requestURIs "/index", "/indexr", "/description" are required.

8 Services and General Resources

8.1 System

/System	**Service v1.0**
Notes:	

8.1.1Device Information

/System/deviceInfo	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get device information.
Query	None
Inbound Data	None
Success Return	**DeviceInfo**
PUT	**Administrator**
Description	It is used to update device information.

Query	None
Inbound Data	**DeviceInfo**
Success Return	ResponseStaus **ResponseStatus**

Notes:
Some fields are read-only and may not be set. If these fields are present in the inbound XML block, they are ignored.
For the <DeviceInfo> uploaded to the device during a PUT operation, all fields are considered optional and any fields that are not present in the inbound XML are not changed on the device. This allows setting of the fields individually without having to load the entire XML block to the device.
<deviceDescription> is a description of the device as defined in RFC1213.
<deviceLocation> is the location of the device as defined in RFC1213
<systemContact> is the contact information for the device as defined in RFC1213.

DeviceInfo XML Block

```
<DeviceInfo version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <deviceName>        <!-- req, xs:string -->          </deviceName>
   <deviceID>           <!-- req, xs:integer, “1-255”-->          </deviceID>
      <!-- Note: The following are read-only parameters -->
   <deviceDescription>   <!-- ro, req, xs:string -->       </deviceDescription>
   <deviceLocation>      <!-- ro, req, xs:string -->       </deviceLocation>
   <systemContact>       <!-- ro, req, xs:string -->        </systemContact>
   <model>                <!-- ro, req, xs:string -->       </model>
   <serialNumber>        <!-- ro, req, xs:string -->       </serialNumber>
   <macAddress>          <!-- ro, req, xs:string;     -->   </macAddress>
   <firmwareVersion>      <!-- ro, req, xs:string -->       </firmwareVersion>
   <firmwareReleasedDate>   <!-- ro, opt, xs:string -->       </firmwareReleasedDate>
   <bootVersion>          <!-- ro, opt, xs:string -->       </bootVersion>
   <bootReleasedDate>      <!-- ro, opt, xs:string -->       </bootReleasedDate>
   <hardwareVersion>      <!-- ro, opt, xs:string -->       </hardwareVersion>
</DeviceInfo>
```

8.1.2Configuration file(s)

/System/configurationFile	**General Resource v1.0**
GET	**Administrator**
Description	It is used to get device’s configuration file(s).
Query	None
Inbound Data	None
Success Return	**Opaque Data**
PUT	**Administrator**

Description	It is used to update device's configuration file(s).
Query	None
Inbound Data	**Opaque Data**
Success Return	ResponseStaus **ResponseStatus**
Notes: Configuration file is device-dependant – it may be binary or any other format. Should reboot device after configuration file is applied.	

8.1.3Factory default

/System/factoryDefault	**General Resource v1.0**
PUT	**Administrator**
Description	It is used to reset the configuration for the device to the factory default.
Query	mode
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Two factory reset modes are supported: "full" resets all device parameters and settings to their factory values. "basic" resets all device parameters and settings except the values in Network Service. The default mode is "full". The device should be rebooted after it is reset.	

8.1.4Firmware upgrade

/System/firmwareUpgrade	**General Resource v1.0**
PUT	**Administrator**
Description	It is used to upgrade the firmware of the device.
Query	None
Inbound Data	**Opaque Data**
Success Return	ResponseStaus **ResponseStatus**
Notes: The device should be rebooted after the upgrade is completed.	

8.1.5Reboot

/System/reboot	**General Resource v1.0**
PUT	**Administrator**
Description	It is used to reboot the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: **ResponseStatus** is returned before the device proceeds to reboot.	

8.1.6Status

/System/status	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the status information of the device.
Query	None
Inbound Data	None
Success Return	**DeviceStatus**
Notes:	

DeviceStatus XML Block

```
<DeviceStatus version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <currentDeviceTime>   <!-- req, xs:datetime -->        </currentDeviceTime>
  <deviceUpTime>        <!-- req, xs:integer, seconds -->   </deviceUpTime>
  <CPUList>             <!-- req -->
    <CPU>
      <cpuDescription>   <!-- req, xs:string -->              </cpuDescription>
      <cpuUtilization>   <!-- req, xs:integer, percentage 0..100 -->   </cpuUtilization>
    </CPU>
  </CPUList>
  <MemoryList>          <!-- req -->
    <Memory>
      <memoryDescription>   <!-- req, xs:string -->          </memoryDescription>
      <memoryUsage>          <!-- req, xs:float, in MB -->   </memoryUsage>
      <memoryAvailable>      <!-- req, xs:float, in MB-->   </memoryAvailable>
    </Memory>
  </MemoryList>
</DeviceStatus>
```

8.1.7Time

/System/time	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the device time information.
Query	None
Inbound Data	None
Success Return	**Time**
PUT	**Administrator**
Description	It is used to udpate the device time information.
Query	None
Inbound Data	**Time**
Success Return	ResponseStaus **ResponseStatus**
Notes: If <timeMode> is present and set to "local", the <localTime> and <timeZone> fields are required. The <localTime> block sets the device time. If <timeMode> is present and set to "NTP", only the <timeZone> field is required. The device time is set by synchronizing with NTP.	

Time XML Block

```
<Time version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <timeMode>     <!-- req, xs:string, "NTP, manual" -->       </timeMode>
   <localTime>      <!-- req, xs:datetime -->                </localTime>
   <timeZone>     <!-- req, xs:string, POSIX time zone string -->       </timeZone>
</Time>
```

8.1.8LocalTime

/System/time/localTime	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the device local time information.
Query	None
Inbound Data	None
Success Return	**ISO 8601 Date-Time String**
PUT	**Administrator**
Description	It is used to udpate the device local time information.
Query	None
Inbound Data	**ISO 8601 Date-Time String**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

An ISO 8601 Date/Time string is accepted and returned. If the date/time value has a time zone, the time is converted into the device's local time zone.
If the device time mode is set to "ntp" setting this value has no effect.

8.1.9TimeZone

/System/time/timeZone	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the device time zone information.
Query	None
Inbound Data	None
Success Return	**Time zone string**
PUT	**Administrator**
Description	It is used to udpate the device time zone information.
Query	None
Inbound Data	**Time zone string**
Success Return	ResponseStaus **ResponseStatus**

Notes:

Time zones are defined by POSIX 1003.1 section 8.3 time zone notations. Note that the value following the +/- is the amount of time that must be added to the local time to result in UTC.

Example:

EST+5EDT01:00:00,M3.2.0/02:00:00,M11.1.0/02:00:00

Defines eastern standard time as "EST" with a GMT-5 offset. Daylight savings time is called "EDT", is one hour later and begins on the second Sunday of March at 2am and ends on the first Sunday of November at 2am.

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 ("CEST") that starts on the last Sunday in March at 2am and ends on the last Sunday in October at 3am.

8.1.10 NtpServers

/System/time/ntpServers	**General Resource v1.0**
GET	**Viewer**

Description	It is used to get the configuration of NTP servers for the device.
Query	None
Inbound Data	None
Success Return	**NTPServerList**
PUT	**Administrator**
Description	It is used to update the configuration of NTP servers for the device.
Query	None
Inbound Data	**NTPServerList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Administrator**
Description	It is used to add the configuration of a NTP server for the device.
Query	None
Inbound Data	**NTPServer**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete the configuration of NTP servers for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: When the <timeMode> is set to "NTP", the servers in this list are used to synchronize the device's system time.	

NTPServerList XML Block

```
<NTPServerList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <NTPServer/> <!-- opt -->
</NTPServerList>
```

8.1.11 NtpServer

/System/time/ntpServers/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the configuration of a NTP server for the device.
Query	None
Inbound Data	None
Success Return	**NTPServer**
PUT	**Administrator**
Description	It is used to update the configuration of a NTP server for the device.
Query	None
Inbound Data	**NTPServer**
Success Return	ResponseStaus **ResponseStatus**

DELETE	Administrator
Description	It is used to delete the configuration of a NTP server for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server.	

NTPServer XML Block

```
<NTPServer version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>   <!-- req, xs:integer, "1"-->      </id>
   <addressingFormatType>
      <!-- xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>       <!-- dep, xs:string -->      </hostName>
   <ipAddress>        <!-- dep, xs:string -->      </ipAddress>
   <portNo>         <!-- ro, opt, xs:integer -->      </portNo>
</NTPServer>
```

8.1.12 Log

/System/logging	General Resource v1.0
GET	**Viewer**
Description	It is used to get the log information of the device.
Query	majorType minorType startTime stopTime
Inbound Data	None
Success Return	**LogList**
Notes: The value of "majorType" is: 0x1:Alarm 0x2:Exception 0x3:Operation When the value of "majorType" is 0x1, the value of "minorType" is: 0x1: alarm input 0x2: alarm output 0x3: motion detection alarm start 0x4: motion detection alarm stop 0x5: shelter alarm start	

0x6: shelter alarm stop
When the value of “majorType” is 0x2, the value of “minorType” is:
0x21: video loss
0x22: illegal access
0x23: hard disk full
0x24: hard disk error
0x25: modem off-line
0x26: ip address conflict
0x27: network not connected
When the value of “majorType” is 0x3, the value of “minorType” is:
0x41: boot
0x42: shutdown
0x43: illegal shut down
0x50: login(local)
0x51: logout(local)
0x52: config parameter(local)
0x53: playback by file name(local)
0x54: playback by time(local)
0x55: start record(local)
0x56: stop record(local)
0x57: PTZ control(local)
0x58: preview(local)
0x59: modify date/time(local)
0x5a: upgrade software(local)
0x70: login(remote)
0x71: logout(remote)
0x72: start record(remote)
0x73: stop record(remote)
0x74: start transparent channel(remote)
0x75: stop transparent channel(remote)
0x76: get parameter(remote)
0x77: config parameter(remote)
0x78: get status(remote)
0x79: on guard(remote)
0x7a: disarm(remote)
0x7b: reboot(remote)
0x7c: start voice talk
0x7d: stop voice talk
0x7e: upgrade software(remote)
0x7f: playback by file name(remote)
0x80: playback by time(remote)
0x81: PTZ control(remote)

The format of “startTime” and “stopTime” is “YYYY-MM-DDThh:mm:ss”.

Devices support up to 2000 log.

LogList XML Block

```
<LogList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <Log>   <!-- opt -->
      <id>        <!-- req, xs:integer -->   </id>
      <time>     <!—req, xs:datetime --> </time>
      <majorType>   <!—req, xs:string --> </majorType>
      <minorType>   <!—req, xs:string -->             </minorType>
      <netUser> <!—req, xs:string --> </netUser>
      <hostIPAddress> <!—req, xs:string --> </hostIPAddress>
      <channel> <!—req, xs:integer --> </channel>
   </Log>
</LogList>
```

8.1.13 Storage

/System/Storage	**resource v1.0**
Notes: service of Storage	

8.1.13.1 Storage/volumes

/System/Storage/volumes	
GET	**Viewer**
Description	It is used to get the storage volumes and files information on a device
Query	None
Inbound Data	None
Success Return	**StorageVolumeList**
PUT	**Operator**
Description	It is used to update the storage volumes and files configuration on a device.
Query	None
Inbound Data	**StorageVolumeList**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

StorageVolumeList XML Block

```
<StorageVolumeList version="1.0" xmlns="urn:psialliance-org">
  <StorageVolume/>   <!-- ro, opt -->
```

```
</StorageVolumeList>
```

8.1.13.2 Storage/volumes/ID

/System/Storage/volumes/ID	
GET	**Viewer**
Description	It is used to get a special storage volume information on a device
Query	None
Inbound Data	None
Success Return	**StorageVolume**
Notes:	

StorageVolume XML Block

```
<StorageVolume version="1.0" xmlns="urn:psialliance-org">
  <id>     <!-- ro, req, xs:string;id --> </id>
  <volumeName> <!-- ro, req, xs:string -->     </volumeName>
  <volumePath>   <!-- ro, opt, xs:string -->     </volumePath>
  <volumeDescription><!-- ro, opt, xs:string -->     </volumeDescription>
  <volumeType>
  <!-- ro, req, xs:string, "VirtualDisk,RAID0,RAID1,RAID0+1,RAID5", etc -->
  </volumeType>
  <storageDescription>
    <!-- ro, opt, xs:string, "DAS","DAS/USB", etc -->
  </storageDescription>
    <storageLocation>
        <!-- ro, opt, xs:string, "HDD","Flash","SDIO", etc-->
    </storageLocation>
    <storageType>
  <!-- ro, opt, xs:string, "internal,external" -->
  </storageType>
      <capacity>      <!-- ro, req, xs:float, in MB -->   </capacity>
  <status> <!--ro, req, xs:string "HD_NORMAL, HD_ERROR, HD_IDLE" --> </status>
</StorageVolume>
```

8.1.13.3 Storage/volumes/ID/status

/System/Storage/volumes/ID/status	
GET	**Viewer**
Description	It is used to get a special storage volume status on a device

Query	None
Inbound Data	None
Success Return	**StorageVolumeStatus**
Notes: Query the volume status. Currently only the amount of free space is returned. Devices may extend the XML to allow for querying additional information.	

StorageVolumeStatus XML Block

```
<StorageVolumeStatus version="1.0" xmlns="urn:psialliance-org">
   <freeSpace>    <!-- ro, req, xs:float, in MB -->    </freeSpace>
</StorageVolumeStatus>
```

8.1.13.4 Storage/volumes/ID/format

/System/Storage/volumes/ID/	
PUT	**Viewer**
Description	It is used to format a storage device
Query	None
Inbound Data	None
Success Return	**StorageVolumeStatus**
Notes: Formating may take time.	

8.1.13.5 Storage/volumes/ID/isFormat

/System/Storage/volumes/ID/IsFormat	
GET	**Viewer**
Description	It is used to access the procedure of formating
Query	None
Inbound Data	None
Success Return	**StorageVolumeFormatSatus**
Notes: formatSatus show the percentage of formatted part of the storage device.	

StorageVolumeStatus XML Block

```
<StorageVolumeFormatSatus version="1.0" xmlns="urn:psialliance-org">
   <formatSatus><!-- req, xs:integer,"0--100"--></formatSatus>
</StorageVolumeFormatSatus>
```

8.2 Network

/Network	Service v1.0
Notes: Network configuration.	

8.2.1Interfaces

/Network/interfaces	General Resource v1.0
GET	**Viewer**
Description	It is used to get the device network interfaces.
Query	None
Inbound Data	None
Success Return	**NetworkInterfaceList**
Notes: As hardwired system resources, network interfaces cannot be created or destroyed.	

NetworkInterfaceList XML Block

```
<NetworkInterfaceList version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <NetworkInterface/>    <!-- opt -->
</NetworkInterfaceList>
```

8.2.2Interface

/Network/interfaces/*ID*	General Resource v1.0
GET	**Viewer**
Description	It is used to get a particular network interface.
Query	None
Inbound Data	None
Success Return	**NetworkInterface**
PUT	**Administrator**
Description	It is used to update a particular network interface.
Query	None
Inbound Data	**NetworkInterface**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

NetworkInterface XML Block

```
<NetworkInterface version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
```

```
   <id>          <!-- req, xs:integer, "1" -->       </id>
   <IPAddress/>   <!-- req -->
   <Discovery/>   <!-- opt -->
   <PPPoE /> <!-- opt -->
   <DDNS /> <!-- opt -->
</NetworkInterface>
```

8.2.3IPAddress

/Network/interfaces/*ID*/ipAddress	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the ip address of a particular network interface.
Query	None
Inbound Data	None
Success Return	**IPAddress**
PUT	**Administrator**
Description	It is used to update the ip address of a particular network interface.
Query	None
Inbound Data	**IPAddress**
Success Return	ResponseStaus **ResponseStatus**
Notes: If <addressingType> is dynamic, fields below it need not be provided. If <addressingType> is dynamic, a DHCP client is used for the device. If <addressingType> is static the device IP address is configured manually and the gateway and DNS fields are optional. <subnetMask> notation is "xxx.xxx.xxx.xxx".	

IPAddress XML Block

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

8.2.4Wireless

/Network/interfaces/*ID*/wireless	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the WIFI information of a wireless network interface.
Query	None
Inbound Data	None
Success Return	**Wireless**
PUT	**Administrator**
Description	It is used to update the WIFI information of a wireless network interface.
Query	None
Inbound Data	**Wireless**
Success Return	ResponseStaus **ResponseStatus**

Notes:

If the <securityMode> field is "WEP", the <WEP> block must be provided.

If the <securityMode> field is "WPA" or "WPA2-personal", the <WPA> block must be provided.

<channel> corresponds to an 802.11g wireless channel number or "auto" for autoconfiguration.

<wmmEnabled> enables 802.11e, QoS for IEEE 802.11 networks (Wi-Fi Multimedia)

<defaultTransmitKeyIndex> indicates which encryption key is used for WEP security.

<encryptionKey> is the WEP encryption key in hexadecimal format.

<sharedKey> is the pre-shared key used in WPA

Wireless XML Block

```
<Wireless version="1.0"   xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <enabled> <!-- req, xs:boolean --> </enabled>
  <wirelessNetworkMode>
    <!-- opt, xs:string, "infrastructure,adhoc" -->
  </wirelessNetworkMode>
  <channel> <!-- opt, xs:string, "1-14,auto" --> </channel>
  <ssid> <!-- opt, xs:string --> </ssid>
  <wmmEnabled> <!-- opt, xs:boolean --> </wmmEnabled>
  <WirelessSecurity> <!-- opt -->
    <securityMode>
      <!-- opt, xs:string, "disable,WEP,WPA-personal,WPA2-personal,WPA-RADIUS,
      WPA-enterprise,WPA2-enterprise" -->
    </securityMode>
    <WEP> <!-- dep, depends on <securityMode> -->
          <authenticationType>
          <!-- req, xs:string, "open,sharedkey,auto" -->
          </authenticationType>
          <defaultTransmitKeyIndex> <!-- req, xs:integer --> </defaultTransmitKeyIndex>
          <wepKeyLength> <!-- opt, xs:integer “64,128,152” --> </wepKeyLength>
          <wepKeyType><!-- opt, xs:string “HEX,ASICII” --> </wepKeyType>
          <EncryptionKeyList>
            <encryptionKey>
            <!-- req, xs: HexBinary string or ASICII string -->
            </encryptionKey>
          </EncryptionKeyList>
      </WEP>
      <WPA> <!-- dep, depends on <securityMode> -->
         <algorithmType> <!-- req, xs:string, “TKIP,AES,TKIP/AES”--> </algorithmType>
         <sharedKey> <!-- req, xs:string, pre-shared key used in WPA --> </sharedKey>
         <wpaKeyLength><!-- req, xs: integer, “8-63”--></wpaKeyLength>
      </WPA>
  </WirelessSecurity>
</Wireless>
```

8.2.5 DetectedWirelessList

/Network/interfaces/*ID*/detectedWirelessList	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get all detected wireless networks.
Query	None
Inbound Data	None
Success Return	detectedWirelessList

detectedWireless XML Block

```
<DetectedWirelessList version=“1.0”
xmlns="http://www.std-cgi.com/ver10/XMLSchema">
<DetectedWireless/>
</DetectedWirelessList>
```

8.2.6 DetectedWireless

/Network/interfaces/*ID*/detectedWirelessList/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a special detected wireless network.
Query	None

Inbound Data	None
Success Return	detectedWireless

detectedWirelessList XML Block

```
<DetectedWireless version="1.0"   xmlns="http://www.std-cgi.com/ver10/XMLSchema">
 <id>   <!-- req, xs:integer-->      </id>
 <wirelessNetworkMode>
   <!-- opt, xs:string, "infrastructure,adhoc" -->
 </wirelessNetworkMode>
 <channel> <!-- opt, xs:string, "1-14,auto" --> </channel>
 <ssid> <!-- req, xs:string --> </ssid>
 <speed> <!-- opt, xs:Integer, in Mbps--></speed>
 <signalStrength><!-- opt, xs:Integer,"0-100"></signalStrength>
 <securityMode>
    <!-- req, xs:string, "disable,WEP,WPA-personal,WPA2-personal,WPA-RADIUS,
    WPA-enterprise,WPA2-enterprise" -->
   </securityMode>
</DetectedWireless>
```

8.2.7Discovery

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

Discovery XML Block

```
<Discovery version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <UPnP>             <!-- req -->
      <enabled>       <!-- req, xs:boolean -->   </enabled>
   </UPnP>
   <Zeroconf>        <!-- opt -->
      <enabled>      <!-- req, xs:boolean -->   </enabled>
   </Zeroconf>
</Discovery>
```

8.2.8PPPoE

/Network/interfaces/*ID*/pppoe	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the PPPoE configuration of a particular network interface.
Query	None
Inbound Data	None
Success Return	**PPPoE**
PUT	**Administrator**
Description	It is used to update the PPPoE configuration of a particular network interface.
Query	None
Inbound Data	**PPPoE**
Success Return	ResponseStaus **ResponseStatus**
Notes: <password> is a write-only field.	

PPPoE XML Block

```
<PPPoE version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled>      <!-- req, xs:boolean -->   </enabled>
   <userName>      <!-- req, xs:string -->   </userName>
   <password>      <!-- wo, req, xs:string -->   </password>
   <dynamicIP>      <!--opt, xs:string -->   </dynamicIP>
</PPPoE>
```

8.2.9DDNS

/Network/interfaces/*ID*/ddns	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get DDNS configuration of a particular network interface.
Query	None
Inbound Data	None
Success Return	**DDNS**
PUT	**Administrator**

Description	It is used to update DDNS configuration of a particular network interface.
Query	None
Inbound Data	**DDNS**
Success Return	ResponseStaus **ResponseStatus**
Notes: When <provider> is “IPServer”, <serverIPAddress> is required. When <provider> is “DysDNS”, all fields are required except the <portNo>. When <provider> is “PeanutHall”, all fields are required except the <serverIPAddress> and <portNo>. <password> is a write-only field.	

DDNS XML Block

```
<DDNS version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <enabled>     <!-- req, xs:boolean --> </enabled>
   <provider>    <!-- req, xs:string, “IPServer, DynDNS, PeanutHall” -->    </provider>
   <serverIPAddress> <!—dep, xs:string --> </serverIPAddress>
   <portNo> <!-- dep, xs:integer --> </portNo>
   <domainName> <!-- dep, xs:string --> </domainName>
   <userName> <!-- dep, xs:string --> </userName>
   <password> <!-- wo, dep, xs:string --></password>
</DDNS>
```

8.2.10 NFSList

/Network/interfaces/*ID*/NFSList	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the configuration of NFSs for a particular network interface.
Query	None
Inbound Data	None
Success Return	**NFSList**
PUT	**Administrator**
Description	It is used to update the configuration of NFSs for a particular network interface.
Query	None
Inbound Data	**NFSList**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

NFSList XML Block

```
<NFSList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <NFS/>
</NFSList>
```

8.2.11 NFS

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

NFS XML Block

```
<NFS version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <NFSIPAddress>     <!-- req, xs:string -->   </NFSIPAddress>
  <NFSDirectory>     <!-- req, xs:string -->   </NFSDirectory>
</NFS>
```

8.2.12 Examples

Example: Getting the Network Settings

```
GET /Network/interfaces HTTP/1.1
…
HTTP/1.1 200 OK
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

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

```
  <NetworkInterface>
    <id>1</id>
    <IPAddress>
      <ipVersion>v4</ipVersion>
      <addressingType>static</addressingType>
      <ipAddress>172.6.64.7</ipAddress>
      <subnetMask>255.255.255.0</subnetMask>
      <DefaultGateway>
        <ipAddress>172.6.64.1</ipAddress>
      </DefaultGateway>
      <PrimaryDNS>
        <ipAddress>192.0.0.200</ipAddress>
      </PrimaryDNS>
    </IPAddress>
    <Discovery>
      <UPnP>
        <enabled>true</enabled>
      </UPnP>
      <Zeroconf>
        <enabled>true</enabled>
      </Zeroconf>
    </Discovery>
    <PPPoE>
      <enabled>true</enabled>
      <userName>std-cgi</userName>
    </PPPoE>
    <DDNS>
      <enabled>true</enabled>
      <provider>PeanutHall</provider>
      <domainName>std-cgi.vicp.net</domainName>
      <userName>std-cgi</userName>
    </DDNS>
  <NetworkInterface>
</NetworkInterfaceList>
```

Example: Setting the IP Address

```
PUT /Network/interfaces/1/ipAddress HTTP/1.1
…
Content-Type: application/xml; charset=“UTF-8”
Content-Length: xxx

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

```
    <ipVersion>v4</ipVersion>
    <addressingType>static</addressingType>
    <ipAddress>172.6.64.16</ipAddress>
    <subnetMask>255.255.255.0</subnetMask>
    <DefaultGateway>
        <ipAddress>172.6.64.1</ipAddress>
    </DefaultGateway>
    <PrimaryDNS>
        <ipAddress>192.0.0.200</ipAddress>
    </PrimaryDNS>
</IPAddress>

HTTP/1.1 200 OK
…
Content-Type: application/xml; charset=”UTF-8”
Content-Length:xxx

<?xml version=”1.0” encoding=”UTF-8”?>
<ResponseStatus version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
    <requestURL>/Network/interfaces/1/ipAddress</requestURL>
    <statusCode>1</statusCode>
    <statusString>OK</statusString>
</ResponseStatus>
```

8.3 IO

/IO	**Service v1.0**
GET	**Viewer**
Description	It is used to get the I/O ports information.
Query	None
Inbound Data	None
Success Return	**IOPortList**
Notes: The allocation of IDs between input and output ports must be unique.	

IOPortList XML Block

```
<IOPortList version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
    <IOInputPortList/>      <!-- opt -->
    <IOOutputPortList/>   <!-- opt -->
</IOPortList>
```

8.3.1Status

/IO/status	General Resource v1.0
GET	**Viewer**
Description	It is used to get the status of the I/O ports.
Query	None
Inbound Data	None
Success Return	**IOPortStatusList**
Notes: <ioPortID> refers to /IO/inputs/ID or /IO/outputs/ID. The port IDs are guaranteed to be unique across input and output ports. <ioState> indicates whether the input port is active or inactive. In most applications, a high signal is considered active.	

IOPortStatus XML Block

```
<IOPortStatusList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <IOPortStatus>     <!-- req -->
      <ioPortID>     <!-- req, xs:integer, "1, 2" -->                    </ioPortID>
      <ioPortType>   <!-- req, xs:string, "input,output" -->          </ioPortType>
      <ioState>      <!-- req, xs:string, "active,inactive" -->      </ioState>
   </IOPortStatus>
</IOPortStatusList>
```

8.3.2Inputs

/IO/inputs	General Resource v1.0
GET	**Viewer**
Description	It is used to get the Input ports information.
Query	None
Inbound Data	None
Success Return	**IOInputPortList**
Notes: IO inputs are hardwired, meaning that the inputs are statically allocated by the device and cannot be created or deleted.	

IOInputPortList XML Block

```
<IOInputPortList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <IOInputPort/>     <!-- opt -->
</IOInputPort>
```

8.3.3Input

/IO/inputs/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get particular input port information.
Query	None
Inbound Data	None
Success Return	**IOInputPort**
PUT	**Operator**
Description	It is used to update particular input port information.
Query	None
Inbound Data	**IOInputPort**
Success Return	ResponseStaus **ResponseStatus**
Notes: <triggering> indicates the signal conditions to trigger the input port. High/Low will continuously trigger for the duration of high/low input signal.	

IOInputPort XML Block

```
<IOInputPort version="1.0"    xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>          <!-- req, xs:integer -->                           </id>
   <triggering>   <!-- req, xs:string, "high,low" -->    <triggering>
</IOInputPort>
```

8.3.4Input status

/IO/inputs/*ID*/status	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the status of a particular input port.
Query	None
Inbound Data	None
Success Return	**IOPortStatus**
Notes: See /IO/status for an explanation of the fields.	

8.3.5Outputs

/IO/outputs	**General Resource v1.0**
GET	**Viewer**

Description	It is used to get the output ports information.
Query	None
Inbound Data	None
Success Return	**IOOutputPortList**
Notes: IO outputs are hardwired, meaning that the outputs are statically allocated by the device and cannot be created or deleted.	

IOOutputPortList XML Block

```
<IOOutputPortList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <IOOutputPort/>    <!-- opt -->
</IOOutputPort>
```

8.3.6Output

/IO/outputs/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get particular output port information.
Query	None
Inbound Data	None
Success Return	**IOOutputPort**
PUT	**Operator**
Description	It is used to update particular output port information.
Query	None
Inbound Data	**IOOutputPort**
Success Return	ResponseStaus **ResponseStatus**
Notes: <PowerOnState> defines the output port configuration when the device is powered on. <defaultState> is the default output port signal when it is not being triggered. <outputState> is the output port signal when it is being triggered. Pulse will cause the output port to send a signal (opposite of the <defaultState>) for a duration specified by the <pulseDuration> tag. <pulseDuration> is the duration of a output port signal when it is being triggered. It must be provided if the <outputState> is "pulse".	

IOOutputPort XML Block

```
<IOOutputPort version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>              <!-- req, xs:integer, "2" -->              </id>
   <PowerOnState>      <!-- req -->
      <defaultState>    <!—ro, req, xs:string, "high,low" -->     </defaultState>
      <outputState>     <!—ro, req, xs:string, "high,low,pulse" -->   </outputState>
```

```
    <pulseDuration>   <!-- dep, xs:integer, milliseconds --> </pulseDuration>
  </PowerOnState>
</IOOutputPort>
```

8.3.7 Output status

/IO/outputs/*ID*/status	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the status of a particular output port.
Query	None
Inbound Data	None
Success Return	**IOPortStatus**
Notes: See /IO/status for an explanation of the fields.	

8.3.8 Output trigger

/IO/outputs/*ID*/trigger	**General Resource v1.0**
PUT	**Operator**
Description	It is used to manually trigger a particular output port.
Query	None
Inbound Data	**IOPortData**
Success Return	ResponseStaus **ResponseStatus**
Notes: Note that the ID used here MUST correspond to the ID in /IO/outputs/ID. The IO output port is toggled to a high or low signal accordingly.	

IOPortData XML Block

```
<IOPortData xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <outputState>     <!-- req, xs:string, "high,low" -->   </outputState>
</IOPortData>
```

8.4 Video

/Video	**Service v1.0**
Notes:	

8.4.1Input

/Video/inputs	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the video inputs configuration on an IP media device.
Query	None
Inbound Data	None
Success Return	**VideoInput**
Notes: An IP media device may contain a set of video inputs. These inputs are hardwired by the device, meaning that the IDs can be discovered but not created or deleted.	

VideoInput XML Block

```
<VideoInput version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <VideoInputChannelList/>   <!-- opt -->
</VideoInput>
```

8.4.2Input channels

/Video/inputs/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the video input channels configuration on an IP media device.
Query	None
Inbound Data	None
Success Return	**VideoInputChannelList**
Notes: Since video input channels are resources that are defined by the hardware configuration of the device, they cannot be created or deleted.	

VideoInputChannelList XML Block

```
<VideoInputChannelList version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <VideoInputChannel/>      <!-- opt -->
</VideoInputChannelList>
```

8.4.3Input channel

/Video/inputs/channels/*ID*	**General Resource v1.0**

GET	Viewer
Description	It is used to get a particular video input channel configuration on an IP media device.
Query	None
Inbound Data	None
Success Return	**VideoInputChannel**
PUT	**Operator**
Description	It is used to update a particular video input channel configuration on an IP media device.
Query	None
Inbound Data	**VideoInputChannel**
Success Return	ResponseStaus **ResponseStatus**
Notes: <powerLineFrequencyMode> is used to adjust/correct video image based on different power frequencies. <whiteBalanceMode> indicates the white balance operational mode. <gainLevel> indicates the gain level percentage value. 0 is low gain, 100 is high gain.	

VideoInputChannel XML Block

```
<VideoInputChannel version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <id>               <!-- req, xs:integer -->            </id>
  <powerLineFrequencyMode>   <!-- opt, xs:string “50hz, 60hz” -->
  </powerLineFrequencyMode>
  <whiteBalanceMode>
    <!-- opt, xs:string, “manual,auto,indoor/incandescent” -->
  </whiteBalanceMode>
  <gainLevel>            <!-- opt, xs:integer, 0..100-->     </gainLevel>
  <brightnessLevel>        <!-- opt, xs:integer, 0..100 -->     </brightnessLevel>
  <contrastLevel>        <!-- opt, xs:integer, 0..100 -->     </contrastLevel>
  <saturationLevel>        <!-- opt, xs:integer, 0..100 -->     </saturationLevel>
  <DayNightFilter>         <!-- opt -->
    <dayNightFilterType>
      <!-- opt, xs:string, “day,night,auto” -->
    </dayNightFilterType>
  </DayNightFilter>
<VideoInputChannel>
```

8.4.4Input channel overlay texts

/Video/inputs/channels/*ID*/overlays/text	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the text overlays configuration for a video input

	channel.
Query	None
Inbound Data	None
Success Return	**TextOverlayList**
PUT	**Operator**
Description	It is used to update the text overlays configuration for a video input channel.
Query	None
Inbound Data	**TextOverlayList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a text overlay for a video input channel.
Query	None
Inbound Data	**TextOverlay**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the text overlays configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: A set of text overlays is managed. They are composited over the video signal in increasing ID-order.	

TextOverlayList XML Block

```
<TextOverlayList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <TextOverlay/>     <!-- opt -->
</TextOverlayList>
```

8.4.5Input channel overlay text

/Video/inputs/channels/*ID*/overlays/text/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular text overlay configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**TextOverlay**
PUT	**Operator**
Description	It is used to update a particular text overlay configuration for a video

	input channel.
Query	None
Inbound Data	**TextOverlay**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular text overlay configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <posY> value is a multiple of 16.	

TextOverlay XML Block

```
<TextOverlay version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>               <!-- req, xs:integer, "1-4" -->          </id>
   <enabled>             <!-- req, xs:boolean -->          </enabled>
   <posX>      <!-- req, xs:integer -->          </posX>
   <posY>      <!-- req, xs:integer -->          </posY>
   <message>      <!-- req, xs:string -->      </message>
</TextOverlay>
```

8.4.6Input channel privacyMask

/Video/inputs/channels/*ID*/privacyMask	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the privacy masking configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**PrivacyMask**
PUT	**Operator**
Description	It is used to update the privacy masking configuration for a video input channel.
Query	None
Inbound Data	**PrivacyMask**
Success Return	ResponseStaus **ResponseStatus**
Notes: Privacy masking can be enabled and the region list configured per channel.	

PrivacyMask XML Block

```
<PrivacyMask version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled>              <!-- req, xs:boolean -->        </enabled>
   <PrivacyMaskRegionList/>   <!-- opt -->
</PrivacyMask>
```

8.4.7Input channel privacyMask regions

/Video/inputs/channels/*ID*/privacyMask/regions	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the privacy mask regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**PrivacyMaskRegionList**
PUT	**Operator**
Description	It is used to update the privacy mask regions configuration for a video input channel.
Query	None
Inbound Data	**PrivacyMaskRegionList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a privacy mask region for a video input channel.
Query	None
Inbound Data	**PrivacyMaskRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the privacy mask regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Privacy masking consists of a set of regions that are combined to grey or black out areas of a video input.	

PrivacyMaskRegionList XML Block

```
<PrivacyMaskRegionList version="1.0"
   xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <PrivacyMaskRegion/>   <!-- opt -->
</PrivacyMaskRegionList>
```

8.4.8Input channel privacyMask region

/Video/inputs/channels/ID/privacyMask/regions/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular privacy mask region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**PrivacyMaskRegion**
PUT	**Operator**
Description	It is used to update a particular privacy mask region configuration for a video input channel.
Query	None
Inbound Data	**PrivacyMaskRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular privacy mask region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Region coordinates are dependent on video resolution. Only support the rectangular region which will be “drawn” from four coordinates. The four points is clockwise direction, and the beginning point is the low-left point. Ordering of <PrivacyMaskRegion> blocks is insignificant.	

PrivacyMaskRegion XML Block

```
<PrivacyMaskRegion version=“1.0”
 xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <id>              <!-- req, xs:integer, “1-4” -->          </id>
  <RegionCoordinatesList>   <!-- req -->
    <RegionCoordinates>   <!-- req -->
      <positionX>        <!-- req, xs:integer;coordinate -->      </positionX>
      <positionY>        <!-- req, xs:integer;coordinate   -->     </positionY>
    </RegionCoordinates>
  </RegionCoordinatesList>
</PrivacyMaskRegion>
```

8.4.9Input channel shelterAlarm

/Video/inputs/channels/*ID*/shelterAlarm	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the shelter alarm configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**ShelterAlarm**
PUT	**Operator**
Description	It is used to update the shelter alarm configuration for a video input channel.
Query	None
Inbound Data	**ShelterAlarm**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

ShelterAlarm XML Block

```
<ShelterAlarm version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema"">
   <enabled> <!-- req, xs:boolean --> </enabled>
   <ShelterAlarmRegionList/> <!-- opt -->
</ShelterAlarm>
```

8.4.10 Input channel shelterAlarm regions

/Video/inputs/channels/*ID*/shelterAlarm/regions	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the shelter alarm regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**ShelterAlarmRegionList**
PUT	**Operator**
Description	It is used to update the shelter alarm regions configuration for a video input channel.
Query	None
Inbound Data	**ShelterAlarmRegionList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a shelter alarm region for a video input channel.

Query	None
Inbound Data	**ShelterAlarmRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the shelter alarm regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

ShelterAlarmRegionList XML Block

```
<ShelterAlarmRegionList version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <sensitivityLevel>       <!-- req -->
       <!-- req, xs:string, "low, middle, high"-->
    </sensitivityLevel>
    <ShelterAlarmRegion /> <!-- opt -->
</ShelterAlarmRegionList>
```

8.4.11 Input channel shelterAlarm region

/Video/inputs/channels/*ID*/shelterAlarm/regions/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular shelter alarm region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**ShelterAlarmRegion**
PUT	**Operator**
Description	It is used to update a particular shelter alarm region configuration for a video input channel.
Query	None
Inbound Data	**ShelterAlarmRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular shelter alarm region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:

Region coordinates are dependent on video resolution. Only support the rectangular region which will be "drawn" from four coordinates. The four points is clockwise direction, and the beginning point is the low-left point.
Ordering of <ShelterAlarmRegion> blocks is insignificant.

ShelterAlarmRegion XML Block

```
<ShelterAlarmRegion version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id>              <!-- req, xs:integer, "1" -->           </id>
  <RegionCoordinatesList>   <!-- req -->
    <RegionCoordinates>   <!-- req -->
      <positionX>       <!-- req, xs:integer;coordinate -->      </positionX>
      <positionY>       <!-- req, xs:integer;coordinate   -->     </positionY>
    </RegionCoordinates>
  </RegionCoordinatesList>
</ShelterAlarmRegion>
```

8.4.12 Input channel osdDatetime

/Video/inputs/channels/*ID*/osdDatetime	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the OSD configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**OsdDatetime**
PUT	**Operator**
Description	It is used to update the OSD configuration for a video input channel.
Query	None
Inbound Data	**OsdDatetime**
Success Return	ResponseStaus **ResponseStatus**

Notes:

<posY> value is a multiple of 16.
<type> is the type of the year month day and should be:
  0: XXXX-XX-XX Y-M-D
  1: XX-XX-XXXX M-D-Y
  4: XX-XX-XXXX D-M-Y
<displayWeek> means display the week or not.
<attribute> is the configuration of the OSD, the value should be:
1: transparent, flash
2: transparent, not flash

3: not transparent, flash 4: not transparent, not flash

OsdDatetime XML Block

```
<OsdDatetime version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled> <!-- req, xs:boolean -->    </enabled>
   <posX> <!-- req, xs:integer;coordinate --> </posX>
   <posY> <!-- req, xs:integer;coordinate --> </posY>
   <type> <!-- req, xs:integer --> </type>
   <displayWeek> <!-- req, xs:boolean --> </displayWeek>
   <attribute> <!-- req, xs:integer --> </attribute>
</OsdDatetime>
```

8.5 Audio

/Audio	**Service v1.0**
Notes:	

8.5.1Channels

/Audio/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the audio channels configuration on an IP media device.
Query	None
Inbound Data	None
Success Return	**AudioChannelList**
Notes: Since inputs are resources that are defined by the hardware configuration of the device, audio channels cannot be created or deleted.	

AudioChannelList XML Block

```
<AudioChannelList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <AudioChannel/>    <!-- opt -->
</AudioChannelList>
```

8.5.2Channel

/Audio/channels/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular audio channel configuration on an IP media device.
Query	None
Inbound Data	None
Success Return	**AudioChannel**
Notes: <audioMode> is the duplex mode for audio transmission between the client and media device. <microphoneVolume> Volume control percentage for device microphone.. <speakerVolume> Volume control percentage for device speaker.	

AudioChannel XML Block

```
<AudioChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>          <!-- req, xs:integer, "11,12" -->                  </id>
   <enabled>      <!-- req, xs:boolean -->                           </enabled>
   <audioMode>
      <!-- req, xs:string, "talkonly, talkandlisten" -->
   </audioMode>
   <microphoneEnabled>   <!-- req, xs:boolean -->                    </microphoneEnabled>
   <microphoneSource>      <!-- req, xs:string, "external" -->      </microphoneSource>
   <microphoneVolume>   <!--req, xs:integer, 0…100 -->        </microphoneVolume>
   <speakerEnabled>      <!-- req, xs:boolean -->                    </speakerEnabled>
   <speakerVolume>      <!-- req, xs:integer,0…100 -->               </speakerVolume>
</AudioChannel>
```

8.6 Two way audio

/TwowayAudio	**Service v1.0**
Notes:	

8.6.1Open

/TwowayAudio/open	**General Resource v1.0**
PUT	**Operator**
Description	It is used to open intercom.

Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.6.2Close

/TwowayAudio/close	**General Resource v1.0**
PUT	**Operator**
Description	It is used to close intercom.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.6.3Send data

/TwowayAudio/sendData	**General Resource v1.0**
PUT	**Operator**
Description	It is used to send the intercom data.
Query	None
Inbound Data	**TwowayAudio Data**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

Example：

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

8.6.4Receive data

/TwowayAudio/receiveData	**General Resource v1.0**

GET	Operator
Description	It is used to receive the intercom data.
Query	None
Inbound Data	None
Success Return	**TwowayAudio 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…
```

8.7 Serial

/Serial	Service v1.0
Notes: Serial port service.	

8.7.1Ports

/Serial/ports	General Resource v1.0
GET	**Viewer**
Description	It is used to get the list of serial ports supported by the device.
Query	None
Inbound Data	None
Success Return	**SerialPorList**
Notes: Since serial ports are resources that are defined by the hardware configuration of the device, they cannot be created or deleted.	

SerialPortList XML Block

```
<SerialPortList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <SerialPort/>    <!-- opt -->
```

```
</SerialPortList>
```

8.7.2Port

/Serial/ports/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the configuration of a serial port supported by the device.
Query	None
Inbound Data	None
Success Return	**SerialPort**
PUT	**Operator**
Description	It is used to update the configuration of a serial port supported by the device.
Query	None
Inbound Data	**SerialPort**
Success Return	ResponseStaus **ResponseStatus**
Notes: <serialPortType> set the type of port; RS232, RS485. When <id> value is 1, <serialPortType> value is “RS485”. When <id> value is 3, <serialPortType> value is “RS232”. <serialPortType> value can not set directly.	

SerialPort XML Block

```
<SerialPort version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>            <!-- req, xs:integer, "1, 3" -->                          </id>
   <enabled>         <!-- ro, req, xs:boolean -->                            </enabled>
   <serialPortType>   <!-- req, xs:string, "RS485, RS232" -->          </serialPortType>
   <baudRate>        <!-- req, xs:integer -->                            </baudRate>
   <dataBits>       <!-- req, xs:integer -->                            </dataBits>
   <parityType>      <!-- req, xs:string, "none,even,odd" --> </parityType>
   <stopBits>       <!-- req, xs:string, "1,1.5,2" -->                 </stopBits>
</SerialPort>
```

8.7.3Command

/Serial/ports/*ID*/command	**General Resource v1.0**
PUT	**Operator**
Description	It is used to send a command to a serial port.
Query	chainNo
Inbound Data	**SerialCommand** or **Raw Data**

Success Return	ResponseStaus **ResponseStatus**
Notes: If the IP device is an analog-to-digital encoder and is connected to analog PTZ-enabled camera(s), it is the device's responsibility to relay the request to the appropriate serial interface based on the <chainNo> tag or query string. If the IP device is itself a PTZ-enabled digital camera, it is the device's responsibility to address the correct serial interface for the corresponding PTZ command. The serial command can either be encapsulated in the <command> field, in which case the data should be encoded in hexadecimal notation, or the data can be uploaded directly as the HTTP payload, in which case the content type should be application/octet-stream.	

SerialCommand XML Block

```
<SerialCommand version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <chainNo>      <!-- req, xs:string -->   </chainNo>
   <command>      <!-- req, xs:string, bytes in hexadecimal -->   </command>
</SerialCommand>
```

8.7.4Transparent channel open

/Serial/ports/*ID*/transChanOpen	**General Resource v1.0**
PUT	**Operator**
Description	It is used to open the transparent channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Only support RS485 transparent channel, so *ID* value in the Resource_URI can only be 1.	

8.7.5Transparent channel close

/Serial/ports/*ID*/transChanClose	**General Resource v1.0**
PUT	**Operator**
Description	It is used to close the transparent channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Only support RS485 transparent channel, so *ID* value in the Resource_URI can only be 1.	

8.7.6Transparent channel send data

/Serial/ports/*ID*/transChanSendData	**General Resource v1.0**
PUT	**Operator**
Description	It is used to send data on the transparent channel.
Query	None
Inbound Data	**Raw Data**
Success Return	ResponseStaus **ResponseStatus**
Notes: Only support RS485 transparent channel, so *ID* value in the Resource_URI can only be 1.	

Example :

```
PUT /Serial/ports/1/transChanSendData HTTP/1.1
…
Content-Type: application/binary; charset="UTF-8"
Content-Length: xxx
\r\n
Raw Data…
```

8.7.7Transparent channel receive data

/Serial/ports/*ID*/transChanRecvData	**General Resource v1.0**
GET	**Operator**
Description	It is used to receive data on the transparent channel.
Query	None
Inbound Data	None
Success Return	**Raw Data**
Notes: Only support RS485 transparent channel, so *ID* value in the Resource_URI can only be 1.	

Example :

```
GET /Serial/ports/1/transChanRecvData HTTP/1.1
…

HTTP/1.1 200 OK
…
Content-Type: application/binary; charset="UTF-8"
Content-Length: xxx
```

```
\r\n
Raw Data…
```

8.8 Security

/Security	Service v1.0
Notes:	

8.8.1Users

/Security/users	General Resource v1.0
GET	**Viewer**
Description	It is used to get the user list for the device.
Query	None
Inbound Data	None
Success Return	**UserList**
PUT	**Administrator**
Description	It is used to update the user list for the device.
Query	None
Inbound Data	**UserList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Administrator**
Description	It is used to add a user for the device.
Query	None
Inbound Data	**User**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete the user list for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: A default user account, “admin”, must be provided. Its default password is “12345”. It has an Administrator user level, and must not be deleted. Passwords can only be uploaded - they are never revealed during GET operations.	

UserList XML Block

```
<UserList version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <User/>      <!-- opt -->
```

```
</UserList>
```

8.8.2User

/Security/users/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular user configuration for the device.
Query	None
Inbound Data	None
Success Return	**User**
PUT	**Administrator**
Description	It is used to update a particular user configuration for the device.
Query	None
Inbound Data	**User**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete a particular user for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <id> of “admin” account is 1. “admin” account must not be deleted. <password> is a write-only field.	

User XML Block

```
<User version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <id>               <!-- req, xs:integer, “1-16” -->         </id>
   <userName>           <!-- req, xs:string -->          </userName>
   <password>           <!-- wo, req, xs:string -->       </password>
   <priority> <!-- opt, xs:string; “low, middle, high” --> </priority>
   <ipAddress> <!-- opt, xs:string --> </ipAddress>
   <macAddress> <!-- opt, xs:string --> </macAddress>
   <userLevel> <!-- opt, xs:string, “Administrator, Operator, Viewer” --> </userLevel>
</User>
```

8.8.3adminAccess

/Security/adminAccess	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get administrative access protocol for the device.

Query	None
Inbound Data	None
Success Return	**AdminAccessProtocol**
PUT	**Administrator**
Description	It is used to update administrative access protocol for the device.
Query	None
Inbound Data	**AdminAccessProtocol**
Success Return	ResponseStaus **ResponseStatus**
Notes: <protocol> is the protocol name for admin access, i.e. “HTTP”, “HTTPS”, etc.	

AdminAccessProtocol XML Block

```
<AdminAccessProtocol version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <protocol> <!-- req, xs:string; "HTTP, HTTPS" --> </protocol>
    <portNo> <!-- req, xs:integer -->    </portNo>
    <netClientPort><!-- req, xs:integer --></netClientPort>
</AdminAccessProtocol>
```

8.9 Streaming

/Streaming	**Service v1.0**
Notes:	

8.9.1Status

/Streaming/status	**General Resource v1.0**
GET	**Administrator**
Description	It is used to get a device streaming status.
Query	None
Inbound Data	None
Success Return	**StreamingStatus**
Notes: This command accesses the status of all device streaming sessions.	

StreamingStatus XML Block

```
<StreamingStatus version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <totalStreamingSessions>          <!-- req, xs:integer -->    </totalStreamingSessions>
```

```
  <StreamingSessionStatusList/>       <!-- dep, only if there are sessions -->
</StreamingStatus>
```

8.9.2Channels

/Streaming/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the properties of streaming channels for the device.
Query	None
Inbound Data	None
Success Return	**StreamingChannelList**
PUT	**Administrator**
Description	It is used to update the properties of streaming channels for the device.
Query	None
Inbound Data	**StreamingChannelList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Administrator**
Description	It is used to add a streaming channel for the device.
Query	None
Inbound Data	**StreamingChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete the list of streaming channels for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Streaming channels may be hardwired, or it may be possible to create multiple streaming channels per input if the device supports it. To determine whether it is possible to dynamically create streaming channels, check the defined HTTP methods in /Streaming/channels/description.	

StreamingChannelList XML Block

```
<StreamingChannelList version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <StreamingChannel/>    <!-- opt -->
</StreamingChannelList>
```

8.9.3Channel

/Streaming/channels/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the properties of a particular streaming channel for the device.
Query	None
Inbound Data	None
Success Return	**StreamingChannel**
PUT	**Administrator**
Description	It is used to update the properties of a particular streaming channel for the device.
Query	None
Inbound Data	**StreamingChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete a particular streaming channel for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:

When <id> value is 1, stream is the main stream. When <id> is 2, stream is the sub stream.
<ControlProtocolList> identifies the control protocols that are valid for this type of streaming.
<Unicast> is for direct unicast streaming.
<Multicast> is for direct multicast streaming.
<sourcePortNo> is the unicast source port parameter for the outbound video and audio streams, and the specific port number is device-dependant.
<destPortNo> is the multicast destination port parameter for the outbound video and audio streams, and the specific port number is device-dependant.
<videoInputChannelID> refers to /Video/inputs/channels/*ID*.
<audioInputChannelID> refers to /Audio/channels/*ID*. It must be configured as an input channel.
<audioResolution> is the resolution for the outbound audio stream in bits.

StreamingChannel XML Block

```
<StreamingChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id>          <!-- req, xs:integer, "1, 2" -->       </id>
  <channelName>   <!-- ro, req, xs:string -->       </channelName>
  <enabled>       <!-- ro, req, xs:boolean -->       </enabled>
  <Transport>       <!-- req -->
```

```
    <rtspPortNo>          <!-- opt, xs:integer -->      </rtspPortNo>
    <maxPacketSize>          <!-- ro, opt, xs:integer -->      </maxPacketSize>
    <sourcePortNo>      <!-- opt, xs:integer -->      </sourcePortNo>
    <ControlProtocolList>      <!-- req -->
      <ControlProtocol>      <!-- opt -->
        <streamingTransport>
          <!-- ro, req, xs:string, "RTSP" -->
        </streamingTransport>
      </ControlProtocol>
    </ControlProtocolList>
    <Unicast>               <!-- opt -->
      <enabled>            <!-- ro, req, xs:boolean, "true"-->         </enabled>
    </Unicast>
    <Multicast>               <!-- opt -->
      <enabled>            <!-- ro, req, xs:boolean, "true" -->         </enabled>
      <destIPAddress>      <!-- opt, xs:string -->         </destIPAddress>
      <destPortNo>      <!-- opt, xs:integer -->         </destPortNo>
    </Multicast>
  </Transport>
  <Video>
    <enabled>               <!-- ro, req, xs:boolean, "true" -->         </enabled>
    <videoInputChannelID>      <!-- req, xs:integer -->         </videoInputChannelID>
    <videoCodecType>
      <!-- ro, opt, xs:string, "H.264,MJPEG" -->
    </videoCodecType>
    <videoScanType>     <!-- ro, opt, xs:string, "progressive" -->     </videoScanType>
    <videoResolutionWidth>      <!-- req, xs:integer -->         </videoResolutionWidth>
    <videoResolutionHeight>    <!--  req,  xs:integer  -->          </videoResolutionHeight>
    <videoQualityControlType>
      <!-- req, xs:string, "CBR,VBR" -->
    </videoQualityControlType>
    <constantBitRate>   <!-- opt, xs:integer, in kbps -->         </constantBitRate>
    <fixedQuality>     <!-- opt, xs:integer, percentage, "0-100" -->      </fixedQuality>
    <maxFrameRate>     <!-- req, xs:integer, maximum frame rate x100 -->
    </maxFrameRate>
    <keyFrameInterval>   <!-- opt, xs:integer, milliseconds -->   </keyFrameInterval>
    <BPFrameInterval> <!-- opt, xs:integer --> </BPFrameInterval>
    <mirrorStatus> <!-- opt, xs:string ,"OFF,UpToDown,LeftToRight"--> </mirrorStatus>
    <rotationDegree><!-- opt, xs: integer,"0,180 "--> </rotationDegree>
    <snapShotImageType><!-- ro, opt, xs:string, "JPEG" -->   </snapShotImageType>
  </Video>
  <Audio>
    <enabled>         <!-- ro, req, xs:boolean, "true,false" -->         </enabled>
    <audioInputChannelID>      <!-- ro, req, xs:integer -->         </audioInputChannelID>
```

```
      <audioCompressionType>
         <!-- ro,opt, xs:string, "G.711ulaw" -->
      </audioCompressionType>
   </Audio>
</StreamingChannel>
```

Example: Getting Streaming Channel Properties

The following is an example of a GET on the streaming parameters of a particular channel that has been preconfigured by the IP media device. Depending on the device, some streaming channels may be already preconfigured or the device while other may require that channels be manually configured before use.

```
GET /Streaming/channels/1 HTTP/1.1
…
HTTP/1.1 200 OK
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<StreamingChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>1</id>
   <channelName>Input 1 H.264</channelName>
   <enabled>true</enabled>
   <Transport>
      <rtspPortNo>554</rtspPortNo>
      <maxPacketSize>1000</maxPacketSize>
      <sourcePortNo>8200</sourcePortNo>
      <ControlProtocolList>
         <ControlProtocol>
            <streamingTransport>RTSP</streamingTransport>
         </ControlProtocol>
      </ControlProtocolList>
      <Unicast>
         <enabled>true</enabled>
      </Unicast>
      <Multicast>
         <enabled>true</enabled>
         <destIPAddress>224.16.74.1</destIPAddress>
         <destPortNo>8600</destPortNo>
      </Multicast>
   </Transport>
   <Video>
      <enabled>true</enabled>
      <videoInputChannelID>1</videoInputChannelID>
```

```
    <videoCodecType>H.264</videoCodecType>
    <videoScanType>progressive</videoScanType>
    <videoResolutionWidth>640</videoResolutionWidth>
    <videoResolutionHeight>480</videoResolutionHeight>
    <videoQualityControlType>CBR</videoQualityControlType>
    <constantBitRate>3072</constantBitRate>
    <fixedQuality>80</fixedQuality>
    <maxFrameRate>2500</maxFrameRate>
    <keyFrameInterval>25</keyFrameInterval>
    <BPFrameInterval>0</BPFrameInterval>
    <mirrorStatus>OFF</mirrorStatus>
    <rotationDegree>180</rotationDegree>
    <snapShotImageType>JPEG</snapShotImageType>
  </Video>
  <Audio>
    <enabled>true</enabled>
    <audioInputChannelID>11</audioInputChannelID>
    <audioCompressionType>G.711ulaw</audioCompressionType>
  </Audio>
</StreamingChannel>
```

Example: Getting Streaming Capabilities

```
GET /Streaming/channels/1/capabilities HTTP/1.1
…
HTTP/1.1 200 OK
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

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

<StreamingChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id opt="1,2">1</id>
  <channelName min="0" max="64">Input 1 H.264</channelName>
  <enabled opt="true">true</enabled>
  <Transport>
    <rtspPortNo min="0" max="65535" def="554">554</rtspPortNo>
    <maxPacketSize opt="1000">1000</maxPacketSize>
    <sourcePortNo min="0" max="65535" def="8200">8200</sourcePortNo>
    <ControlProtocolList>
      <ControlProtocol>
        <streamingTransport opt="RTSP">RTSP</streamingTransport>
      </ControlProtocol>
    </ControlProtocolList>
    <Unicast>
      <enabled opt="true" def="true">true</enabled>
    </Unicast>
    <Multicast>
      <enabled opt="true" def="true">true</enabled>
      <destIPAddress min="8" max="16">224.16.74.1</destIPAddress>
      <destPortNo min="0" max="65535" def="8600">8600</destPortNo>
    </Multicast>
  </Transport>
  <Video>
    <enabled opt="true">true</enabled>
    <videoInputChannelID opt="1">1</videoInputChannelID>
    <videoCodecType opt="H.264,MJPEG">H.264</videoCodecType>
    <videoScanType opt="progressive">progressive</videoScanType>
    <videoResolutionWidth opt="640*480">640</videoResolutionWidth>
    <videoResolutionHeight opt="640*480">480</videoResolutionHeight>
    <videoQualityControlType opt="CBR,VBR">CBR</videoQualityControlType>
    <constantBitRate min="32" max="4000">3072</constantBitRate>
    <fixedQuality opt="1,20,40,60,80,100">80</fixedQuality>
    <maxFrameRate
      opt="2500,2200,2000,1800,1600,1500,1200,1000,800,600,400,200,100,50,25,
      12,6">2500</maxFrameRate>
    <keyFrameInterval min="1", max="400">25</keyFrameInterval>
    <BPFrameInterval opt="0, 1, 2">0</BPFrameInterval>
    <mirrorStatus opt="OFF,UpToDown,LeftToRight">OFF</mirrorStatus>
    <rotationDegree opt="0,180">180</rotationDegree>
    <snapShotImageType opt="JPEG" def="JPEG">JPEG</snapShotImageType>
  </Video>
  <Audio>
    <enabled opt="true,false">true</enabled>
    <audioInputChannelID opt="11,12">11</audioInputChannelID>
    <audioCompressionType opt="G.711ulaw">G.711ulaw</audioCompressionType>
  </Audio>
</StreamingChannel>
```

8.9.4Channel status

/Streaming/channels/*ID*/status	**General Resource v1.0**
GET	**Administrator**
Description	It is used to get the list of streaming sessions associated with a particular channel.
Query	None

Inbound Data	None
Success Return	**StreamingSessionStatusList**
Notes:	

StreamingSessionStatusList XML Block

```
<StreamingSessionStatusList version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <StreamingSessionStatus>
    <clientAddress>   <!-- req -->
      <ipAddress>      <!-- req, xs:string -->             </ipAddress>
    </clientAddress>
  </StreamingSessionStatus>
</StreamingSessionStatusList>
```

8.9.5Picture

/Streaming/channels/*ID*/picture	**General Resource v1.0**
GET	**Operator**
Description	It is used to get a snapshot of the current image.
Query	videoResolutionWidth videoResolutionHeight snapShotImageType
Inbound Data	None
Success Return	**Picture over HTTP**

Notes:

All devices must support <snapShotImageType> of "JPEG".
Only support the main stream channel snapshot.
To determine the format of the picture returned, either the parameters in <Video> or the query string values are used, or, if the Accept: header field is present in the request and the server supports it, the picture is returned in that format.
For supported values, query /Streaming/channels/*ID*/picture/capabilities.
Examples:

```
  GET /Streaming/channels/1/picture?snapShotImageType=JPEG
  …

  GET /Streaming/channels/1/picture
  Accept: image/jpeg
  …
```

8.9.6Request keyframe

/Streaming/channels/*ID*/requestKeyFrame	**General Resource v1.0**
PUT	**Operator**
Description	It is used to request that the device issue a key frame on a particular channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: The key frame that is issued should include everything necessary to initialize a video decoder, i.e. parameter sets for H.264.	

8.10 Motion Detection

/MotionDetection	**Service v1.0**
GET	**Viewer**
Description	It is used to get the motion detection configuration for all video input channels.
Query	None
Inbound Data	None
Success Return	**MotionDetectionList**
Notes: If motion detection is supported by the device, a motion detection ID will be allocated for each video input channel ID. The motion detection ID must correspond to the video input channel ID.	

MotionDetectionList XML Block

```
<MotionDetectionList version=“1.0”
  xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
    <MotionDetection/>          <!-- opt -->
</MotionDetectionList>
```

8.10.1 One channel motion detection

/MotionDetection/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the motion detection configuration for a video input channel.

Query	None
Inbound Data	None
Success Return	**MotionDetection**
PUT	**Operator**
Description	It is used to udpate the motion detection configuration for a video input channel.
Query	None
Inbound Data	**MotionDetection**
Success Return	ResponseStaus **ResponseStatus**
Notes: Note that the ID used here MUST correspond to the video input ID. The interface supports grid-based motion detection. Grid-based motion detect divides the image into a set of fixed “bins” that delimit the motion detection area boundaries.	

MotionDetection XML Block

```
<MotionDetection version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <id>              <!-- req, xs:integer -->                  </id>
   <enabled>         <!-- req, xs:boolean -->                  </enabled>
   <regionType>      <!-- ro, req, xs:string, “grid” -->       </regionType>
   <Grid>            <!-- req -->
      <rowGranularity>      <!-- ro, req, xs:integer -->   </rowGranularity>
      <columnGranularity>   <!-- ro, req, xs:integer -->   </columnGranularity>
   </Grid>
   <MotionDetectionRegionList/>   <!-- req -->
</MotionDetection>
```

8.10.2 Motion detection regions

/MotionDetection/*ID*/regions	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the motion detection regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**MotionDetectionRegionList**
PUT	**Operator**
Description	It is used to update the motion detection regions configuration for a video input channel.
Query	None
Inbound Data	**MotionDetectionRegionList**

Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a motion detection region for a video input channel.
Query	None
Inbound Data	**MotionDetectionRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the motion detection regions configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: All motion detection regions share a sensitivity level. It is possible to define mask regions that are subtracted from other regions.	

MotionDetectionRegionList XML Block

```
<MotionDetectionRegionList version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <sensitivityLevel>       <!-- req -->
        <!-- req, xs:integer, "0-5", 0 is least sensitive -->
    </sensitivityLevel>
    <MotionDetectionRegion/>    <!-- opt -->
</MotionDetectionRegionList>
```

8.10.3 Motion detection region

/MotionDetection/ID/regions/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular motion detection region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	**MotionDetectionRegion**
PUT	**Operator**
Description	It is used to update a particular motion detection region configuration for a video input channel.
Query	None
Inbound Data	**MotionDetectionRegion**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**

Description	It is used to delete a particular motion detection region configuration for a video input channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:
The region detection coordinate space depends on the value of <regionType>.
Only support the rectangular region which will be "drawn" from four coordinates. The four points is clockwise direction, and the beginning point is the low-left point.

MotionDetectionRegion XML Block

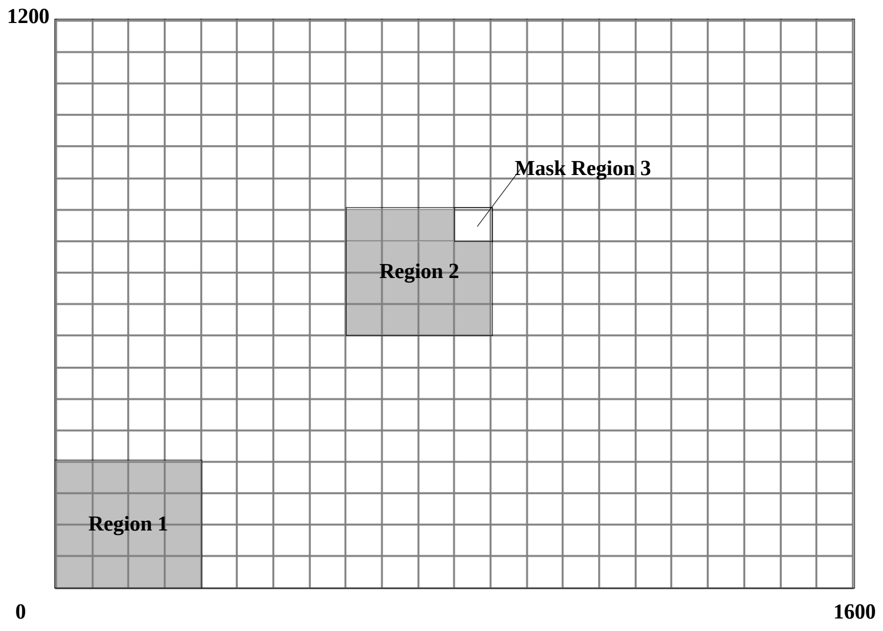
```
<MotionDetectionRegion version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id>              <!-- req, xs:integer, "1-16" -->   </id>
  <enabled>            <!-- req, xs:boolean -->   </enabled>
  <maskEnabled>         <!-- req, xs:boolean -->   </maskEnabled>
  <RegionCoordinatesList>   <!-- req -->
    <RegionCoordinates>   <!-- Note: at least four coordinates are required -->
      <positionX>         <!-- req, xs:integer -->   </positionX>
      <positionY>         <!-- req, xs:integer -->   </positionY>
    </RegionCoordinates>
  </RegionCoordinatesList>
</MotionDetectionRegion>
```

8.10.4 Motion Detection Example

Set up Motion Detection

The following command configures two rectangular detection regions, with one "masked" region on video input channel ID 1. Example assumes a resolution of 1600x1200 and a grid motion detection algorithm:

- Motion detection is enabled with a granularity of a 22x18 grid – this means the detection region coordinates will ultimately be defined by a grid of 396 regions. For a resolution of 1600x1200, this means that each "granule" will be 1600/22 x 1200/18 pixels. (If a coordinate doesn't exactly match the configured granularity, it should be mapped internally to the nearest possible point).
- Two detection regions are defined, the second containing an inner/overlapping region that is disabled. Region 1 occupies the bottom-left 16 granules. Region 2 occupies the middle 16 granules, with the top-right-most corner granule (region 3) disabled by use of the <maskEnabled> tag.



```
PUT /MotionDetection/1 HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<MotionDetection version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id>1</id>
  <enabled>true</enabled>
  <MotionDetectionRegionList>
    <sensitivityLevel>2</sensitivityLevel>
    <MotionDetectionRegion>
      <id>1</id>
      <enabled>true</enabled>
      <maskEnabled>false</maskEnabled>
      <RegionCoordinatesList>
        <RegionCoordinates>
          <positionX>0</positionX>
          <positionY>0</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>0</positionX>
          <positionY>4</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>4</positionX>
          <positionY>4</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
```

```
          <positionX>4</positionX>
          <positionY>0</positionY>
        </RegionCoordinates>
      </RegionCoordinatesList>
    </MotionDetectionRegion>
    <MotionDetectionRegion>
      <id>2</id>
      <enabled>true</enabled>
      <maskEnabled>false</maskEnabled>
      <RegionCoordinatesList>
        <RegionCoordinates>
          <positionX>8</positionX>
          <positionY>8</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>8</positionX>
          <positionY>12</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>12</positionX>
          <positionY>12</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>12</positionX>
          <positionY>8</positionY>
        </RegionCoordinates>
      </RegionCoordinatesList>
    </MotionDetectionRegion>
    <MotionDetectionRegion>
      <id>3</id>
      <enabled>true</enabled>
      <maskEnabled>true</maskEnabled>
      <RegionCoordinatesList>
        <RegionCoordinates>
          <positionX>11</positionX>
          <positionY>11</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>11</positionX>
          <positionY>12</positionY>
        </RegionCoordinates>
        <RegionCoordinates>
          <positionX>12</positionX>
          <positionY>12</positionY>
```

```
            </RegionCoordinates>
            <RegionCoordinates>
               <positionX>12</positionX>
               <positionY>11</positionY>
            </RegionCoordinates>
         </RegionCoordinatesList>
      </MotionDetectionRegion>
   </MotionDetectionRegionList>
</MotionDetection>
```

8.11 Event

/Event	Service v1.0
GET	**Viewer**
Description	It is used to get the configuration of the device event behavior, scheduling and notifications.
Query	None
Inbound Data	None
Success Return	**EventNotification**
PUT	**Operator**
Description	It is used to udpate the configuration of the device event behavior, scheduling and notifications.
Query	None
Inbound Data	**EventNotification**
Success Return	ResponseStaus **ResponseStatus**
Notes: The event trigger list defines the set of device behaviors that trigger events. The event schedule defines when event notifications are active. The event notification methods define what types of notification (e-mail) are supported.	

EventNotification XML Block

```
<EventNotification version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <EventTriggerList/>          <!-- opt -->
   <EventSchedule/>             <!-- opt -->
   <EventNotificationMethods/>   <!-- opt -->
</EventNotification>
```

8.11.1 Triggers

/Event/triggers	General Resource v1.0

GET	Viewer
Description	It is used to get the list of event triggers.
Query	None
Inbound Data	None
Success Return	**EventTriggerList**
PUT	**Operator**
Description	It is used to update the list of event triggers.
Query	None
Inbound Data	**EventTriggerList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add an event trigger.
Query	None
Inbound Data	**EventTrigger**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the list of event triggers.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Event triggering defines how the device reacts to particular events, such as video loss or motion detection.	

EventTriggerList XML Block

```
<EventTriggerList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <EventTrigger/>    <!-- opt -->
</EventTriggerList>
```

8.11.2 Trigger

/Event/triggers/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular event trigger configuration.
Query	None
Inbound Data	None
Success Return	**EventTrigger**
PUT	**Operator**
Description	It is used to update a particular event trigger configuration.
Query	None

Inbound Data	**EventTrigger**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular event trigger.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:

An event trigger determines how the device reacts when a particular event is detected. The following types are supported:

IO: trigger when an input IO port changes state.

VMD: trigger on video motion detection.

Video loss: trigger when the input video signal cannot be detected.

Shelter alarm: trigger when shelter is set.

The "ID" in the URI is the sequence number of a trigger , the max value of <id> is depend on device. The first trigger id is 1.

<inputIOPortID> is only required if <eventType> is "IO".

EventTriggerList XML Block

```
<EventTrigger version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>                    <!-- req, xs:integer-->             </id>
   <eventType>   <!-- req, xs:string, "IO,VMD,videoloss,shelteralarm" -->     </eventType>
   <eventDescription>          <!-- ro, req, xs:string -->             </eventDescription>
   <inputIOPortID>          <!-- ro, req, xs:string -->             </inputIOPortID>
   <EventTriggerNotificationList/>    <!-- req -->
</EventTrigger>
```

8.11.3 Trigger notifications

/Event/triggers/*ID*/notifications	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of notification methods and behaviors for an event trigger.
Query	None
Inbound Data	None
Success Return	**EventTriggerNotificationList**
PUT	**Operator**
Description	It is used to update the list of notification methods and behaviors for an event trigger.
Query	None
Inbound Data	**EventTriggerNotificationList**

Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a notification method and behavior for an event trigger.
Query	None
Inbound Data	**EventTriggerNotification**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the list of notification method and behavior for an event trigger.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: This section determines the kinds of notifications that are supported for a particular event trigger and their recurrences and behaviors.	

EventTriggerNotificationList XML Block

```
<EventTriggerNotificationList version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <EventTriggerNotification/>    <!-- opt -->
</EventTriggerNotificationList>
```

8.11.4 Trigger notification

/Event/triggers/ID/notifications/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular notification method and behavior for an event trigger.
Query	None
Inbound Data	None
Success Return	**EventTriggerNotification**
PUT	**Operator**
Description	It is used to update a particular notification method and behavior for an event trigger.
Query	None
Inbound Data	**EventTriggerNotification**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular notification method and behavior for an event trigger.

Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:
The first "ID" in the URI is the sequence number of a trigger , the max value of <id> is depend on device. The first trigger id is 1.
The second "ID" in the URI is the sequence number a notification , the max value of <id> is depend on device. The first notification id is 1.
<outputIOPortID> is only required if the <notifiocationMethod> is "IO".

EventTriggerNotification XML Block

```
<EventTriggerNotification version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <id>                  <!-- req, xs:integer -->                </id>
  <notificationMethod>   <!-- req, xs:string, "email, IO,record" -->   </notificationMethod>
  <notificationRecurrence> <!-- ro, req, xs:string, "beginning" -->
  </notificationRecurrence>
  <outputIOPortID>        <!-- ro, dep, xs:integer -->                </outputIOPortID>
</EventTriggerNotification>
```

8.11.5 Schedule

/Event/schedule	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get event schedules.
Query	None
Inbound Data	None
Success Return	**EventSchedule**
PUT	**Operator**
Description	It is used to update event schedules.
Query	None
Inbound Data	**EventSchedule**
Success Return	ResponseStaus **ResponseStatus**

Notes:
Defines the schedule. The schedule is defined as a set of time blocks that define when the events are active.
The schedule is always valid.
It only supports one TimeBlock every day now.

EventSchedule XML Block

```
<EventSchedule version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
```

```
  <TimeBlockList> <!-- req -->
    <TimeBlock>
      <dayOfWeek>
        <!-- opt, xs:integer, ISO8601 weekday number, 1=Monday, … -->
      </dayOfWeek>
      <TimeRange>          <!-- req -->
        <beginTime>      <!-- req, xs:time, ISO8601 time -->    </beginTime>
        <endTime>      <!-- req, xs:time, ISO8601 time -->    </endTime>
      </TimeRange>
   </TimeBlock>
  </TimeBlockList>
</EventSchedule>
```

8.11.6 Schedule/ID

/Event/Schedule/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get event schedules.
Query	None
Inbound Data	None
Success Return	**EventSchedule**
PUT	**Operator**
Description	It is used to update event schedules.
Query	None
Inbound Data	**EventSchedule**
Success Return	ResponseStaus **ResponseStatus**
Notes: ID is defined as TypeName. If the event type is IO, the ID is IO_IN_PortNumber. Examples : VMD : Video Motion Detection videoloss : Video Loss shelteralarm : Shelter Alarm IO_ IN _1 :the first IO input port IO_OUT_2 : the second IO output port	

EventSchedule XML Block

```
<EventSchedule version="1.0"  xmlns="http://www.std-cgi.com/ver10/XMLSchema">

 <eventType>                <!-- req -->
     <!-- req, xs:string,
      "IO,VMD,videoloss, shelteralarm"

    -->
  </eventType>
  <inputIOPortID>        <!-- dep, xs:string -->          </inputIOPortID>
  <outputIOPortID>        <!-- dep, xs:string -->          </inputIOPortID>
<TimeBlockList> <!-- req -->
    <TimeBlock>
      <dayOfWeek>

        <!-- opt, xs:integer, ISO8601 weekday number, 1=Monday,  …  -->

      </dayOfWeek>
      <TimeRange>         <!-- req -->
        <beginTime>      <!-- req, xs:time, ISO8601 time -->   </beginTime>
        <endTime>      <!-- req, xs:time, ISO8601 time -->   </endTime>
      </TimeRange>
   </TimeBlock>
</TimeBlockList>
</EventSchedule>
```

8.11.7 Notification

/Event/notification	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get event notifications configuration.
Query	None
Inbound Data	None
Success Return	**EventNotificationMethods**
PUT	**Operator**
Description	It is used to update event notifications configuration.
Query	None
Inbound Data	**EventNotificationMethods**
Success Return	ResponseStaus **ResponseStatus**
Notes: E-mail notification type is supported. E-mail: a mail with relevant information is sent in an e-mail to a list of servers.	

EventNotificationMethods XML Block

```
<EventNotificationMethods version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
```

```
  <MailingNotificationList/>       <!-- opt -->
  <EmailFormat>                  <!-- opt -->
    <senderEmailAddress>       <!-- req, xs:string -->    </senderEmailAddress>
    <receiverEmailAddress>       <!-- req, xs:string -->    </receiverEmailAddress>
  </EmailFormat>
</EvenNotificationMethods>
```

8.11.8 Mails notification

/Event/notification/mailing	General Resource v1.0
GET	**Viewer**
Description	It is used to get the list of E-mail notifications.
Query	None
Inbound Data	None
Success Return	**MailingNotificationList**
PUT	**Operator**
Description	It is used to update the list of E-mail notifications.
Query	None
Inbound Data	**MailingNotificationList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add an E-mail notification.
Query	None
Inbound Data	**MailingNotification**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the list of E-mail notifications.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: When the notification is triggered, an e-mail with relevant information is mailed to the each of the addresses in the mailing list.	

MailingNotificationList XML Block

```
<MailingNotificationList version="1.0"
 xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <MailingNotification/>       <!-- opt -->
</MailingNotificationList>
```

8.11.9 Mail notification

/Event/notification/mailing/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular E-mail notification configuration.
Query	None
Inbound Data	None
Success Return	**MailingNotification**
PUT	**Operator**
Description	It is used to update a particular E-mail notification configuration.
Query	None
Inbound Data	**MailingNotification**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular E-mail notification.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the SMTP server. <authenticationMode> determines the authentication requirements for sending an email from the device. <portNo> is the port number of the SMTP server entry. <accountName> is the user account name for the SMTP server.	

MailingNotification XML Block

```
<MailingNotification version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>               <!-- req, xs:integer, "1" -->      </id>
   <authenticationMode>
      <!-- req, xs:string, "SMTP" -->
   </authenticationMode>
   <addressingFormatType>
      <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>          <!-- dep, xs:string -->      </hostName>
   <ipAddress>          <!-- dep, xs:string -->      </ipAddress>
   <portNo>            <!-- ro, req, xs:integer -->      </portNo>
   <accountName>         <!-- req, xs:string -->   </accountName>
   <password>           <!-- req, xs:string -->   </password>
   <attachmentEnable>     <!-- opt, xs:Boolean,"true,false" -->   </attachmentEnable>
</MailingNotification>
```

8.11.10 Notification alertStream

/Event/notification/alertStream	General Resource v1.0
GET	**Viewer**
Description	It is used to get the event notification data stream through HTTP server push.
Query	None
Inbound Data	None
Success Return	**Stream of <EventNotificationAlert>**
Notes: This function is used to get an event notification alert stream from the media device via HTTP or HTTPS. This function does not require that a client/VMS system be added as an HTTP(S) destination on the media device. Instead, the client/VMS system can call this API to initialize a stream of event information from the device. In other words, a connection is established with the device when this function is called, and stays open to constantly receive event notifications. This API uses HTTP server-push with the MIME type multipart/mixed defined in RFC 2046. <protocol> is the protocol name, i.e. “HTTP” or “HTTPS”. <channelID> is present for video and analytics events. <activePostCount> is the sequence number of current notification for this particular event. It starts at 1. Useful for recurring notifications of an event. Each event maintains a separate post count.	

EventNotificationAlert XML Block

```
<EventNotificationAlert version=“1.0”
 xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <ipAddress>         <!-- dep, xs:string -->       </ipAddress>
  <portNo>         <!-- opt, xs:integer -->       </portNo>
  <protocol>        <!-- opt, xs:string -->       </protocol>
  <macAddress>      <!-- opt, xs:string;MAC -->   </macAddress>
  <channelID>         <!-- dep, xs:string -->       </channelID>
  <dateTime>        <!-- req, xs:datetime -->     </dateTime>
  <activePostCount>   <!-- req, xs:integer -->       </activePostCount>
  <eventType>   <!-- req, xs:string, “IO,VMD,videoloss, shelteralarm” -->   </eventType>
  <eventState>      <!-- req, xs:string, “active,inactive” -->   </eventState>
  <eventDescription>   <!-- req, xs:string -->                     </eventDescription>
  <inputIOPortID>   <!-- dep, xs:integer, if <eventType> is “IO” -->      </inputIOPortID>
  <DetectionRegionList>          <!-- dep, if <eventType> is “VMD” -->
    <DetectionRegionEntry>       <!-- req -->
      <regionID>           <!-- req, xs:string -->          </regionID>
```

```
        <sensitivityLevel>      <!-- req, xs:integer, 0..100 -->   </sensitivityLevel>
      </DetectionRegionEntry>
    </DetectionRegionList>
</EventNotificationAlert>
```

Example

The following is an example of an HTTP event stream that pushes a VMD event from video channel 1.

```
GET /Event/notification/alertStream HTTP/1.1
…
HTTP/1.1 200 OK
MIME-Version: 1.0
Content-Type:   multipart/mixed; boundary="<boundary>"
--<boundary>
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<EventNotificationAlert version="1.0"
  xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <ipAddress>172.6.64.7</ipAddress>
    <portNo>80</portNo>
    <protocol>HTTP</protocol>
    <macAddress>01:17:24:45:D9:F4</macAddress>
    <channelID>1</channelID>
    <dateTime>2009-11-14T15:27Z</dateTime>
    <activePostCount>1</activePostCount>
    <eventType>VMD</eventType>
    <eventState>active</eventState>
    <eventDescription>Motion alarm</eventDescription>
    <DetectionRegionList>
      <DetectionRegionEntry>
        <regionID>2</regionID>
        <sensitivityLevel>4</sensitivityLevel>
      </DetectionRegionEntry>
    </DetectionRegionList>
</EventNotificationAlert>
--<boundary>
…
```

8.11.11 Event Triggering Examples

Example: Trigger Events on IO Port

The command below enables detection for input port 1. When the input signal is detected according to <inputIOPortID>, two event notification responses are used – output port 2 will be triggered for the duration of the input signal detection, and an SMTP server will be notified with the “E-mail Event Notification Alert”. The behavior of this notification is as follows:

- A SMTP notification is sent at detection time, and every some seconds after while the signal is present. This is denoted by the <notificationRecurrence> tags. These APIs will have an <eventState> of “active”.
- When the input port 1 signal detection stops, one last E-mail notification is sent to the server with an <eventState> of “active”.
- After the signal detection stops for input port 1, the device will wait some seconds before starting to detect the signal again for this port.

```
PUT /Event/triggers HTTP/1.1
Content-Type: application/xml; charset=“UTF-8”
Content-Length: xxx

<?xml version=“1.0” encoding=“UTF-8”?>
<EventTrigger version=“1.0”   xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <id>1</id> <!-- “eventType: IO” -->
  <EventTriggerNotificationList>
     <EventTriggerNotification>
        <id>1</id> <!-- “notificationMethod: email” -->
     </EventTriggerNotification>
     <EventTriggerNotification>
        <id>2</id> <!-- “notificationMethod: IO” -->
     </EventTriggerNotification>
  </EventTriggerNotificationList>
</EventTrigger>
```

Example: Schedule event detection and triggering

The command below schedules event detection and triggering from 7:00 am to 5:00 pm. every Tuesday.

```
PUT /Event/schedule HTTP/1.1
Content-Type: application/xml; charset=“UTF-8”
Content-Length: xxx

<?xml version=“1.0” encoding=“UTF-8”?>
<EventSchedule version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <TimeBlockList>
```

```
    <TimeBlock>
      <dayOfWeek>2</dayOfWeek>
      <TimeRange>
        <beginTime>07:00:00</beginTime>
        <endTime>17:00:00</endTime>
      </TimeRange>
    </TimeBlock>
  </TimeBlockList>
</EventSchedule>
```

8.12 PTZ

/PTZ	**Service v1.0**
Notes: PTZ control service.	

8.12.1 Channels

/PTZ/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of PTZ channels for the device.
Query	None
Inbound Data	None
Success Return	**PTZChannelList**
PUT	**Operator**
Description	It is used to update the list of PTZ channels for the device.
Query	None
Inbound Data	**PTZChannelList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a PTZ channel for the device.
Query	None
Inbound Data	**PTZChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the list of PTZ channels for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

PTZ channels may be hardwired, or it may be possible to create channels if the device supports it. To determine whether it is possible to dynamically PTZ channels, check the defined HTTP methods in /PTZ/channels/description.

PTZChannelList XML Block

```
<PTZChannelList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <PTZChannel/>       <!-- opt -->
</PTZChannelList>
```

8.12.2 Channel

/PTZ/channels/ID	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular PTZ channel configuration for the device.
Query	None
Inbound Data	None
Success Return	**PTZChannel**
PUT	**Operator**
Description	It is used to update a particular PTZ channel configuration for the device.
Query	None
Inbound Data	**PTZChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <videoInputID> links the PTZ channel to a video channel. <controlProtocol> indicates the control protocol to be used for PTZ.	

PTZChannel XML Block

```
<PTZChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>   <!-- req, xs:integer -->   </id>
   <videoInputID>       <!-- req, xs:integer -->   </videoInputID>
   <controlProtocol>   <!-- req: xs:string --> </controlProtocol>
   <controlAddress>          <!-- req: xs:integer -->   </controlAddress>
   <PresetIDList>   <! – opt -->
      <PresetID>   <! – opt -->
         <id> <!-- ro, req, xs:integer, "1-128" --> </id>
```

```
        <enabled> <!-- req, xs:boolean --> </enabled>
      </PresetID>
   </PresetIDList>
   <PatrolIDList>   <! – opt -->
      <PatrolID>     <! – opt -->
         <id> <!-- ro, req, xs:integer, “1-16” --> </id>
         <enabled> <!-- req, xs:boolean --> </enabled>
      </PatrolID>
   </PatrolIDList>
   <PatternIDList>   <! – opt -->
      <PatternID>     <! – opt -->
         <id> <!-- ro, req, xs:integer, “1-16” --> </id>
         <enabled> <!-- req, xs:boolean --> </enabled>
      </PatternID>
   </PatternIDList>
</PTZChannel>
```

8.12.3 Patrols

/PTZ/channels/*ID*/patrols	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of patrols for a PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZPatrolList**
Notes:	

PTZPatrolList XML Block

```
<PTZPatrolList version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <PTZPatrol>   <!-- opt -->
</PTZPatrolList>
```

8.12.4 Patrol

/PTZ/channels/*ID*/patrols/*ID*	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular patrol configuration for a PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZPatrol**

PUT	**Operator**
Description	It is used to update a particular patrol configuration for a PTZ channel.
Query	None
Inbound Data	**PTZPatrol**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

PTZPatrol XML Block

```
<PTZPatrol version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>     <!-- req, xs:integer -->     </id>
   <PatrolPointList />     <!—opt -->
</PTZPatrol>
```

8.12.5 Patrol keyPoints

/PTZ/channels/*ID*/patrols/*ID*/keyPoints	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of key points of a particular patrol for a PTZ channel.
Query	None
Inbound Data	None
Success Return	**PatrolPointList**
PUT	**Operator**
Description	It is used to update the list of key points of a particular patrol for a PTZ channel.
Query	None
Inbound Data	**PatrolPointList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a key point of a particular patrol for a PTZ channel.
Query	None
Inbound Data	**PatrolPoint**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete the list of key points of a particular patrol for a PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:

PatrolPointList XML Block

```
<PatrolPointList version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <PatrolPoint />   <!—opt -->
</PatrolPointList>
```

8.12.6 Patrol keyPoint

/PTZ/channels/*ID*/patrols/*ID*/keyPoints/*ID*	General Resource v1.0
GET	**Viewer**
Description	It is used to get a particular key point of a particular patrol for a PTZ channel.
Query	None
Inbound Data	None
Success Return	**PatrolPoint**
PUT	**Operator**
Description	It is used to update a particular key point of a particular patrol for a PTZ channel.
Query	None
Inbound Data	**PatrolPoint**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular key point of a particular patrol for a PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <presetNo> is Preset’s series number. <speed> is Patrol speed. <dwellTime> is the stay time for the patrol point, the unit is second	

PatrolPoint XML Block

```
<PatrolPoint version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <id>   <!-- req, xs:integer -->   </id>
   <presetNo> <!-- req, xs:integer -->   </presetNo>
   <speed> <!—opt, xs:integer -->   </speed>
   <dwellTime> <!—opt, xs:integer -->   </dwellTime>
</PatrolPoint>
```

8.12.7 PTZControl

/PTZ/channels/*ID*/PTZControl	General Resource v1.0
PUT	**Operator**
Description	It is used to control PTZ.
Query	command presetNo patrolNo mode speed
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: The value of command is: LIGHT: Light WIPER: Wiper FAN: Fan HEATER: Heater. AUX1: auxiliary equipment 1. AUX2: auxiliary equipment 2 SET_PRESET: Set preset CLE_PRESET: Clear preset. ZOOM_IN: Zoom in the specified speed. ZOOM_OUT: Zoom out in the specified speed. FOCUS_NEAR: focus near in the specified speed. FOCUS_FAR: focus far in the specified speed. IRIS_OPEN: IRIS is open in the specified speed IRIS_CLOSE: IRIS is cloesd in the specified speed TILT_UP: PTZ is tilt up in the specified speed TILT_DOWN: PTZ is tilt down in the specified speed PAN_LEFT: PTZ is pan left in the specified speed PAN_RIGHT: PTZ is pan right in the specified speed UP_LEFT: PTZ is up-left in the specified speed UP_RIGHT: PTZ is up-right in the specified speed DOWN_LEFT: PTZ is down-left in the specified speed DOWN_RIGHT: PTZ is down-right in the specified speed PAN_AUTO: PTZ scans pan with the specified speed. MEM_PATTERN: memory pattern. RUN_PATTERN: Start pattern. PATROL: patrol. GOTO_PRESET: Go to preset. “mode” value is “start” and “stop”. It indicates the “start” or “stop” of some actions for PTZ,	

or the “turn on” or “turn off” of external equipment power for PTZ. The default is “start”. In addition to the “SET_PRESET”, “CLE_PRESET”, “RUN_PATTERN” and “GOTO_PRESET” command, all commands require the “mode” query parameters. “speed” range is 1-7. When the command is “ZOOM_IN”, “ZOOM_OUT”, “FOCUS_NEAR”, “FOCUS_FAR”, “IRIS_OPEN”, or “IRIS_CLOSE”, the default is 1. When the command is “TILT_UP”, “TILT_DOWN”, “PAN_LEFT”, “PAN_RIGHT”, “UP_LEFT”, “UP_RIGHT”, “DOWN_LEFT”, “DOWN_RIGHT”, “PAN_AUTO”, the default is 3.

8.13 PTZCtrl

/PTZCtrl	**Service v1.0**
Notes: PTZCtrl control service.	

8.13.1 /PTZCtrl/channels

/PTZCtrl/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of PTZ channels for the device
Query	None
Inbound Data	None
Success Return	**PTZChannelList**
PUT	**Operator**
Description	It is used to update the list of PTZ channels for the device.
Query	None
Inbound Data	**PTZChannelList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a PTZ channel for the device.
Query	None
Inbound Data	**PTZChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete the list of PTZ channels for the device.
Query	None
Inbound Data	None

Success Return	ResponseStaus **ResponseStatus**
Notes: PTZ channels may be hardwired, or it may be possible to create channels if the device supports it. To determine whether it is possible to dynamically PTZ channels, check the defined HTTP methods in /PTZCtrl/channels/description.	

PTZChannelList XML Block

```
<PTZChannelList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <PTZChannel/>       <!-- opt -->
</PTZChannelList>
```

8.13.2 /PTZCtrl/channels/<ID>

/PTZCtrl/channels/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular PTZ channel configuration for the device.
Query	None
Inbound Data	None
Success Return	**PTZChannel**
PUT	**Operator**
Description	It is used to update a particular PTZ channel configuration for the device.
Query	None
Inbound Data	**PTZChannel**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular PTZ channel on a device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <videoInputID> links the PTZ channel to a video channel. <presetSpeed> indicates the movement speed level about calling preset <autoScanSpeed> indicates the movement speed level about park function <keyPadControlSpeed> indicates the movement speed level to be controlled by keyboard <controlProtocol> indicates the control protocol to be used for PTZ. <controlAddress> indicates the soft address (enabled means soft address is used)	

PTZChannel XML Block

```
<PTZChannel version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <id>   <!-- req, xs:integer -->   </id>
   <enabled>   <!—ro,req, xs:boolean -->   </enabled>
   <videoInputID>       <!-- req, xs:integer -->   </videoInputID>
   <panMaxSpeed> <!—ro,opt, xs:integer, degrees/sec --> </panMaxSpeed>
   <tiltMaxSpeed> <!—ro,opt, xs:integer, degrees/sec --> </tiltMaxSpeed>
   <presetSpeed>     <!—opt, xs:integer 1..8 –>    </presetSpeed>
   <autoPatrolSpeed> <!-- opt, xs:integer, 0..100 --> </autoPatrolSpeed>
   <keyBoardControlSpeed>
         <!-- opt, xs:string, "low, normal, high">
   </keyBoardControlSpeed>
   <controlProtocol> <!-- opt, xs:string, "pelco-d,…" --> </controlProtocol>
   <controlAddress>           <!—opt -->
         <enabled>          <!-- req, xs:boolean -->    </enabled>
         <Address>       <!—opt, xs:string   1-255 -->    </Address>
   </controlAddress>
   <defaultPresetID> <!-- opt, xs:string;id --> </defaultPresetID>
</PTZChannel>
```

8.13.3 /PTZCtrl/channels/<ID>/homeposition

/PTZCtrl/channels/<ID>/homeposition	**General Resource v1.0**
PUT	**Operator**
Description	It is used to set the current horizontal position as horizontal coordinate zero point for the device
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete system horizontal coordinate zero point and restore default zero point for the device (The photoelectric detection location)
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.4 /PTZCtrl/channels/<ID>/homeposition/goto

/PTZCtrl/channels/<ID>/homeposition/goto	**General Resource v1.0**
PUT	**Operator**
Description	It is used to move a particular PTZ channel to horizontal coordinate zero point position for the device.

Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.5 /PTZCtrl/channels/<ID>/continuous

/PTZCtrl/channels/<ID>/continuous	**General Resource v1.0**
PUT	**Operator**
Description	It is used to control PTZ move around and zoom for the device.
Query	pan, tilt, zoom
Inbound Data	**PTZData**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

PTZData XML Block

```
<PTZData version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <pan> <!-- opt, xs:integer, -100..100 --> </pan>
    <tilt> <!-- opt, xs:integer, -100..100 --> </tilt>
    <zoom> <!-- opt, xs:integer, -100.. 100--> </zoom>
</PTZData>
```

8.13.6 /PTZCtrl/channels/<ID>/momentary

/PTZCtrl/channels/<ID>/momentary	**General Resource v1.0**
PUT	**Operator**
Description	It is used to control PTZ move around and zoom in a period of time for the device.
Query	pan, tilt, zoom, duration
Inbound Data	**PTZData**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

PTZData XML Block

```
<PTZData version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <pan> <!-- opt, xs:integer, -100..100 --> </pan>
    <tilt> <!-- opt, xs:integer, -100..100 --> </tilt>
    <zoom> <!-- opt, xs:integer, -100.. 100--> </zoom>
    <Momentary>
        <duration>   <!—opt , xs:integer, milliseconds -->   </duration>
    </Momentary>
</PTZData>
```

8.13.7 /PTZCtrl/channels/<ID>/relative

/PTZCtrl/channels/<ID>/relative	**General Resource v1.0**
PUT	**Operator**
Description	It is used to move the position which is defined by positionX, positionY to the screen center and relative zoom for the device.
Query	positionX, positionY, relativeZoom
Inbound Data	**PTZData**
Success Return	ResponseStaus **ResponseStatus**
Notes: Mouse clicking function	

PTZData XML Block

```
<PTZData version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <Relative>
        <positionX> <!-- opt, xs:integer --> </positionX>
        <positionY> <!-- opt, xs:integer --> </positionY>
        <relativeZoom> <!-- opt, xs:integer, -100-.. 100---> </relativeZoom>
    </Relative>
</PTZData>
```

8.13.8 /PTZCtrl/channels/<ID>/absolute

/PTZCtrl/channels/<ID>/absolute	**General Resource v1.0**
PUT	**Operator**
Description	It is used to move a particular PTZ channel to a absolute position which is defined by Absolute for the device.
Query	elevation, azimuth, absoluteZoom
Inbound Data	**PTZData**
Success Return	ResponseStaus **ResponseStatus**
Notes: Absolute position function <AbsoluteHigh> is high precision positioning which is accurate to a bit after the decimal point; For example elevation -900..2700 is corresponding to vertical -90.0-270.0 degree, and azimuth 0..3600 is corresponding to horizontal 0.0-360.0 degree, absoluteZoom is corresponding to zoom 0.0..100.0;	

PTZData XML Block

```
<PTZData version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <AbsoluteHigh>
        <elevation> <!-- opt, xs:integer, -900..2700 --> </elevation>
        <azimuth> <!-- opt, xs:integer, 0..3600 --> </azimuth>
        <absoluteZoom> <!-- opt, xs:integer,0.. 1000---> </absoluteZoom>
    </AbsoluteHigh>
</PTZData>
```

8.13.9 /PTZCtrl/channels/<ID>/digital

/PTZCtrl/channels/<ID>/digital	**General Resource v1.0**
PUT	**Operator**
Description	It is used to move the position number which is defined by positionX, position to the screen center and digital zoom for the device.
Query	position, positionY, digitalZoomLevel
Inbound Data	**PTZData**
Success Return	ResponseStaus **ResponseStatus**
Notes: Digital zoom function digitalZoomLevel: 0 indicates that maintain the original image ratio.	

PTZData XML Block

```
<PTZData version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
<Digital>
        <positionX> <!-- opt, xs:integer --> </positionX>
        <positionY> <!-- opt, xs:integer --> </positionY>
        <digitalZoomLevel> <!-- opt, xs:integer, 0.. 100---> </digitalZoomLevel>
    </Digital>
</PTZData>
```

8.13.10 /PTZCtrl/channels/<ID>/status

/PTZCtrl/channels/<ID>/status	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get currently PTZ coordinate position for the device.
Query	None
Inbound Data	None
Success Return	**PTZStatus**
Notes: <AbsoluteHigh> is high precision positioning which is accurate to a bit after the decimal point; For example elevation -900..2700 is corresponding to vertical -90.0-270.0 degree, and azimuth 0..3600 is corresponding to horizontal 0.0-360.0 degree, absoluteZoom is corresponding to zoom 0.0..100.0;	

PTZStatus XML Block

```
<PTZStatus version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <AbsoluteHigh>
        <elevation> <!-- opt, xs:integer, -900..2700 --> </elevation>
        <azimuth> <!-- opt, xs:integer, 0..3600 --> </azimuth>
        <absoluteZoom> <!-- opt, xs:integer,0.. 1000---> </absoluteZoom>
    </AbsoluteHigh>
</PTZStatus>
```

8.13.11 /PTZCtrl/channels/<ID>/presets

/PTZCtrl/channels/<ID>/presets	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	**PTZPresetList**
PUT	**Operator**
Description	It is used to update preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	**PTZPresetList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	**PTZPreset**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete a preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

PTZPresetList XML Block

```
<PTZPresetList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <PTZPreset>   <!-- opt -->
```

```
</PTZPresetList>
```

8.13.12 /PTZCtrl/channels/<ID>/presets/<ID>

/PTZCtrl/channels/<ID>/presets/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get particular preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	**PTZPreset**
PUT	**Operator**
Description	It is used to update particular preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	**PTZPreset**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular preset configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <id> indicates the preset number. <presetName> indicates the preset name Enable is used to indicate whether preset have been set. PUT is used to set preset and update title of new preset. (Enable value import to **PTZPreset** should be 1 when PUT)	

PTZPreset XML Block

```
<PTZPreset version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <enabled>   <!-- req, xs:boolean -->   </enabled>
     <id> <!-- req, xs:string;id --> </id>
     <presetName> <!-- req, xs:string --> </presetName>
</PTZPreset>
```

8.13.13 /PTZCtrl/channels/<ID>/presets/<ID>/goto

/PTZCtrl/channels/<ID>/presets/<ID>/goto	**General Resource v1.0**
PUT	**Operator**
Description	It is used to move a particular PTZ channel to a ID preset position for the device.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.14 /PTZCtrl/channels/<ID>/patrols

/PTZCtrl/channels/<ID>/presets	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get patrol configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	None
Success Return	**PTZPatrolList**
PUT	**Operator**
Description	It is used to update patrol configuration information of a particular PTZ channel for the device.
Query	None
Inbound Data	**PTZPatrolList**
Success Return	ResponseStaus **ResponseStatus**
POST	**Operator**
Description	It is used to add a patrol point configuration for a particular PTZ channel.
Query	None
Inbound Data	**PTZPatrol**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Administrator**
Description	It is used to delete patrol configuration for a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: It is similar to presets!!	

PTZPatrolList XML Block

```
<PTZPatrolList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <PTZPatrol>   <!-- opt -->
</PTZPatrolList>
```

8.13.15 /PTZCtrl/channels/<ID>/patrols/<ID>

/PTZCtrl/channels/<ID>/patrols/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular patrol route configuration of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZPatrol**
PUT	**Operator**
Description	It is used to update a particular patrol configuration of a particular PTZ channel.
Query	None
Inbound Data	**PTZPatrol**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular patrol route configuration of a particular PTZ channel
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <PatrolSequence> indicates the patrol point. <presetID> indicates the preset number <seqSpeed> indicates the patrol speed <delay> indicates the dwell time, in seconds	

PTZPatrol XML Block

```
<PTZPatrol version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <enabled>   <!-- req, xs:boolean -->   </enabled>
     <id> <!-- req, xs:string;id --> </id>
    <patrolName> <!-- req, xs:string --> </patrolName>
    <resumeType> <!-- opt, xs:string, "relative,absolute" --> </resumeType>
     <PatrolSequenceList> <!-- req, at least one entry -->
         <PatrolSequence> <!-- req -->
             <id> <!-- req, xs:string;id --></id>
              <presetID> <!-- req, xs:string;id --> </presetID>
              <seqSpeed> <!-- req, xs:string;id --> </seqSpeed>
```

```
            <delay> <!-- req, xs:integer, milliseconds --> </delay>
        </PatrolSequence>
    </PatrolSequenceList>
</PTZPatrol>
```

8.13.16 /PTZCtrl/channels/<ID>/patrols/<ID>/start

/PTZCtrl/channels/<ID>/patrols/<ID>/start	**General Resource v1.0**
PUT	**Operator**
Description	It is used to start running particular patrol route of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.17 /PTZCtrl/channels/<ID>/patrols/<ID>/stop

/PTZCtrl/channels/<ID>/patrols/<ID>/stop	**General Resource v1.0**
PUT	**Operator**
Description	It is used to stop running particular patrol route of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: It is available to stop the patrol route which is in running state or in pause state.	

8.13.18 /PTZCtrl/channels/<ID>/patrols/<ID>/pause

/PTZCtrl/channels/<ID>/patrols/<ID>/pause	**General Resource v1.0**
PUT	**Operator**
Description	It is used to pause particular patrol route which is in running state of a particular channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Patrolstart is used to restart patrol route. It doesn’t support dome at this moment.	

8.13.19 /PTZCtrl/channels/<ID>/patrols/<ID>/status

/PTZCtrl/channels/<ID>/patrols/<ID>/status	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get particular patrol route state of a particular PTZ channel.
Query	None
Inbound Data	**PTZPatrolStatus**
Success Return	ResponseStaus **ResponseStatus**
Notes: It doesn't support dome at this moment!!	

PTZPatrolStatus XML Block

```
<PTZPatrol version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <PTZPatrolStatus>     <!—opt -->
        <patrolID> <!-- req, xs:string;id --> </patrolID>
        <patrolStatus> <!-- req, xs:string, "running,stopped,paused" --> </patrolStatus>
    </PTZPatrolStatus>
</PTZPatrol>
```

8.13.20 /PTZCtrl/channels/<ID>/patrols/<ID>/schedule

/PTZCtrl/channels/<ID>/schedule	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get patrol schedule of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	**TimeBlockList**
PUT	**Operator**
Description	It is used to update patrol schedule of a particular PTZ channel.
Query	None
Inbound Data	**TimeBlockList**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.21 /PTZCtrl/channels/<ID>/patterns

/PTZCtrl/channels/<ID>/patterns	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get pattern configuration of a particular PTZ channel.

Query	None
Inbound Data	None
Success Return	**PTZPatternList**
Notes: It is similar to presets!!	

PTZPatternList XML Block

```
<PTZPatternList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <PTZPattern>   <!-- opt -->
</PTZPatternList>
```

8.13.22 /PTZCtrl/channels/<ID>/patterns/<ID>

/PTZCtrl/channels/<ID>/patterns/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a particular pattern configuration of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZPattern**
PUT	**Operator**
Description	It is used to update a particular pattern configuration of a particular PTZ channel.
Query	None
Inbound Data	**PTZPattern**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a particular pattern configuration of a particular PTZ channel
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: <space> x% indicates the remaining space for pattern	

PTZPattern XML Block

```
<PTZPattern version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <enabled>   <!-- req, xs:boolean -->   </enabled>
  <id>    <!-- req, xs:integer -->    </id>
  <space>    <!-- req, xs:integer, 0..100-->     </space>
</PTZPattern>
```

8.13.23 /PTZCtrl/channels/<ID>/patterns/<ID>/recordstart

/PTZCtrl/channels/<ID>/patterns/<ID>/recordstart	**General Resource v1.0**
PUT	**Operator**
Description	It is used to start particular pattern information recording of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Remaining space information will be uploaded in real time during the recording process.	

8.13.24 /PTZCtrl/channels/<ID>/patterns/<ID>/recordstop

/PTZCtrl/channels/<ID>/patterns/<ID>/recordstop	**General Resource v1.0**
PUT	**Operator**
Description	It is used to stop a particular pattern information recording of a particular PTZ channel
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.25 /PTZCtrl/channels/<ID>/patterns/<ID>/run

/PTZCtrl/channels/<ID>/patterns/<ID>/run	**General Resource v1.0**
PUT	**Operator**
Description	It is used to start a particular pattern of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.26 /PTZCtrl/channels/<ID>/patterns/<ID>/stop

/PTZCtrl/channels/<ID>/patterns/<ID>/stop	**General Resource v1.0**

PUT	Operator
Description	It is used to stop a particular pattern which is in running status of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.13.27 /PTZCtrl/channels/<ID>/PTZOSDDisplay

/PTZCtrl/channels/<ID>/PTZOSDDisplay	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get OSD display information of a particular PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZOSDDisplay**
PUT	**Operator**
Description	It is used to update OSD display information of a particular PTZ channel.
Query	None
Inbound Data	**PTZOSDDisplay**
Success Return	ResponseStaus **ResponseStatus**
Notes: <zoomlable> indicates the zoom progress bar display <azimuth> indicates the azimuth display <presetlable> indicates the preset title display	

PTZOSDDisplay XML Block

```
<PTZOSDDisplay version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <zoomlable>
        <!-- req, xs:strings, "2sec, 5sec, 10sec, alwaysclose, alwaysopen"-->
    </zoomlable>
    <azimuth>
        <!-- req, xs:strings, "2sec, 5sec, 10sec, alwaysclose, alwaysopen"-->
    </azimuth>
    <presetlable>
        <!-- req, xs:strings, "2sec, 5sec, 10sec, alwaysclose, alwaysopen"-->
    </presetlable>
</PTZOSDDisplay>
```

8.13.28 /PTZCtrl/channels/<ID>/parkaction

/PTZCtrl/channels/<ID>/parkaction	General Resource v1.0
GET	**Viewer**
Description	It is used to get park action information of a PTZ channel.
Query	None
Inbound Data	None
Success Return	**ParkAction**
PUT	**Operator**
Description	It is used to update park action information of a PTZ channel.
Query	None
Inbound Data	**ParkAction**
Success Return	ResponseStaus **ResponseStatus**
Notes: <Parktime> Time span that will trigger an park action <Action> park action <ActionNum> park action number. It is used when park action is patrol, pattern or preset. For others, it is 0	

ParkAction XML Block

```
<ParkAction version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled>   <!-- req, xs:boolean -->   </enabled>
   <Parktime>   <!-- req, xs:integer, seconds -->   </Parktime>
    <Action>
        <ActionType>
            <!-- req, xs:strings, "atuoscan, framescan, randomscan, panoramascan,
    patrol, pattern, preset" -->
        </ActionType>
        <ActionNum>    <!-- req, xs:integer, 0..255-->      </ActionNum>
    </Action>
</ParkAction>
```

8.13.29 /PTZCtrl/channels/<ID>/ptzlimiteds

/PTZCtrl/channels/<ID>/ptzlimiteds	General Resource v1.0
GET	**Viewer**
Description	It is used to get movement limitations of PTZ channels.
Query	None
Inbound Data	None
Success Return	**PTZLimitedList**
Notes:	

PTZLimitedList XML Block

```
<PTZLimitedList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <PTZLimited>   <!-- opt -->
</PTZLimitedList>
```

8.13.30 /PTZCtrl/channels/<ID>/ptzlimiteds/<ID>

/PTZCtrl/channels/<ID>/ptzlimiteds/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get movement limitations of a PTZ channel.
Query	None
Inbound Data	None
Success Return	**PTZLimited**
PUT	**Operator**
Description	It is used to update movement limitations of a PTZ channel.
Query	None
Inbound Data	**PTZLimited**
Success Return	ResponseStaus **ResponseStatus**
DELETE	
Description	It is used to clear movement limitations of a PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: It is used to get or set the parameter that whether movement limitation is enabled or disabled. Speed dome add two types of movement limitation. <ID>=1 Manual control movement limitation <ID>=2 Panorama scan movement limitation	

PTZLimited XML Block

```
<PTZLimited version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
      <enabled>   <!-- req, xs:boolean -->   </enabled>
      <id> <!-- req, xs:string;id --> </id>
</PTZLimited>
```

8.13.31 /PTZCtrl/channels/<ID>/ptzlimiteds/<ID>/setstart

/PTZCtrl/channels/<ID>/ptzlimiteds/<ID>/setstart	**General Resource v1.0**
PUT	**Operator**
Description	Set the start position of a movement limitation of a PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Only used when movement limitation is enabled.	

8.13.32 /PTZCtrl/channels/<ID>/ptzlimiteds/<ID>/set

/PTZCtrl/channels/<ID>/ptzlimiteds/<ID>/set	**General Resource v1.0**
PUT	**Operator**
Description	Set other positions of a movement limitation of a PTZ channel.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Only used when movement limitation is enabled and **setstart** already been used. Order of the positions is left→right→up→down. Please save the settings after setup.	

8.13.33 /PTZCtrl/channels/<ID>/saveptzpoweroff

/PTZCtrl/channels/<ID>/saveptzpoweroff	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the PTZ power off memory settings information
Query	None
Inbound Data	None
Success Return	**PTZChannel**
PUT	**Operator**
Description	It is used to update the PTZ power off memory settings information
Query	None
Inbound Data	**PTZChannel**
Success Return	ResponseStaus **ResponseStatus**

Notes:
<savePtzPoweroff>Power off memory

savePtzPoweroff XML Block

```
<savePtzPoweroff version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <savePtzPoweroffType>
         <!-- req, xs:integer, "disable, 30sec, 60sec, 300sec, 600sec"-->
   </savePtzPoweroffType>
</savePtzPoweroff>
```

8.13.34 /PTZCtrl/channels/<ID>/timetasks

/PTZCtrl/channels/<ID>/timetasks	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a list of tasks based on a schedule
Query	None
Inbound Data	None
Success Return	TimeTaskList
PUT	**Operator**
Description	It is used to update a list of tasks based on a schedule
Query	None
Inbound Data	TimeTaskList
Success Return	ResponseStaus **ResponseStatus**
Notes: GET is used to get a list of tasks of a whole week(7) <enabled>Enable all the tasks <Parktime> Time span for a task to resume.	

EventSchedule XML Block

```
<TimeTaskList version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled>   <!-- req, xs:boolean -->   </enabled>
   <Parktime>    <!-- req, xs:integer, seconds -->    </Parktime>
   <TimeTaskBlock /> <!-- opt -->
</TimeTaskList>
```

8.13.35 /PTZCtrl/channels/<ID>/timetasks/<ID>

/PTZCtrl/channels/<ID>/timetasks/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a list of tasks of one day
Query	None

Inbound Data	None
Success Return	**TimeTaskBlock**
PUT	**Operator**
Description	It is used to update a list of tasks of one day
Query	None
Inbound Data	**TimeTaskBlock**
Success Return	ResponseStaus **ResponseStatus**
DELETE	**Operator**
Description	It is used to delete a list of tasks of one day
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**

Notes:

Tasks based on a schedule consist of time blocks ad tasked. This task is enabled always.

<TimeTaskBlock> get all the time span and tasks of one day

<dayOfWeek> specify the day of a week, ranging from 1 to 7

<TimeTaskRange> time span of each task. Up to ten time spans and 10 tasks are supported in one day.

<beginDateTime> specify the begin time of each task, ranig from 0:0:0-23:59:00, format is consistent to ISO 8601.

<endDateTime> specify the end time of each task, ranig from 0:0:0-23:59:00, format is consistent to ISO 8601. endDateTime should be larger than or equal to beginDateTime.

<TaskType> Tasks type

<TaskNum> Tasks number. Enabled when park action is patrol, pattern, preset or auxoutput, otherwise the value is 0.

TimeTaskBlock XML Block

```
<TimeTaskBlock version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <dayOfWeek>
      <!-- req, xs:integer, ISO8601 weekday number, 1=Monday, … -->
   </dayOfWeek>
   <TimeTaskRange>
        <TaskID><!-- req, xs:string;id --></TaskID>
        <beginTime> <!-- req, xs:time, ISO8601 time --> </beginTime>
        <endTime> <!-- req, xs:time, ISO8601 time --> </endTime>
        <Task>
            <TaskType>
              <!-- req, xs:strings, "disable, atuoscan, framescan, randomscan, panoramascan,
patrol, pattern, preset, tiltscan,periodreboot,periodadjust,auxoutput" -->
           </TaskType>
           <TaskNum><!-- dep, xs:integer, 0.8--></TaskNum>
        </Task>
   </TimeTaskRange>
```

```
</TimeTaskBlock>
```

8.13.36 /PTZCtrl/channels/<ID>/timetasks/<ID>/copytask

/PTZCtrl/channels/<ID>/timetasks/<ID>/copytask	General Resource v1.0
GET	**Viewer**
Description	It is used to get the default copy time of a tasks list of a specified PTZ channel.
Query	None
Inbound Data	None
Success Return	TimeTaskCopy
PUT	**Operator**
Description	It is used to update the default copy time of a tasks list of a specified PTZ channel.
Query	None
Inbound Data	TimeTaskCopy
Success Return	ResponseStaus **ResponseStatus**
Notes: <curDayOfWeek> specify the current day of a week； <copyDayOfWeek> specify the days that will have the same settings as the current day；	

EventSchedule XML Block

```
<TimeTaskCopy version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <curDayOfWeek>
       <!-- req, xs:integer, ISO8601 weekday number, 1=Monday, … -->
   </curDay>
   <copyDayOfWeek>
       <!-- req, xs:integer, ISO8601 weekday number, 1=Monday, … -->
   </copyDay>
</TimeTaskCopy>
```

8.13.37 /PTZCtrl/channels/<ID>/auxcontrol

/PTZCtrl/channels/<ID>/auxcontrol	General Resource v1.0
GET	**Viewer**
Description	It is used to get auxillary PTZ control information of a specified PTZchannel.
Query	command

Inbound Data	None
Success Return	**PTZAuxStatus**
PUT	**Operator**
Description	It is used to update auxillary PTZ control information of a specified PTZchannel.
Query	command
Inbound Data	**PTZAuxStatus**
Success Return	ResponseStaus **ResponseStatus**
Notes: Auxillary PTZ functions: Commands: LIGHT_PWRON: open light WIPER_PWRON: turn on wiper FAN_PWRON: turn on fun HEATER_PWRON: turn on heater <enabled> 1 means turned on, 0 means turned off.	

PTZAuxStatus XML Block

```
<PTZAuxStatus version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <enabled>   <!-- req, xs:boolean -->   </enabled>
</PTZAuxStatus>
```

8.14 Image

/Image	**Service v1.0**
Notes: service of camera Image	

8.14.1 /Image/channels

/Image/channels	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the list of channel Image configuration.
Query	None
Inbound Data	None
Success Return	**ImageChannellist**
PUT	**Operator**
Description	It is used to update Image configuration for all channels.
Query	None
Inbound Data	**ImageChannellist**

Success Return	ResponseStaus **ResponseStatus**
Notes:	

ImageChannellist XML Block

```
<ImageChannellist version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <ImageChannel/>    <!--opt-->
</ImageChannellist>
```

8.14.2 /Image/channels/<ID>

/Image/channels/<ID>	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get a special channel Image configuration.
Query	None
Inbound Data	None
Success Return	**ImageChannel**
PUT	**Operator**
Description	It is used to update Image configuration for a special channel.
Query	None
Inbound Data	**ImageChannel**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

ImageChannellist XML Block

```
<ImageChannel version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
  <id><!-- req, xs:integer --></id>
  <enabled>    <!-- req, xs:boolean -->    </enabled>
  <videoInputID>          <!-- req, xs:integer -->    </videoInputID>
  <resetImage/><!-- opt -->
  <restoreImageparam/> <!-- opt -->
  <Focus/> <!-- opt -->
  <LensInitialization /> <!-- opt -->
  <ImageFilp/> <!-- opt -->
  <ImageFreeze/> <!-- opt -->
  <proportionalpan/> <!-- opt -->
  <WDR/> <!-- opt -->
  <BLC/> <!-- opt -->
  <NoiseReduce/> <!-- opt -->
  <ImageEnhancement/> <!-- opt -->
  <IrcutFilter/> <!-- opt -->
  <DSS/> <!-- opt -->
  <WhiteBlance/> <!-- opt -->
  <Exposure/> <!-- opt -->
  <Sharpness/> <!-- opt -->
  <Iris/> <!-- opt -->
  <Shutter/> <!-- opt -->
  <Gain/> <!-- opt -->
  <gamaCorrection/> <!-- opt -->
  <powerLineFrequency/> <!-- opt -->
  <Color/> <!-- opt -->
</ImageChannel>
```

8.14.3 /Image/channels/<ID>/resetImage

/Image/channels/<ID>/resetImage	**General Resource v1.0**
PUT	**Operator**
Description	It is used to reset an image channel (cut off the power and reboot the speed dome).
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes: Image reset only reboot the camera unit.	

8.14.4 /Image/channels/<ID>/restoreImageparam

/Image/channels/<ID>/restoreImageparam	**General Resource v1.0**
PUT	**Operator**
Description	It is used to reset the image configure parameter to the factory default.
Query	None
Inbound Data	None
Success Return	ResponseStaus **ResponseStatus**
Notes:	

8.14.5 /Image/channels/<ID>/Focus

/Image/channels/<ID>/Focus	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get focus parameters of a specified image channel.

Query	None
Inbound Data	None
Success Return	**Focus**
PUT	**Operator**
Description	It is used to update focus parameters of a specified image channel.
Query	None
Inbound Data	**Focus**
Success Return	ResponseStaus **ResponseStatus**
Notes: AUTO: auto focus MANUAL: manual focus SEMIAUTOMATIC: seni automatic FocusValue's PUT operator is enabled only when FocusStyle's value is MANUAL. focusSpeed: focus vector data. Negative numbers focus near, positive numbers focus far. Numerical value is a percentage of the maximum focus speed of the lens module.	

Focus XML Block

```
<Focus version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <FocusStyle/>    <!-- req, xs:string, "AUTO, MANUAL, SEMIAUTOMATIC" -->
  <FocusLimited/>    <!-- req, xs:string, "1cm, 30cm, 1m, 3m" -->
  <FocusValue/>    <!-- optdep,depends on <FocusStyle>, xs:integer-->
  <focusSpeed>    <!-- opt, xs:intger, -100..100 --> </focusSpeed>
</Focus>
```

8.14.6 /Image/channels/<ID>/LensInitialization

/Image/channels/<ID>/LensInitialization	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the initizlization status of the lens of a specified image channel.
Query	None
Inbound Data	None
Success Return	**LensInitialization**
PUT	**Operator**
Description	It is used to update focus parameters of a specified image channel.
Query	None
Inbound Data	**LensInitialization**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

LensInitialization XML Block

```
<LensInitialization version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/>                    <!-- req, xs:boolean -->
</LensInitialization>
```

8.14.7 /Image/channels/<ID>/ImageFlip

/Image/channels/<ID>/ImageFlip	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the mirror status of a specified image channel.
Query	None
Inbound Data	None
Success Return	**ImageFlip**
PUT	**Operator**
Description	It is used to update mirror status of a specified image channel.
Query	None
Inbound Data	**ImageFlip**
Success Return	ResponseStaus **ResponseStatus**
Notes: ImageFlipStyle is enabled only when enabled value is true.	

ImageFlip XML Block

```
<ImageFlip version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <enabled/>                    <!--req, xs:boolean -->
  <ImageFlipStyle/>    <!--opt, xs:string, "LEFTRIGHT, UPDOWN, CENTER" -->
</ImageFlip>
```

8.14.8 /Image/channels/<ID>/ImageFreeze

/Image/channels/<ID>/ImageFreeze	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get ImageFreeze status of a specified Image channel.
Query	None
Inbound Data	None
Success Return	**ImageFreeze**
PUT	**Operator**
Description	It is used to update ImageFreeze status of a specified image channel.
Query	None
Inbound Data	**ImageFreeze**
Success Return	ResponseStaus **ResponseStatus**

Notes:

ImageFreeze XML Block

```
<ImageFreeze version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/>                    <!-- req, xs:boolean -->
</ImageFreeze>
```

8.14.9 /Image/channels/<ID>/proportionalpan

/Image/channels/<ID>/proportionalpan	General Resource v1.0
GET	**Viewer**
Description	It is used to get proportional pan status of a specified image channel.
Query	None
Inbound Data	None
Success Return	**proportionalpan**
PUT	**Operator**
Description	It is used to update proportional pan status of a specified image channel.
Query	None
Inbound Data	**proportionalpan**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

proportionalpan XML Block

```
<ImageFlip version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <enabled/>                   <!--req, xs:boolean -->
  <ImageFlipStyle/>    <!--opt, xs:string, "LEFTRIGHT, UPDOWN, CENTER" -->
</ImageFlip>
```

8.14.10 /Image/channels/<ID>/WDR

/Image/channels/<ID>/WDR	General Resource v1.0
GET	**Viewer**
Description	It is used to get the value of wide dynamic range for a specified Image channel.
Query	None
Inbound Data	None
Success Return	**WDR**
PUT	**Operator**

Description	It is used to configure the value of wide dynamic range for a specified Image channel.
Query	None
Inbound Data	**WDR**
Success Return	ResponseStaus **ResponseStatus**
Notes: The range of WDRLevel's value is needed according to the capbilites of devices.	

WDR XML Block

```
<WDR version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/>       <!-- req, xs:boolean -->
   <WDRLevel/><!--opt,xs:string,"0,1,2...100,B0,B1,B2...B100"-->
   <WDRContrastLevel/>   <!--opt, xs:integer, "0--100" -->
</WDR>
```

8.14.11 /Image/channels/<ID>/BLC

/Image/channels/<ID>/BLC	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the configuration of backlight compensation for a specified image channel.
Query	None
Inbound Data	None
Success Return	**BLC**
PUT	**Operator**
Description	It is used to configure the configuration of backlight compensation for a specified image channel.
Query	None
Inbound Data	**BLC**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

BLC XML Block

```
<BLC version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/>       <!-- req, xs:boolean -->
   <BLCMode/>   <!--opt, xs:string, "UP, DOWN, LEFT, RIGHT, CENTER" -->
   <BLCLevel/>   <!--opt, xs:integer, "0--100" -->
</BLC>
```

8.14.12 /Image/channels/<ID>/Imageenhancement

/Image/channels/<ID>/Imageenhancement	**General Resource v1.0**

GET	Viewer
Description	It is used to get the ImageEnhancement's configuration of a specified image channel.
Query	None
Inbound Data	None
Success Return	**ImageEnhancement**
PUT	**Operator**
Description	It is used to configure the ImageEnhancement's configuration of a specified image channel.
Query	None
Inbound Data	**ImageEnhancement**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

Imageenhancement XML Block

```
<ImageEnhancement version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/> <!-- req, xs:boolean -->
   <ImageEnhancementLevel> <!—opt, xs:string, "low, normal, high" -->
</ImageEnhancement>
```

8.14.13 /Image/channels/<ID>/IrcutFilter

/Image/channels/<ID>/IrcutFilter	General Resource v1.0
GET	**Viewer**
Description	It is used to get the IrcutFilter's configuration of a specified image channel.
Query	None
Inbound Data	None
Success Return	**IrcutFilter**
PUT	**Operator**
Description	It is used to configure the IrcutFilter's configuration of a specified image channel.
Query	None
Inbound Data	**IrcutFilter**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

IrcutFilter XML Block

```
<IrcutFilter version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <IrcutFilterType/>     <!-- opt, xs:string, " auto, day, night,"-->
   <IrcutFilterLevel/>     <!—opt, xs:string, "low, normal, high" -->
   <IrcutFilterTime/>       <!—opt xs:integer -->
```

```
</IrcutFilter>
```

8.14.14 /Image/channels/<ID>/NoiseReduce

/Image/channels/<ID>/NoiseReduce	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the NoiseReduce's value of a specified image channel.
Query	None
Inbound Data	None
Success Return	**NoiseReduce**
PUT	**Operator**
Description	It is used to configure the NoiseReduce's value of a specified image channel.
Query	None
Inbound Data	**NoiseReduce**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

NoiseReduce XML Block

```
<NoiseReduce version=“1.0” xmlns=“http://www.std-cgi.com/ver10/XMLSchema”>
   <enabled/>                <!-- req, xs:boolean -->
   <NoiseReduceLevel> <!—opt, xs:string, ”low, normal, high” -->
</NoiseReduce>
```

8.14.15 /Image/channels/<ID>/DSS

/Image/channels/<ID>/DSS	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the the configuration of digital slow shutter for a specified Image channel.
Query	None
Inbound Data	None
Success Return	**DSS**
PUT	**Operator**
Description	It is used to configure the configuration of digital slow shutter for a specified Image channel.

Query	None
Inbound Data	**DSS**
Success Return	ResponseStaus **ResponseStatus**
Notes: DSSLevel is only enabled when enabled value is true.	

DSS XML Block

```
<DSS version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <enabled/>                  <!-- req, xs:boolean -->
   <DSSLevel/>    <!—opt, xs:string, "low, normal, high" -->
</DSS>
```

8.14.16 /Image/channels/<ID>/WhiteBlance

/Image/channels/<ID>/WhiteBlance	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the WhiteBlance value of a specified iImage channel.
Query	None
Inbound Data	None
Success Return	**WhiteBlance**
PUT	**Operator**
Description	It is used to configure the WhiteBlance value of a specified iImage channel.
Query	None
Inbound Data	**WhiteBlance**
Success Return	ResponseStaus **ResponseStatus**
Notes: WhiteBlanceRed and WhiteBlanceBlue's PUT operator is enabled only when WhiteBlanceStyle's value is manual.	

WhiteBlance XML Block

```
<WhiteBlance version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <WhiteBlanceStyle/> <!-- req, xs:string, "auto, manual, indoor, outdoor, autotrace, onece"
                -->
   <WhiteBlanceRed/>    <!--dep, depends on <WhiteBlanceStyle>,xs:integer,"0--100"    -->
   <WhiteBlanceBlue/>    <!--dep, depends on <WhiteBlanceStyle>,xs:integer,"0--100" -->
</WhiteBlance>
```

8.14.17 /Image/channels/<ID>/Exposure

/Image/channels/<ID>/Exposure	**General Resource v1.0**

GET	Viewer
Description	It is used to get the exposure mode of a specified image channel.
Query	None
Inbound Data	None
Success Return	**Exposure**
PUT	**Operator**
Description	It is used to configure the exposure mode of a specified image channel.
Query	None
Inbound Data	**Exposure**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

Exposure XML Block

```
<Exposure version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <ExposureType/>   <!--req, xs:string, "auto, IrisFirst, ShutterFirst, gainFirst, manual" -->
</Exposure>
```

8.14.18 /Image/channels/<ID>/Sharpness

/Image/channels/<ID>/Sharpness	General Resource v1.0
GET	**Viewer**
Description	It is used to get the sharpness's value of a specified image channel.
Query	None
Inbound Data	None
Success Return	**Sharpness**
PUT	**Operator**
Description	It is used to configure the sharpness's value of a specified image channel.
Query	None
Inbound Data	**Sharpness**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

Sharpness XML Block

```
<Sharpness version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <SharpnessLevel/>   <!--req, xs:integer,"0--100" -->
</Sharpness>
```

8.14.19 /Image/channels/<ID>/Iris

/Image/channels/<ID>/Iris	General Resource v1.0
GET	**Viewer**
Description	It is used to get the iris's value of a specified image channel.
Query	None
Inbound Data	None
Success Return	**Iris**
PUT	**Operator**
Description	It is used to configure the iris's value of a specified image channel.
Query	None
Inbound Data	**Iris**
Success Return	ResponseStaus **ResponseStatus**
Notes: Iris's PUT operate is enabled only when <ExposureType> is IrisFirst irisSpeed: negative numbers close iris, positive numbers open iris. Numerical value is a percentage of the maximum iris speed of the lens module.	

IrisValue XML Block

```
<Iris version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
   <IrisLevel/>
       <!--dep, depends on <ExposureType>, xs:string, "f1.4, f1.6, f2.0, f2.4, f2.8, f3.4, f4.0,
          f4.8, f5.6, f6.8, f8.0, f9.6, f11, f14, f16, f19, f22" -->
   <irisSpeed>     <!-- opt, xs:integer, -100..100 -->      </irisSpeed>
</Iris>
```

8.14.20 /Image/channels/<ID>/Shutter

/Image/channels/<ID>/Shutter	General Resource v1.0
GET	**Viewer**
Description	It is used to get the Shutter value of a specified image channel.
Query	None
Inbound Data	None
Success Return	**Shutter**
PUT	**Operator**
Description	It is used to configure the Shutter value of a specified image channel.
Query	None
Inbound Data	**Shutter**
Success Return	ResponseStaus **ResponseStatus**
Notes: Shutter's PUT operate is enabled only when <ExposureType> is ShutterFirst	

ShutterValue XML Block

```
<Shutter version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
  <ShutterLevel/>
    <!--dep,depends on <ExposureType>, xs:string, "1/1, 1/2, 1/3, 1/6, 1/12, 1/25, 1/50,
      1/75, 1/100, 1/120, 1/150, 1/215, 1/300, 1/425, 1/600, 1/1000, 1/1250, 1/1750,
      1/2500, 1/3500, 1/6000, 1/10000" -->
</Shutter>
```

8.14.21 /Image/channeles/<ID>/Gain

/Image/channels/<ID>/Gain	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the gain configuration of a specified Image channel.
Query	None
Inbound Data	None
Success Return	**Gain**
PUT	**Operator**
Description	It is used to configure the gain configuration of a specified Image channel.
Query	None
Inbound Data	**Gain**
Success Return	ResponseStaus **ResponseStatus**
Notes: Gain's PUT operate is enabled only when <ExposureType> is gainFirst.	

gain XML Block

```
<Gain version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <GainLevel/>   <!--dep,depends on <ExposureType>, xs:integer,"0--100"   -- -->
    <GainWindow><!-- opt -->
    <RegionCoordinatesList> <!-- opt -->
    <RegionCoordinates><!-- opt -->
    <positionX><!-- req, xs:integer;coordinate --></positionX>
    <positionY><!-- req, xs:integer;coordinate --></positionY>
    </RegionCoordinates>
    </RegionCoordinatesList>
    </GainWindow>
</Gain>
```

8.14.22 /Image/channeles/<ID>/GamaCorrection

/Image/channels/<ID>/gamaCorrection	**General Resource v1.0**
GET	**Viewer**

Description	It is used to get the gama correction of a specified Image channel.
Query	None
Inbound Data	None
Success Return	**gammaCorrection**
PUT	**Operator**
Description	It is used to configure the gama correction of a specified Image channel.
Query	None
Inbound Data	**gammaCorrection**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

gammaCorrection XML Block

```
<gammaCorrection version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
    <gammaCorrectionEnabled> <!-- opt, xs:boolean --> </gammaCorrectionEnabled>
    <gammaCorrectionLevel> <!-- opt, xs:integer, 0--100 --> </gammaCorrectionLevel>
</gammaCorrection>
```

8.14.23 /Image/channels/<ID>/powerLineFrequency

/Image/channels/<ID>/powerLineFrequency	**General Resource v1.0**
GET	**Viewer**
Description	It is used to get the powerLineFrequency value of a specified Image channel.
Query	None
Inbound Data	None
Success Return	**powerLineFrequency**
PUT	**Operator**
Description	It is used to configure the powerLineFrequency value of a specified Image channel.
Query	None
Inbound Data	**powerLineFrequency**
Success Return	ResponseStaus **ResponseStatus**
Notes: Configure the powerlineFrequency requires to reboot the camera.	

powerlineFrequency XML Block

```
<powerLineFrequency version="1.0" mlns="http://www.std-cgi.com/ver10/XMLSchema">
      <powerLineFrequencyMode>   <!-- opt, xs:string "50hz, 60hz" -->
     </powerLineFrequencyMode>
</powerLineFrequency>
```

8.14.24 /Image/channels/<ID>/Color

/Image/channels/<ID>/Color	General Resource v1.0
GET	**Viewer**
Description	It is used to get the color's value of a specified Image channel.
Query	None
Inbound Data	None
Success Return	**Color**
PUT	**Operator**
Description	It is used to configure the color's value of a specified Image channel.
Query	None
Inbound Data	**Color**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

color XML Block

```
<Color version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
        <brightnessLevel>          <!-- opt, xs:integer, 0--100 -->       </brightnessLevel>
        <contrastLevel>          <!-- opt, xs:integer, 0--100 -->       </contrastLevel>
        <saturationLevel>          <!-- opt, xs:integer, 0--100 -->       </saturationLevel>
        <hueLevel><!-- opt, xs:integer, 0--100 -->       </hueLevel>
</Color>
```

8.15 /Record

/Record	Service v1.0
Notes: service of Recording	

8.15.1 /Record/Schedule

/Record/schedule	
GET	**Viewer**
Description	It is used to get recording time range.
Query	None
Inbound Data	None
Success Return	**RecordSchedule**
PUT	**Operator**

Description	It is used to update recording time range.
Query	None
Inbound Data	**RecordSchedule**
Success Return	ResponseStaus **ResponseStatus**
Notes:	

RecordSchedule XML Block

```
<RecordSchedule version="1.0" xmlns="http://www.std-cgi.com/ver10/XMLSchema">
<enalbled><!-- req, xs:boolean --> <enalbled/>
<RecordDelayTime><!-- req, xs:integer   --></RecordDelayTime>
<PreRecordTime><!-- req, xs:integer --></PreRecordTime>
<TimeBlockList> <!-- req -->
<TimeBlock>
   <recordType> <!-- req, xs:string,"Alarm,Motion,Timing,"--></recordType>
          <dayOfWeek>

            <!-- opt, xs:integer, ISO8601 weekday number, 1=Monday,  …  -->

          </dayOfWeek>
          <TimeRange>          <!-- req -->
            <beginTime>      <!-- req, xs:time, ISO8601 time -->   </beginTime>
            <endTime>      <!-- req, xs:time, ISO8601 time -->   </endTime>
        </TimeRange>
    </TimeBlock>
</TimeBlockList>
</RecordSchedule>
```

Annex A (normative):

XML Schema Definition

A.0 std-cgi.xsd

The following XML Schema Document contains XML schema definitions for data structures in this specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:std-cgi="http://www.std-cgi.com/ver10/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:xlink=”http://www.w3.org/1999/xlink”
targetNamespace=”http://www.std-cgi.com/ver10/XMLSchema”
elementFormDefault=”qualified”>
<xs:import namespace=”http://www.w3.org/1999/xlink” schemaLocation=”xlink.xsd”/>
<xs:annotation>
   <xs:documentation>
      STD-CGI Core XML Schema
   </xs:documentation>
</xs:annotation>
<!-- ======================== -->
<!--             Resource Types             -->
<!-- ======================== -->
<xs:simpleType name=”ResourceType”>
   <xs:restriction base=”xs:string”>
      <xs:enumeration value=”Special Resource” />
      <xs:enumeration value=”Service”/>
      <xs:enumeration value=”General Resource” />
   </xs:restriction>
</xs:simpleType>
<!-- ======================== -->
<xs:complexType name=”QueryParameter”>
   <xs:sequence>
      <xs:element name=”name” type=”xs:string” />
      <xs:element name=”type” type=”xs:string” />
      <xs:element name=”description” type=”xs:string” minOccurs=”0” maxOccurs=”1” />
   </xs:sequence>
<xs:complexType>
<!-- ======================== -->
<xs:complexType name=”QueryParameterList”>
   <xs:sequence>
      <xs:element name=”queryParameter” type=” QueryParameter” minOccurs=”0”
           maxOccurs=”unbounded” />
   </xs:sequence>
<xs:complexType>
<!-- ======================== -->
<xs:complexType name=”OperationParameter”>
   <xs:sequence>
      <xs:element name=”description” type=”xs:string” />
      <xs:element name=”queryParameterList” type=” QueryParameterList” />
      <xs:element name=”inboundData” type=”xs:string” />
      <xs:element name=”successReturn” type=”xs:string” />
   </xs:sequence>
<xs:complexType>
<!-- ======================== -->
```

```
<xs:complexType name=”ResourceDescription”>
  <xs:sequence>
    <xs:element name=”name” type=”xs:string” />
    <xs:element name=”version” type=”xs:string” />
    <xs:element name=”type” type=” ResourceType” />
    <xs:element name=”get” type=” OperationParameter” minOccurs=”0”
     maxOccurs=”1” />
    <xs:element name=”put” type=” OperationParameter” minOccurs=”0”
     maxOccurs=”1” />
    <xs:element name=”post” type=” OperationParameter” minOccurs=”0”
     maxOccurs=”1” />
    <xs:element name=”delete” type=” OperationParameter” minOccurs=”0”
     maxOccurs=”1” />
    <xs:element name=”notes” type=”xs:string” minOccurs=”0”
     maxOccurs=”1” />
  </xs:sequence>
  <xs:attribute name=”version” type=”xs:string” use=”required” />
<xs:complexType>
<!-- ======================= -->
<xs:complexType name=”Resource”>
  <xs:sequence>
    <xs:element name=”name” type=”xs:string” />
    <xs:element name=”version” type=”xs:string” />
    <xs:element name=”type” type=” ResourceType” />
    <xs:element name=”description” type=”xs:string” minOccurs=”0”
     maxOccurs=”1” />
    <xs:element name=”ResourceList” type=” ResourceList” minOccurs=”0”
     maxOccurs=”1” />
  </xs:sequence>
  <xs:attribute name=”version” type=”xs:string” use=”required” />
<xs:complexType>
<!-- ======================= -->
<xs:complexType name=”ResourceList”>
  <xs:sequence>
    <xs:element name=”Resource” type=” Resource” minOccurs=”0”
     maxOccurs=”unbounded”/>
  </xs:sequence>
  <xs:attribute name=”version” type=”xs:string” use=”required” />
<xs:complexType>
<!-- ======================= -->
<!--     ResponseStatus Types              -->
<!-- ======================= -->
<xs:simpleType name=”StatusCode”>
  <xs:restriction base=”xs:integer”>
```

```
        <xs:minInclusive value=”1” />
        <xs:maxInclusive value=”7” />
      </xs:restriction>
      <!-- 1-OK, 2-Device Busy, 3-Device Error, 4-Invalid Operation, 5-Invalid XML Format,
        6-Invalid XML Content, 7-Reboot Required -->
  </xs:simpleType>
  <!-- ======================= -->
  <xs:simpleType name=”ID”>
      <xs:restriction base=”xs:integer”>
        <xs:minInclusive value=”1” id=”id.minInclusive” />
      </xs:restriction>
  </xs:simpleType>
  <!-- ======================= -->
  <xs:complexType name=”ResponseStatus”>
      <xs:sequence>
        <xs:element name=”requestURL” type=”xs:anyURI” />
        <xs:element name=”statusCode” type=” StatusCode” />
        <xs:element name=”statusString” type=”xs:string” />
        <xs:element name=”id” type=” ID” minOccurs=”0” maxOccurs=”1” />
      </xs:sequence>
      <xs:attribute name=”version” type=”xs:string” use=”required” />
  </xs:complexType>
</xs:schema>
```

Notes:

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

Annex A (normative):

XML Schema Definition

RTSP protocol explanation

（1）URL address format: rtsp://<ipaddress>/<videotype>/ch<number>/<streamtype> /av_stream. Of these, ipaddress is the IP address of device, videotype is mpeg4 or h.264, number is numeric, streamtype is main or sub stream. It is not sensitive to the capital letter or lowercase letter.

For example: rtsp://192.0.1.100/mpeg4/ch1/main/av_stream /*the video type is MPEG4 and it comes from main stream of device's channel 1 which IP address is 192.0.1.100*/

（2）the explanation of authentication

Support URL carry with user name and password.

Support authentication in Describe step.

Annex B:

RTSP protocol explanation

（1）URL address format: rtsp://<ipaddress>/<videotype>/ch<number>/<streamtype>/av_stream. Of these, ipaddress is the IP address of device, videotype is mpeg4 or h.264, number is numeric, streamtype is main or sub stream. It is not sensitive to the capital letter or lowercase letter.

For example: rtsp://192.0.1.100/mpeg4/ch1/main/av_stream /*the video type is MPEG4 and it comes from main stream of device’s channel 1 which IP address is 192.0.1.100*/

（2）the explanation of authentication

Support URL carry with user name and password.

Support authentication in Describe step.