Physical Security Interoperability Alliance
IP Media Device API Specification
Version 1.1
Revision 1
12 November 2009

Revision History	Description	Date	By
Version 1.0 Revision 1	Initial version	2008-09-11	PSIA IP Video Group
Version 1.0 Revision 2	Removed login/logout, 'getRTSPToken'; fixed numbering (section 15/16) …	2008-12-11	PSIA IP Video Group
Version 1.0 Revision 3	Conversion to REST model	2009-02-16	PSIA IP Video Group
Version 1.0 Revision 4	Addition of XML data and minimal device requirements	2009-02-16	PSIA IP Video Group
Version 1.0 Revision 5	First reviewable draft	2009-02-16	PSIA IP Video Group
Version 1.0 Revision 6	Incorporation of comments from first review round, pulled more content from 0.9 draft specification. Added Scope section	2009-03-07	PSIA IP Video Group
Version 1.0 Revision 7	Corrections, expanded examples.	2009-03-13	PSIA IP Video Group
Version 1.1 Revision 1	Modified required services, resources, and fields. Corrections.	2009-11-12	PSIA IP Video Group

Disclaimer

Contents

1. Overview

This document specifies an interface that enables physical security and video management systems to communicate with various IP media devices in a standardized way. This eliminates the need for device driver customization in order to achieve interoperability among products from different manufacturers. The intent of this specification is to improve the interoperability of IP-based physical security products from different vendors.

2. Scope

As the first PSIA specification adhering to the PSIA Service Model, this document defines the mandatory PSIA services for *ALL PSIA specifications* (future PSIA specifications will reference this IP Media Device Specification for these mandatory services). In addition, it defines several services and subordinate resources that are specific to Media Devices.

All of the Mandatory Services and the Mandatory Resources of both Mandatory and Optional services are complete in this version of the specification. In contrast, several of the optional resources may undergo some changes in the next version of the specification based on lessons learned during implementation of this first version. These optional resources are considered preliminary and are indicated as such in the resource definition notes.

All of the services and resources under the Custom service are to be considered preliminary and there is a high probability that they will be moved into another service as and when applicable. Use of resources currently under the Custom service is not discouraged, as every attempt will be made to provide backward compatibility to these existing resources in subsequent PSIA specifications. For example, the Custom/Event/Notification services and resources are usable in their current form and might be retained as is but moved into a different service when a PSIA specification addressing events is published.

Suggestions for corrections to this version of the specification and additions to the protocol should be submitted to the PSIA Forum's IP Media Specification area located at:

http://www.psialliance.org/forums/

Please post your suggested correction or addition in an applicable thread.

3. Problem Definition

Security and/or network management applications require the ability to change configurations and control the behaviors of IP media devices – cameras, encoders, decoders, recorders, etc. This functionality can be achieved by sending a standard HTTP(S) request to the unit. The scope of this specification is to define all HTTP(S) application programming interfaces (APIs) for media devices and their functionality; namely, for setting/retrieving various configurations, and controlling device behaviors.

4. Conformance

This document conforms to the PSIA Service model, which describes the methods used for service discovery and introspection. The mandatory service and resources requirements defined by this model are implied in addition to any requirements defined herein.

The required services defined below are the fundamental services for all PSIA specifications and are intended to be referenced by other specifications.

The optional services defined are specific to IP media devices.

4.1. Service Requirements

The following table describes the service requirements of the PSIA Service Model.

REQ	Service URL	Notes
✓	/PSIA	
✓	/PSIA/System	
	/PSIA/System/Storage	Not all IP media devices support storage.
✓	/PSIA/System/Network	
	/PSIA/System/IO	
	/PSIA/System/Audio	
	/PSIA/System/Video	
	/PSIA/System/Serial	
	/PSIA/Diagnostics	
✓	/PSIA/Security	
	/PSIA/Security/AAA	
	/PSIA/Streaming	
	/PSIA/PTZ	
	/PSIA/Custom/MotionDetection	
	/PSIA/Custom/Event	

4.2. Resource Requirements

The following resources are required for the implemented services.

4.2.1. /PSIA Root Service

REQ	Command	GET	PUT	POST	DEL
✓	index	✓			
✓	indexr	✓			
✓	description	✓			

4.2.2. /PSIA/System

REQ	Command	GET	PUT	POST	DEL
✓	reboot		✓		
✓	updateFirmware		✓		
✓	configurationData	✓	✓		
✓	factoryReset		✓		
✓	deviceInfo	✓	✓		
	supportReport	✓			
✓	status	✓			

REQ	Command	GET	PUT	POST	DEL
	time	✓	✓		
	time/localTime	✓	✓		
	time/timeZone	✓	✓		
	time/ntpServers	✓	✓	✓	✓
	time/ntpServers/<ID>	✓	✓		✓
	logging	✓	✓		
	logging/messages	✓			

/PSIA/System/Storage

REQ	Command	GET	PUT	POST	DEL
	volumes	✓			
	volumes/<ID>	✓			
	volumes/<ID>/status	✓			
	volumes/<ID>/format		✓		
	volumes/<ID>/files	✓			✓
	volumes/<ID>/files/<ID>	✓			✓
	volumes/<ID>/files/<ID>/data	✓			

/PSIA/System/Network

REQ	Command	GET	PUT	POST	DEL
✓	interfaces	✓			
✓	interfaces/<ID>	✓	✓		
✓	interfaces/<ID>/ipAddress	✓	✓		
	interfaces/<ID>/wireless	✓	✓		
	interfaces/<ID>/ieee802.1x	✓	✓		
	interfaces/<ID>/ipFilter	✓	✓		
	interfaces/<ID>/ipFilter/filterAddresses	✓	✓	✓	✓
	interfaces/<ID>/ipFilter/filterAddresses/<ID>	✓	✓		✓
	interfaces/<ID>/snmp	✓	✓		
	interfaces/<ID>/snmp/v2c	✓	✓		
	interfaces/<ID>/snmp/v2c/trapReceivers	✓	✓	✓	✓
	interfaces/<ID>/snmp/v2c/trapReceivers/<ID>	✓	✓		✓
	interfaces/<ID>/snmp/advanced	✓	✓		
	interfaces/<ID>/snmp/advanced/users	✓	✓	✓	✓
	interfaces/<ID>/snmp/advanced/users/<ID>	✓	✓		✓
	interfaces/<ID>/snmp/advanced/notificationFilters	✓	✓	✓	✓

REQ	Command	GET	PUT	POST	DEL
	interfaces/<ID>/snmp/advanced/notificationFilters/<ID>	✓	✓		✓
	interfaces/<ID>/snmp/advanced/notificationReceivers	✓	✓	✓	✓
	interfaces/<ID>/snmp/advanced/notificationReceivers/<ID>	✓	✓		✓
	interfaces/<ID>/snmp/v3	✓	✓		
	interfaces/<ID>/qos	✓	✓		
	interfaces/<ID>/qos/cos	✓	✓	✓	✓
	interfaces/<ID>/qos/cos/<ID>	✓	✓		✓
	interfaces/<ID>/qos/dscp	✓	✓	✓	✓
	interfaces/<ID>/qos/dscp/<ID>	✓	✓		✓
✓	interfaces/<ID>/discovery	✓	✓		
	interfaces/<ID>/syslog	✓	✓		
	interfaces/<ID>/syslog/servers	✓	✓	✓	✓
	interfaces/<ID>/syslog/servers/<ID>	✓	✓		✓

/PSIA/System/IO

REQ	Command	GET	PUT	POST	DEL
	status	✓			
	inputs	✓			
	inputs/<ID>	✓	✓		
	inputs/<ID>/status	✓			
	outputs	✓			
	outputs/<ID>	✓	✓		
	outputs/<ID>/trigger		✓		
	outputs/<ID>/status	✓			

/PSIA/System/Audio

REQ	Command	GET	PUT	POST	DEL
	channels	✓			
	channels/<ID>	✓	✓		

/PSIA/System/Video

REQ	Command	GET	PUT	POST	DEL
	overlayImages	✓		✓	✓
	overlayImages/<ID>	✓	✓		✓
✓	inputs	✓			
✓	inputs/channels	✓			

REQ	Command	GET	PUT	POST	DEL
	overlayImages	✓		✓	✓
	overlayImages/<ID>	✓	✓		✓
✓	inputs/channels/<ID>	✓	✓		
	inputs/channels/<ID>/focus		✓		
	inputs/channels/<ID>/iris		✓		
	inputs/channels/<ID>/lens	✓			
	inputs/channels/<ID>/overlays	✓	✓		✓
	inputs/channels/<ID>/overlays/text	✓	✓	✓	✓
	inputs/channels/<ID>/overlays/text/<ID>	✓	✓		✓
	inputs/channels/<ID>/overlays/image	✓	✓	✓	✓
	inputs/channels/<ID>/overlays/image/<ID>	✓	✓		✓
	inputs/channels/<ID>/privacyMask	✓	✓		
	inputs/channels/<ID>/privacyMask/regions	✓	✓	✓	✓
	inputs/channels/<ID>/privacyMask/regions/<ID>	✓	✓		✓

/PSIA/System/Serial

REQ	Command	GET	PUT	POST	DEL
	ports	✓			
	ports/<ID>	✓	✓		
	ports/<ID>/command		✓		

4.2.3. /PSIA/Diagnostics

REQ	Command	GET	PUT	POST	DEL
	commands	✓		✓	✓
	commands/<ID>	✓			✓

4.2.4. /PSIA/Security

REQ	Command	GET	PUT	POST	DEL
	srtpMasterKey	✓	✓		
	deviceCertificate	✓	✓		

/PSIA/Security/AAA

REQ	Command	GET	PUT	POST	DEL
✓	users	✓	✓	✓	✓
✓	users/<ID>	✓	✓		✓

REQ	Command	GET	PUT	POST	DEL
	certificate	✓	✓		
	adminAccesses	✓	✓	✓	✓
	adminAccesses/<ID>	✓	✓		✓

4.2.5. /PSIA/Streaming

REQ	Command	GET	PUT	POST	DEL
✓	status	✓			
✓	channels	✓	✓	✓?	✓?
✓	channels/<ID>	✓	✓		✓?
✓	channels/<ID>/status	✓			
	channels/<ID>/http	✓			
	channels/<ID>/picture	✓			
	channels/<ID>/requestKeyFrame		✓		

4.2.6. /PSIA/PTZ

REQ	Command	GET	PUT	POST	DEL
	channels	✓	✓	✓?	✓?
	channels/<ID>	✓	✓		✓?
	channels/<ID>/homePosition		✓		
	channels/<ID>/continuous		✓		
	channels/<ID>/momentary		✓		
	channels/<ID>/relative		✓		
	channels/<ID>/absolute		✓		
	channels/<ID>/digital		✓		
	channels/<ID>/status	✓			
	channels/<ID>/presets	✓	✓	✓	✓
	channels/<ID>/presets/<ID>	✓	✓		✓
	channels/<ID>/presets/<ID>/goto		✓		
	channels/<ID>/patrols	✓	✓	✓	✓
	channels/<ID>/patrols/status	✓			
	channels/<ID>/patrols/<ID>	✓	✓		✓
	channels/<ID>/patrols/<ID>/start		✓		
	channels/<ID>/patrols/<ID>/stop		✓		
	channels/<ID>/patrols/<ID>/pause		✓		

REQ	Command	GET	PUT	POST	DEL
	channels/<ID>/patrols/<ID>/status	✓			
	channels/<ID>/patrols/<ID>/schedule	✓	✓		

4.2.7. /PSIA/Custom/MotionDetection

REQ	Command	GET	PUT	POST	DEL
		✓			
	<ID>	✓	✓		
	<ID>/regions	✓	✓	✓	✓
	<ID>/regions/<ID>	✓	✓		✓

4.2.8. /PSIA/Custom/Event

REQ	Command	GET	PUT	POST	DEL
	trigger	✓	✓		
	trigger/triggers	✓	✓	✓	✓
	trigger/triggers/<ID>	✓	✓		✓
	trigger/triggers/<ID>/notifications	✓	✓	✓	✓
	trigger/triggers/<ID>/notifications/<ID>	✓	✓		✓
	trigger/schedule	✓	✓		
	notification	✓	✓		
	notification/mailing	✓	✓	✓	✓
	notification/mailing/<ID>	✓	✓		✓
	notification/ftp	✓	✓	✓	✓
	notification/ftp/<ID>	✓	✓		✓
	notification/httpHost	✓	✓	✓	✓
	notification/httpHost/<ID>	✓	✓		✓
	notification/alertStream	✓			

5. Media Streaming

There are several methods to stream live video and audio from an IP media device to a client.

5.1. Streaming with RTP and RTSP

An IP media device must support streaming of video and audio content using RTSP [RFC2326], SDP [4566] and RTP [RFC3550, RFC3551].

RTP provides a framework for the transport of real-time media. RTP is flexible and has been used successfully to transmit telephone signals over IP, real-time media from voice and audio teleconferencing over low-bandwidth links to high-definition television over high-bandwidth connections. Its flexibility and widespread acceptance have made RTP a de facto standard since its inception.

RTP has been adopted as a standard by the Internet Engineering Task Force (IETF). RTP has also been adopted by the International Telecommunications Union (ITU) as part of its H.323 series of recommendations.

RTP can stream media over both unicast and multicast networks. As defined, RTP focuses on streaming and leaves streaming control and session establishment to other standard protocols that are addressed below. RTP is deliberately incomplete: additional profiles specify algorithms and frameworks for media playout and timing regeneration, synchronization between media streams, error concealment and correction or congestion control. RTP also does not specify mappings between payload and media types.

RTP is based on two important principles: application-level framing and the end-to-end principle. Application-level framing, as it applies to media transport, implies that the transmission protocol should make a minimum set of assumptions about the requirements of the data being streamed, leaving it up to the application (sender and receivers) to frame (packetizer) content and manage unreliable transmission. The end-to-end principle implies that intelligence lies with the sender and receiver. The network is considered a stateless, “dumb” packet-delivery system. Combined together, these two principles provide a unifying framework for real-time audio/video transport, satisfying most applications directly yet being malleable for those applications that stretch its limits.

RTSP is a control protocol designed for serving multimedia sessions. RTSP can act as a “network remote control” for media servers (including surveillance equipment). RTSP is similar in syntax and operation to HTTP.

RTSP defines a means to deliver RTP streaming using TCP. This can be useful for streaming data in situations where loss cannot be tolerated, such as when downloading a video clip or streaming important metadata.

SDP is a protocol that describes a media session. Because of the format of a session description, certain information is always needed. SDP is used to convey the transport addresses on which media flows, the format of the media, the corresponding RTP payload formats and profiles and the times when the session as well as the media durations.

There is some overlap between RTSP and the SDP: some of the information available in the session description is also available via RTSP commands.

5.1.1. Use of RTP and RTCP

It is highly recommended that IP media devices implement the sending of RTCP sender reports in order to facilitate time synchronization and for network diagnosis and reporting. The sender report must contain an absolute NTP timestamp relative to Jan 1, 1900 with its corresponding RTP timestamp.

It is highly recommended that IP media devices send a BYE RTCP packet when disconnecting from a session, even in unicast. This information permits a client to distinguish between voluntary and involuntary session termination.

5.1.2. Use of RTSP and SDP

Media Request URI

An IP media device makes streaming channels accessible in RTSP via an associated RTSP URI. This URI has the form:

```
rtsp://<address>:<port>/<path>?<paramName>=<paramValue>&…
```

An RTSP URI consists of a base path and a set of parameter name-value pairs.

For delivery of live streaming content, the RTSP URIs have the following form:

```
rtsp://<address>:<port>/PSIA/Streaming/channels/ID?<parameters>
```

Where the path corresponds to the REST resource for a given streaming channel as defined in section 7.12.3. The set of valid parameters corresponds to the query strings and their types in section 7.12.5. This correspondence is specified in order to allow introspection on the supported set of query parameters for a given streaming channel.

Example:

```
rtsp://192.168.99.11:554/PSIA/Streaming/channels/123456?videoCodecType=MJPEG&rotationDegr
ee=90
```

Minimal Implementation

The default port number for an IPMD RTSP server is 554.

All clients and server must implement all required features of the minimal RTSP implementation described in Appendix D of RFC 2326.

The DESCRIBE method must be supported and must support SDP as the description format according to Appendix C of RFC 2326.

The RTP/AVP profile defined in RFC 3551 must be supported. For SETUP requests, the “Transport”, “client_port”, “server_port”, “source”, “ssrc” and “RTP-Info” headers must be supported.

If multicast is supported, the “destination”, “port” and “ttl” headers **must** be supported.

Supported Transport Protocols

IP media device RTSP servers are required to support UDP as a transmission transport protocol.

It is highly recommended that IP media device RTSP servers support TCP as a transmission transport protocol. If TCP is supported, the RTSP server must support interleaving of RTP/RTCP packets over the RTSP TCP connection.

It is highly recommended that IP media device RTSP servers support UDP multicast transport.

Keep Alive

The RTSP server must indicate the session timeout: any clients who do not notify the server that they are still alive within this time limit (and periodically afterwards) should have their corresponding media streams terminated.

An IP media device RTSP server must support the RTSP OPTIONS method and RTCP receiver reports for keepalives. The media device should treat any valid RTSP command from the client as a "keep-alive" request and maintain an active session accordingly. Supporting GET_PARAMETER for keepalives is optional.

RTSP Authentication

IP media device RTSP servers must support HTTP basic authentication. It is highly recommended that RTSP servers support HTTP digest authentication (RFC 2069).

The set of valid user names and passwords required to access an RTSP session is configured in `/System/AAA`. Permission granularity is left to the device implementation.

5.1.3. RTP Packetization Rules for Codecs

Rules for the description and packetization of codecs required for interoperability over RTSP, SDP and RTP are defined in additional IETF RFC documents. The following table lists some of the commonly used codecs for video surveillance applications:

Codec	Normative Reference	Mime Type(s)	RFC
Video			
Motion JPEG	ISO/IEC IS 10918-1 ITU-T Rec. T.81	`video/jpeg`	RFC 2435
MPEG-2	ISO/IEC 13818-2	`video/MPV`	RFC 2250
MPEG-4 SP/ASP	ISO/IEC 14496-2:2004	`video/MP4V-ES`	RFC 3016 RFC 3640
H.264 AVC	ISO/IEC 14496-10:2005 ITU-T Rec. H.264:2005	`video/H264`	RFC 3984
H.264 SVC	ISO/IEC 14496-10 Amd 3 ITU-T Rec. H.264 Annex G	`video/H264-SVC`	IETF Draft

Codec	Normative Reference	Mime Type(s)	RFC
Audio			
G.726[1]	ITU-T Rec. G.726	audio/G726-40 audio/G726-32 audio/G726-24 audio/G726-16 audio/AAL2-G726-40 audio/AAL2-G726-32 audio/AAL2-G726-24 audio/AAL2-G726-16	RFC 3551
MPEG-1 Layer II & III	ISO/IEC 11172-3:1993	audio/mpeg	RFC 2250
AAC	ISO/IEC 14496-3	audio/aac-lbr audio/aac-hbr	RFC 3640

For RTP packetization, any additional codecs must conform to payload format defined in an IETF specification or draft specification if present.

5.2. Streaming using HTTP Server Push

It is highly recommended that an IP media device support streaming of video and audio content over an HTTP server push connection. See `/PSIA/Streaming/channels/`*`ID`*`/http` in section 7.12.5 for a description of HTTP streaming.

[1] The ordering of G.726 code words in RTP packets is currently ambiguous. According to ITU-T Recommendation I.366.2 Annex E for ATM AAL2 transport, G.726 code words should be packed in "big-endian" order (network byte order) into the payload bytes of an RTP packet. This conflicts, however, with the latest IETF revised draft specification for RTP audio and video profiles that specifies a "little-endian" packing order and delegates different MIME types for the big-endian packing ("AAL2-G726-32"). It should be noted that there is no straightforward means to detect the packing order from the data itself. As some equipment manufacturers have already implemented RTP transport with the older packing order, assuring interoperability between devices is problematic. It appears, however, that the "big-endian" packing order for G.726 is widely accepted in the industry. Implementations must interpret and produce the MIME type that is appropriate for the G.726 codeword ordering according to RFC 3551. In order to integrate equipment that does not correctly implement the standard, it must be possible to identify the source with the RTSP "Server" header field or the SDP "a=tool" field in order to modify the code word order for further transmission or decoding.

6. Common Data Types

The XML Data Blocks described in this document contains annotations that describe the properties of the field. For a complete definition, see the XML schema definitions.

The following information is inserted into the comments to describe the data carried in the field:

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

Note that XML structures that are optional may have required fields. This means that the entire XML block is optional, however if it is present the required fields are mandatory.

6.1. Built-in Types

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]
ID	ID from service model.
IPv4 Address	Notation is xxx.xxx.xxx.xxx Example: 3.137.217.220
IPv6 Address	Notation is xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx using CIDR notation. Example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
MAC	MAC Address Notation is aa:bb:cc:dd:ee:ff with 6 hex bytes.
TTL	A positive numerical value indicating the number of hops (routers) that traffic is permitted to pass through before expiring. Value is >=0.

6.2. ReceiverAddress

```
<ReceiverAddress>
   <addressingFormatType>
       <!-- xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>     <!-- dep, xs:string -->       </hostName>
   <ipAddress>    <!-- dep, xs:string -->       </ipAddress>
   <ipv6Address>  <!-- dep, xs:string -->       </ipv6Address>
   <portNo>       <!-- opt, xs:integer -->      </portNo>
</ReceiverAddress>
```

Notes:

- Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server.
- Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in `/System/Network/interfaces/`*`ID`*`/ipAddress`.

6.3. TimeBlockList

<TimeBlockList> holds a set of <TimeBlock> XML that define a set of time ranges.

```
<TimeBlockList version="1.0" xmlns="urn:psialliance-org">
   <TimeBlock>
       <dayOfWeek>
           <!-- req, xs:integer, ISO8601 weekday number, 1=Monday, … -->
       </dayOfWeek>
       <TimeRange>        <!-- dep -->
           <beginTime>    <!-- req, xs:time, ISO8601 time -->  </beginTime>
           <endTime>      <!-- req, xs:time, ISO8601 time -->  </endTime>
       </TimeRange>
       <bitString>    <!-- dep, xs:string, Hour 0..24, 1/0 per hour -->   </bitString>
   </TimeBlock>
</TimeBlockList>
```

Notes:

- If <dayOfWeek> is not present the time block is valid every day. No two <TimeBlock> in the same list should provide the same <dayOfWeek>.
- If the <bitString> tag is provided, <TimeRange> should not be provided, and vice versa.
- The <bitString> field can be used to reduce the amount of required, transferable XML. The field is a string of 24 bits, where each bit specifies an hour of the day. The left-most bit is hour 0, and the right-most bit is hour 24. A '1' indicates that the specified hour is enabled for event detection and triggering, and a '0' indicates that it is not. Thus, all <bitString> fields must be 24 bits in length.

7. Service Command Details

7.1. /PSIA/System

URI	/PSIA/System		Type	Service
Function	System services.			
Methods	Query String(s)	Inbound Data	Return Result	
Notes				

7.1.1. /PSIA/System/reboot

URI	/PSIA/System/reboot		Type	Resource
Function	Reboot the device.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	The <ResponseStatus> XML data is returned before the device proceeds to reboot.			

7.1.2. /PSIA/System/updateFirmware

URI	/PSIA/System/updateFirmware		Type	Resource
Function	Update the firmware of the device.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	After successful completion of this API, the <ResponseStatus> XML data is returned, and the device proceeds to reboot.			

7.1.3. /PSIA/System/configurationData

URI	/PSIA/System/configurationData		Type	Resource
Function	The function is used to get or set the configuration data for the device. This is opaque data that can be used to save and restore the device configuration.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			Opaque data	
PUT		Opaque Data	<ResponseStatus>	
Notes	Configuration data is device-dependent – it may be binary or any other format. Client may use the HTTP `Accept:` header field to inform server what formats are expected. May reboot device after configuration data is applied.			

7.1.4. /PSIA/System/factoryReset

URI	/PSIA/System/factoryReset		Type	Resource
Function	This function is used to reset the configuration for the device to the factory default.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT	Mode		<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 /PSIA/System/Network and /PSIA/Security. The default mode is "full". The device may be rebooted after it is reset.			

7.1.5. /PSIA/System/deviceInfo

URI	/PSIA/System/deviceInfo		Type	Resource
Function	This function is used to get or set device information.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DeviceInfo>	
PUT		<DeviceInfo>	<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. <systemObjectID> is the System Object Identifier defined in RFC1213.			

DeviceInfo XML Block

```
<DeviceInfo version="1.0" xmlns="urn:psialliance-org">
   <deviceName>          <!-- req, xs:string -->          </deviceName>
   <deviceID>            <!-- ro, req, xs:string;uuid --> </deviceID>
   <deviceDescription>   <!-- opt, xs:string -->          </deviceDescription>
   <deviceLocation>      <!-- opt, xs:string -->          </deviceLocation>
   <systemContact>       <!-- opt, xs:string -->          </systemContact>
       <!-- Note: The following are read-only parameters -->
   <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>
   <logicVersion>        <!-- ro, opt, xs:string -->      </logicVersion>
   <logicReleasedDate>   <!-- ro, opt, xs:string -->      </logicReleasedDate>
   <bootVersion>         <!-- ro, opt, xs:string -->      </bootVersion>
   <bootReleasedDate>    <!-- ro, opt, xs:string -->      </bootReleasedDate>
   <rescueVersion>       <!-- ro, opt, xs:string -->      </rescueVersion>
   <rescueReleasedDate>  <!-- ro, opt, xs:string -->      </rescueReleasedDate>
```

```
   <hardwareVersion>     <!-- ro, opt, xs:string -->      </hardwareVersion>
   <systemObjectID>      <!-- ro, opt, xs:string -->      </systemObjectID>
</DeviceInfo>
```

7.1.6. /PSIA/System/supportReport

URI	/PSIA/System/supportReport		Type	Resource
Function	This function is used to get a compressed archive of support information for the device. The archive must contain at least the device's current configuration and log files. Other items that might also be packaged include syslog and operating system information, statistics, etc.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Support Data	
Notes	The format of the archive is device-dependent (could be tar, zip, etc.). Use http `Accept:` header field to inform server what formats are accepted by client.			

7.1.7. /PSIA/System/status

URI	/PSIA/System/status		Type	Resource
Function	This function is used to get the status of the device.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<DeviceStatus>	
Notes	Not all fields of <DeviceStatus> may be present.			

DeviceStatus XML Block

```
<DeviceStatus version="1.0" xmlns="urn:psialliance-org">
   <currentDeviceTime>   <!-- opt, xs:datetime -->        </currentDeviceTime>
   <deviceUpTime>        <!-- opt, xs:integer, seconds --> </deviceUpTime>
   <TemperatureList>     <!-- opt -->
      <Temperature>
         <tempSensorDescription>   <!-- req, xs:string -->   </tempSensorDescription>
         <temperature>             <!-- req, xs:float -->    </temperature>
      </Temperature>
   </TemperatureList>
   <FanList>             <!-- opt -->
      <Fan>
         <fanDescription>          <!-- req, xs:string -->      </fanDescription>
         <speed>                   <!-- req, xs:integer -->     </speed>
      </Fan>
   </FanList>
   <PressureList>        <!-- opt -->
      <Pressure>
         <pressureSensorDescription>   <!-- req, xs:string --></pressureSensorDescription>
         <pressure>                    <!-- req, xs:integer -->  </pressure>
      </Pressure>
   </PressureList>
   <TamperList>          <!-- opt -->
      <Tamper>
         <tamperSensorDescription>     <!-- req, xs:string -->   </tamperSensorDescription>
         <tamper>                      <!-- req, xs:boolean -->  </tamper>
      </Tamper>
```

```
   </TamperList>
   <CPUList>             <!-- opt -->
      <CPU>
         <cpuDescription>   <!-- req, xs:string -->                     </cpuDescription>
         <cpuUtilization>   <!-- req, xs:integer, percentage 0..100 -->  </cpuUtilization>
      </CPU>
   </CPUList>
   <MemoryList>          <!-- opt -->
      <Memory>
         <memoryDescription>   <!-- req, xs:string -->       </memoryDescription>
         <memoryUsage>         <!-- req, xs:float, in MB --> </memoryUsage>
         <memoryAvailable>     <!-- req, xs:float, in MB-->  </memoryAvailable>
      </Memory>
   </MemoryList>
   <openFileHandles>     <!-- opt, xs:integer -->  </openFileHandles>
</DeviceStatus>
```

7.1.8. /PSIA/System/time

URI	/PSIA/System/time		Type	Resource
Function	Access the device time information.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<Time>	
PUT	timeMode localTime timeZone	<Time>	<ResponseStatus>	
Notes	If the “localTime” query string with a value is specified, the <Time> XML block is not required as inbound data. If <timeMode> is set to “manual” the <localTime> and <timeZone> fields are required. The <LocalTime> block sets the device time. If <timeMode> is 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="urn:psialliance-org">
   <timeMode>     <!-- req, xs:string, "NTP,manual" -->    </timeMode>
   <localTime>    <!-- req, xs:datetime -->                </localTime>
   <timeZone>     <!-- req, xs:string, POSIX time zone string; see below -->     </timeZone>
</Time>
```

7.1.9. /PSIA/System/time/localTime

URI	/PSIA/System/time/localTime		Type	Resource
Function	Access the device local time information.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			ISO 8601 Date-Time String	
PUT		ISO 8601 Date-Time String	<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.			

7.1.10. /PSIA/System/time/timeZone

URI	/PSIA/System/time/timeZone		Type	Resource
Function	Access the device time zone.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			Time zone string	
PUT		Time zone string	<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.			

7.1.11. /PSIA/System/time/ntpServers

URI	/PSIA/System/time/ntpServers		Type	Resource
Function	Access the NTP servers configured for the device.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<NTPServerList>	
PUT		<NTPServerList>	<ResponseStatus>	
POST		<NTPServer>	<ResponseStatus>	

DELETE			<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="urn:psialliance-org">
   <NTPServer/>   <!-- opt -->
</NTPServerList>
```

7.1.12. /PSIA/System/time/ntpServers/<ID>

URI	/PSIA/System/time/ntpServers/*ID*	**Type**	Resource
Function	Access an NTP server configured for the device.		

Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<NTPServer>
PUT		<NTPServer>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.		

NTPServer XML Block

```
<NTPServer version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string; id -->   </id>
   <addressingFormatType>
       <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>        <!-- dep, xs:string -->       </hostName>
   <ipAddress>       <!-- dep, xs:string -->       </ipAddress>
   <ipv6Address>     <!-- dep, xs:string -->       </ipv6Address>
   <portNo>          <!-- opt, xs:integer -->      </portNo>
</NTPServer>
```

7.1.13. /PSIA/System/logging

URI	/PSIA/System/logging	**Type**	Resource
Function	This function is used to access the logging parameters.		

Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<Logging>
PUT		<Logging>	<ResponseStatus>
Notes	The device maintains a rolling log of <maxEntries> that can be configured and queried.		

Logging XML Block

```
<Logging version="1.0" xmlns="urn:psialliance-org">
```

```
   <LogTrigger>       <!-- opt -->
      <severity>      <!-- req, xs:string, Severities are defined in RFC3164 --> </severity>
   </LogTrigger>
   <LocalLog>         <!-- opt -->
      <maxEntries>    <!-- req, xs:integer -->  </maxEntries>
   </LocalLog>
</Logging>
```

7.1.14. /PSIA/System/logging/messages

URI	/PSIA/System/logging/messages		Type	Resource
Function	This function is used to access the message log.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<LogMessageList>	
Notes	Devices may define additional logging fields for extended information. The message log is read-only.			

LogMessageList XML Block

```
<LogMessageList version="1.0" xmlns="urn:psialliance-org">
   <LogMessage>       <!-- opt -->
      <logNo>         <!-- req, xs:integer -->                        </logNo>
      <dateTime>      <!-- req, xs:datetime -->                       </dateTime>
      <severity>      <!-- req, xs:integer, defined in RFC3164 -->    </severity>
      <eventID>       <!-- opt, xs:string;id -->                      </eventID>
      <message>       <!-- req, xs:string -->                        </message>
   </LogMessage>
</LogMessageList>
```

7.2. /PSIA/System/Storage

URI	/PSIA/System/Storage		Type	Service
Function	This function is used to access storage parameters.			
Methods	Query String(s)	Inbound Data	Return Result	
Notes	Storage service.			

7.3. /PSIA/System/Storage/volumes

URI	/PSIA/System/Storage/volumes		Type	Resource
Function	This function is used to access the storage volumes and files on a device.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<StorageVolumeList>	
Notes	Storage is organized into volumes. Each volume is an individual storage space. Creation and configuration of volumes is outside the scope of this interface, thus the information is available on a read-only basis.			

StorageVolumeList XML Block

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

7.3.1. /PSIA/System/Storage/volumes/<ID>

URI	/PSIA/System/Storage/volumes/*ID*		Type	Resource
Function	This function is used to access a particular storage volume by its ID.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<StorageVolume>	
Notes	Volume information can only be read using this interface.			

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>
</StorageVolume>
```

7.3.2. /PSIA/System/Storage/volumes/<ID>/status

URI	/PSIA/System/Storage/volumes/*ID*/status		Type	Resource
Function	This function is used to query the status of a particular storage.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<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>
```

7.3.3. /PSIA/System/Storage/volumes/<ID>/format

URI	/PSIA/System/Storage/volumes/*ID*/format		Type	Resource
Function	Format a storage volume.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	Formatting may take time.			

7.3.4. /PSIA/System/Storage/volumes/<ID>/files

URI	/PSIA/System/Storage/volumes/*ID*/files		Type	Resource
Function	Get the list of files stored on a particular storage volume.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<StorageFileList>	
DELETE			<ResponseStatus>	
Notes	Storage files are read-only, except for the possibility to delete. DELETE removes all of the files on the storage volume.			

StorageFileList XML Block

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

7.3.5. /PSIA/System/Storage/volumes/<ID>/files/<ID>

URI	/PSIA/System/Storage/volumes/*ID*/files/*ID*		Type	Resource
Function	Access and manipulate a file.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<StorageFile>	
DELETE			<ResponseStatus>	
Notes	DELETE removes a particular file from the storage volume.			

StorageFile XML Block

```
<StorageFile version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- ro, req, xs:string;id -->       </id>
   <fileName>        <!-- ro, req, xs:string -->          </fileName>
   <fileTimeStamp>   <!-- ro, req, xs:datetime -->        </fileTimeStamp>
   <fileSize>        <!-- ro, req, xs:float, in MB -->    </fileSize>
</StorageFile>
```

7.3.6. /PSIA/System/Storage/volumes/<ID>/files/<ID>/data

URI	/PSIA/System/Storage/volumes/*ID*/data		Type	Resource
Function	This function is used to access the data of a particular file.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Raw File Data	
Notes	The video/audio data may be encrypted according to device-dependent specifications. The video/audio format is dependent on device capabilities and configurations. The client may use the HTTP `Accept:` header to negotiate the data format.			

7.4. /PSIA/System/Network

URI	/PSIA/System/Network		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	System network configuration.			

7.4.1. /PSIA/System/Network/interfaces

URI	/PSIA/System/Network/interfaces		Type	Resource
Function	Access the device network interfaces.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<NetworkInterfaceList>	
Notes	As hardwired system resources, network interfaces cannot be created or destroyed.			

NetworkInterfaceList XML Block

```
<NetworkInterfaceList version="1.0" xmlns="urn:psialliance-org">
   <NetworkInterface/>   <!-- req -->
</NetworkInterfaceList>
```

7.4.2. /PSIA/System/Network/interfaces/<ID>

URI	/PSIA/System/Network/interfaces/*ID*		Type	Resource
Function	Access a particular network interface.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<NetworkInterface>	
PUT		<NetworkInterface>	<ResponseStatus>	
Notes				

NetworkInterface XML Block

```
<NetworkInterface version="1.0" xmlns="urn:psialliance-org">
   <id>          <!-- ro, req, xs:string;id -->       </id>
   <IPAddress/>  <!-- req -->
   <Wireless/>   <!-- opt -->
   <IEEE802_1x/> <!-- opt -->
   <IPFilter/>   <!-- opt -->
   <SNMP/>       <!-- opt -->
   <QoS/>        <!-- opt -->
   <Discovery/>  <!-- opt -->
   <Syslog/>     <!-- opt -->
</NetworkInterface>
```

7.4.3. /PSIA/System/Network/interfaces/<ID>/ipAddress

URI	/PSIA/System/Network/interfaces/*ID*/ipAddress		Type	Resource
Function	Access IP addressing settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IPAddress>	
PUT		<IPAddress>	<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. If <addressingType> refers to APIPA, the device IP address is automatically configured without DHCP. In this case the gateway and DNS fields are optional. Use of <ipAddress> or <ipv6Address> in fields is dictated by the <ipVersion> field. If <ipVersion> is “v4” the <ipAddress> fields are used; if <ipVersion> is “v6” the <ipv6Address> fields are used. If <ipVersion> is "dual", both <ipAddress> and <ipv6Address> fields may be used. <subnetMask> notation is “xxx.xxx.xxx.xxx”. <IPV6Address> is “xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx” using CIDR notation.			

IPAddress XML Block

```
<IPAddress version="1.0" xmlns="urn:psialliance-org">
   <ipVersion>       <!-- req, xs:string, "v4,v6,dual" --></ipVersion>
   <addressingType>  <!-- req, xs:string, "static,dynamic,apipa" --> </addressingType>
   <ipAddress>       <!-- dep, xs:string -->                         </ipAddress>
   <subnetMask>      <!-- dep, xs:string, subnet mask for IPv4 address -->   </subnetMask>
   <ipv6Address>     <!-- dep, xs:string -->                         </ipv6Address>
   <bitMask>         <!-- dep, xs:integer, bitmask IPv6 address -->  </bitMask>
   <DefaultGateway>  <!-- dep -->
       <ipAddress>   <!-- dep, xs:string -->      </ipAddress>
       <ipv6Address> <!-- dep, xs:string -->      </ipv6Address>
   </DefaultGateway>
   <PrimaryDNS>      <!-- dep -->
       <ipAddress>   <!-- dep, xs:string -->      </ipAddress>
       <ipv6Address> <!-- dep, xs:string -->      </ipv6Address>
   </PrimaryDNS>
   <SecondaryDNS>    <!-- dep -->
       <ipAddress>   <!-- dep, xs:string -->      </ipAddress>
       <ipv6Address> <!-- dep, xs:string -->      </ipv6Address>
   </SecondaryDNS>
</IPAddress>
```

7.4.4. /PSIA/System/Network/interfaces/<ID>/wireless

URI	/PSIA/System/Network/interfaces/*ID*/wireless		Type	Resource
Function	Access wireless network settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<Wireless>	
PUT		<Wireless>	<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. If the "WPA" or "WPA2-enterprise" security mode is used, the <WPA> block must be used and settings related to 802.1x must be set using the /PSIA/System/Network/interfaces/*ID*/ieee802.1x resource. <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="urn:psialliance-org">
   <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" --> </wepKeyLength>
           <EncryptionKeyList>
               <encryptionKey>
                   <!-- req, xs:hexBinary, WEP encryption key in hexadecimal format -->
               </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>
       </WPA>
   </WirelessSecurity>
</Wireless>
```

7.4.5. /PSIA/System/Network/interfaces/<ID>/ieee802.1x

URI	/PSIA/System/Network/interfaces/*ID*/ieee802.1x		Type	Resource
Function	Access IEEE 802.1x settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IEEE802_1x>	
PUT		<IEEE802_1x>	<ResponseStatus>	
Notes	If the <authenticatonProtocolType> tag corresponds to "EAP-TTLS", then the <innerTTLSAuthenticationMethod> tag must be provided. If the <authenticationProtocolType> corresponds to "EAP-PEAP" or “EAP-FAST”, then the <innerEAPProtocolType> tag must be provided. The <anonymousID> tag is optional. If the <authenticationProtocolType> corresponds to “EAP-FAST”, then the <autoPACProvisioningEnabled> tag must be provided. <anonymousID> is the optional anonymous ID to be used in place of the <userName>.			

IEEE802_1x XML Block

```
<IEEE802_1x version="1.0" xmlns="urn:psialliance-org">
   <enabled>  <!-- req, xs:boolean -->  </enabled>
   <authenticationProtocolType>
       <!-- req, xs:string, "EAP-TLS,EAP-TTLS,EAP-PEAP,EAP-LEAP,EAP-FAST" -->
   </authenticationProtocolType>
   <innerTTLSAuthenticationMethod>
       <!-- dep, xs:string, "MS-CHAP,MS-CHAPv2,PAP,EAP-MD5" -->
   </innerTTLSAuthenticationMethod>
   <innerEAPProtocolType>
       <!-- dep, xs:string, "EAP-POTP,MS-CHAPv2" -->
   </innerEAPProtocolType>
   <validateServerEnabled>   <!-- dep, xs:boolean -->  </validateServerEnabled>
   <userName>         <!-- dep, xs:string -->   </userName>
   <password>         <!-- dep, xs:string -->   </password>
   <anonymousID>      <!-- opt, xs:string -->   </anonymousID>
   <autoPACProvisioningEnabled>  <!-- dep, xs:boolean -->  </autoPACProvisioningEnabled>
</IEEE802_1x>
```

7.4.6. /PSIA/System/Network/interfaces/<ID>/ipFilter

URI	/PSIA/System/Network/interfaces/*ID*/ipFilter		Type	Resource
Function	Access IP filtering settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IPFilter>	
PUT		<IPFilter>	<ResponseStatus>	
Notes	The <permissionType> field, if provided as a direct child of <IPFilter>, acts as a system level configuration and will apply to all of the <IPFilterAddress> entries, overriding the value provided in a particular <IPFilterAddress> block.			

IPFilter XML Block

```
<IPFilter version="1.0" xmlns="urn:psialliance-org">
   <enabled>          <!-- req, xs:boolean -->                 </enabled>
   <permissionType>   <!-- opt, xs:string, "deny,allow" -->    </permissionType>
```

```
   <IPFilterAddressList/> <!-- opt -->
</IPFilter>
```

7.4.7. /PSIA/System/Network/interfaces/<ID>/ipFilter/filterAddresses

URI	/PSIA/System/Network/interfaces/*ID*/ipFilter/filterAddresses		Type	Resource
Function	Access IP filtering list			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IPFilterAddressList>	
PUT		<IPFilterAddressList>	<ResponseStatus>	
POST		<IPFilterAddress>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	The IP filter address list allows addresses to be added and removed from the list, or the entire list to be uploaded at once.			

IPFilterAddressList XML Block

```
<IPFilterAddressList version="1.0" xmlns="urn:psialliance-org">
   <IPFilterAddress/>    <!-- opt -->
</IPFilterAddressList>
```

7.4.8. /PSIA/System/Network/interfaces/<ID>/ipFilter/filterAddresses/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/ipFilter/filterAddresses/*ID*		Type	Resource
Function	Access a particular IP filtering entry.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IPFilter>	
PUT		<IPFilter>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	If the <permissionType> tag is not provided as a direct child of <IPFilter>, the <permissionType> tag must be provided for each <IPFilterAddress>. Since the ordering of the filters can change the behavior, filtering will be applied consecutively starting with the first <IPFilterAddress> in the list. The <bitMask> field is applied to the corresponding IP address to identify a range of addresses. It indicates the number of '1' bits used to mask the address. For example: '24' would correspond to a subnet mask of 255.255.255.0 and '32' would correspond to a subnet mask of 255.255.255.255 (a single IP address) for IPv4. If <addressFilterType> refers to "mask", the <AddressMask> block must be provided in place of the <AddressRange> block. If it refers to "range", the <Range> block must be provided in place of the <AddressMask> block. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.			

IPFilterAddress XML Block

```
<IPFilterAddress version="1.0" xmlns="urn:psialliance-org">
   <id>                    <!-- req, xs:string;id -->              </id>
   <permissionType>        <!-- dep, xs:string, "deny,allow" -->   </permissionType>
```

```
   <addressFilterType>       <!-- req, xs:string, "mask,range" -->    </addressFilterType>
   <AddressRange>            <!-- dep, depends on <addressFilterType> -->
      <startIPAddress>       <!-- dep, xs:string -->                  </startIPAddress>
      <endIPAddress>         <!-- dep, xs:string -->                  </endIPAddress>
      <startIPv6Address>     <!-- dep, xs:string -->                  </startIPv6Address>
      <endIPv6Address>       <!-- dep, xs:string -->                  </endIPv6Address>
   </AddressRange>
   <AddressMask>             <!-- dep, depends on <addressFilterType> -->
      <ipAddress>            <!-- dep, xs:string -->                  </ipAddress>
      <ipv6Address>          <!-- dep, xs:string -->                  </ipv6Address>
      <bitMask>              <!-- req, xs:string -->                  </bitMask>
   </AddressMask>
</IPFilterAddress>
```

7.4.9. /PSIA/System/Network/interfaces/<ID>/snmp

URI	/PSIA/System/Network/interfaces/*ID*/snmp		Type	Resource
Function	SNMP settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<SNMP>	
PUT		<SNMP>	<ResponseStatus>	
Notes	At least one of the <SNMPv2c> block or <SNMPAdvanced> block must be provided.			

SNMP XML Block

```
<SNMP version="1.0" xmlns="urn:psialliance-org">
   <SNMPv2c/>         <!-- dep, either <SNMPv2c> or <SNMPAdvanced> is required -->
   <SNMPAdvanced/>    <!-- dep -->
</SNMP>
```

7.4.10. /PSIA/System/Network/interfaces/<ID>/snmp/v2c

URI	/PSIA/System/Network/interfaces/*ID*/snmp/v2c		Type	Resource
Function	SNMP V2C parameters.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<SNMPv2c>	
PUT		<SNMPv2c>	<ResponseStatus>	
Notes	SNMP v2c configuration includes SNMP notification parameters and a set of SNMP trap receivers. SNMP v2c comprises SNMP v2 without the controversial new SNMP v2 security model, using instead the simple community-based security scheme of SNMP v1			

SNMPv2c XML Block

```
<SNMPv2c version="1.0" xmlns="urn:psialliance-org">
   <notificationEnabled>     <!-- req, xs:boolean -->  </notificationEnabled>
   <SNMPTrapReceiverList/>   <!-- opt -->
</SNMPv2c>
```

7.4.11. /PSIA/System/Network/interfaces/<ID>/snmp/v2c/trapReceivers

URI	/PSIA/System/Network/interfaces/*ID*/snmp/v2c/trapReceivers		Type	Resource
Function	SNMP trap receivers list.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPTrapReceiverList>	
PUT		<SNMPTrapReceiverList>	<ResponseStatus>	
POST		<SNMPTrapReceiver>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	It is possible to PUT the entire list at once.			

SNMPTrapReceiverList XML Block

```
<SNMPTrapReceiverList version="1.0" xmlns="urn:psialliance-org">
   <SNMPTrapReceiver/>   <!-- opt -->
</SNMPTrapReceiverList>
```

7.4.12. /PSIA/System/Network/interfaces/<ID>/snmp/v2c/trapReceivers/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/snmp/v2c/trapReceivers/*ID*		Type	Resource
Function	SNMP trap receiver information.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPTrapReceiver>	
PUT		<SNMPTrapReceiver>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<communityString> format must conform to the SNMPv2c standard.			

SNMPTrapReceiver XML Block

```
<SNMPTrapReceiver version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->              </id>
   <ReceiverAddress/>    <!-- req -->
   <notificationType>    <!-- req, xs:string, "trap,inform" -->   </notificationType>
   <communityString>     <!-- opt, xs:string -->                 </communityString>
</SNMPTrapReceiver>
```

7.4.13. /PSIA/System/Network/interfaces/<ID>/snmp/advanced

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced		Type	Resource
Function	Advanced SNMP settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPAdvanced>	
PUT		<SNMPAdvanced>	<ResponseStatus>	
Notes	<localEngineID> is a hexadecimal string indicating the local device engine ID.			

	<authenticationNotificationEnabled> indicates if SNMP authentication failure notification is enabled on the device. <SNMPNotificationFilterList> is a list to filter traps based on OIDs.

SNMPAdvanced XML Block

```
<SNMPAdvanced version="1.0" xmlns="urn:psialliance-org">
   <localEngineID>    <!-- req, xs:hexBinary, see RFC2571 -->         </localEngineID>
   <authenticationNotificationEnabled>
       <!-- opt, xs:boolean -->
   </authenticationNotificationEnabled>
   <SNMPUserList/>                      <!-- opt -->
   <SNMPNotificationFilterList/>        <!-- opt -->
   <notificationEnabled>                <!-- opt, xs:boolean -->      </notificationEnabled>
   <SNMPNotificationReceiverList/>      <!-- opt -->
</SNMPAdvanced>
```

7.4.14. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/users

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/users		**Type**	Resource
Function	SNMP users.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPUserList>	
PUT		<SNMPUserList>	<ResponseStatus>	
POST		<SNMPUser>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Defines the set of SNMP users and their permissions.			

SNMPUserList XML Block

```
<SNMPUserList version="1.0" xmlns="urn:psialliance-org">
   <SNMPUser/>    <!-- opt -->
</SNMPUserList>
```

7.4.15. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/users/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/users/*ID*		Type	Resource
Function	SNMP user settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPUser>	
PUT		<SNMPUser>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<remoteEngineID> indicates the remote SNMP entity to which the user is connected. <snmpAuthenticationMethod> indicates the authentication method used. <snmpAuthenticationKey> defines the authentication key if encryption is used for <snmpAuthenticationMethod>. <snmpAuthenticationPassword> optional password used to calculate the <snmpAuthenticationKey> value if encryption is used for <snmpAuthenticationMethod>. <snmpPrivacyMethod> indicates if messages are protected from disclosure, and if so, the type of privacy protocol used. <snmpPrivacyKey> defines the privacy key if encryption is used for <snmpPrivacyMethod>. <snmpPrivacyPassword> optional password used to calculate the <snmpPrivacyKey> value if encryptions is used for <snmpPrivacyMethod>.			

SNMPUser XML Block

```
<SNMPUser version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->    </id>
   <userName>        <!-- req, xs:string -->       </userName>
   <remoteEngineID>  <!-- req, xs:hexBinary -->    </remoteEngineID>
   <snmpAuthenticationMethod>
       <!-- req, xs:string, "MD5,SHA,none" -->
   </snmpAuthenticationMethod>
   <snmpAuthenticationKey>   <!-- dep, xs:string -->              </snmpAuthenticationKey>
   <snmpAuthenticationPassword>
       <!-- dep, xs:string, see RFC3414 -->
   </snmpAuthenticationPassword>
   <snmpPrivacyMethod>       <!-- req, xs:string, "DES,none" -->  </snmpPrivacyMethod>
   <snmpPrivacyKey>          <!-- dep, xs:string -->              </snmpPrivacyKey>
   <snmpPrivacyPassword>     <!-- dep, xs:string, see RFC3414 --> </snmpPrivacyPassword>
</SNMPUser>
```

7.4.16. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/ notificationFilters

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/ notificationFilters		Type	Resource
Function	SNMP notification filters.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPNotificationFilterList>	
PUT		<SNMPNotificationFilterList>	<ResponseStatus>	
POST		<SNMPNotificationFilter>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Manages a list of notification filters for SNMP v2 or v3.			

SNMPNotificationFilterList XML Block

```
<SNMPNotificationFilterList version="1.0" xmlns="urn:psialliance-org">
   <SNMPNotificationFilter/>     <!-- req -->
</SNMPNotificationFilterList>
```

7.4.17. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/ notificationFilters/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/ notificationFilters/*ID*		Type	Resource
Function	SNMP notification filter settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPNotificationFilter>	
PUT		<SNMPNotificationFilter>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<oidSubtree> specifies the OID for which notifications are sent or blocked. <filterAction> indicates whether notifications regarding the OID are sent to the trap recipients.			

SNMPNotificationFilter XML Block

```
<SNMPNotificationFilter version="1.0" xmlns="urn:psialliance-org">
   <id>           <!-- req, xs:string;id -->    </id>
   <filterName>   <!-- req, xs:string -->       </filterName>
   <OIDSubtreeList>          <!-- opt -->
      <OID>
         <oidSubtree>       <!-- req, xs:string -->                          </oidSubtree>
         <filterAction>     <!-- req, xs:string, "included,excluded" -->     </filterAction>
      </OID>
   </OIDSubtreeList>
</SNMPNotificationFilter>
```

7.4.18. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/ notificationReceivers

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/ notificationReceivers		Type	Resource
Function	SNMP notification receivers.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPNotificationReceiverList>	
PUT		<SNMPNotificationReceiverList>	<ResponseStatus>	
POST		<SNMPNotificationReceiver>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Manage the list of SNMP notification receivers for v2 or v3.			

SNMPNotificationReceiverList XML Block

```
<SNMPNotificationReceiverList version="1.0" xmlns="urn:psialliance-org">
   <SNMPNotificationReceiver>    <!-- opt -->
</SNMPNotificationReceiverList>
```

7.4.19. /PSIA/System/Network/interfaces/<ID>/snmp/advanced/ notificationReceivers/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/snmp/advanced/ notificationReceivers/*ID*		Type	Resource
Function	SNMP notification receiver settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPNotificationReceiver>	
PUT		<SNMPNotificationReceiver>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<notificationType> indicates whether this receiver entry is for a trap or an inform. <userID> must correspond to a user ID in /PSIA/System/Network/interfaces/*ID*/snmp/advanced/users/*ID*. <securityType> defines the security level attached to the user. The "authentication" option will authenticate SNMP messages and ensure the origin is authenticated. The "privacy" option authenticates and encrypts the SNMP messages. <filterName> associates a filter if <filterEnabled> is true. <timeout> indicates the amount of time (seconds) the device waits before re-sending informs. <retries> indicates the number of times the device re-sends an inform request.			

SNMPNotificationReceiver XML Block

```
<SNMPNotificationReceiver version="1.0" xmlns="urn:psialliance-org">
   <ReceiverAddress/>     <!-- req -->
   <notificationType>     <!-- req, xs:string, "trap,inform" -->       </notificationType>
   <userID>               <!-- req, xs:string -->                     </userID>
   <securityType>
       <!-- req, xs:string, "noauthentication,authentication,privacy" -->
```

```
   </securityType>
   <filterEnabled>    <!-- req, xs:boolean -->                </filterEnabled>
   <filterName>       <!-- req, xs:integer -->                </filterName>
   <timeout>          <!-- opt, xs:integer, seconds -->       </timeout>
   <retries>          <!-- opt, xs:integer -->                </retries>
</SNMPNotificationReceiver>
```

7.4.20. /PSIA/System/Network/interfaces/<ID>/snmp/v3

URI	/PSIA/System/Network/interfaces/*ID*/snmp/v3		Type	Resource
Function	SNMP v3 settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SNMPAdvanced>	
PUT		<SNMPAdvanced>	<ResponseStatus>	
Notes	This resource is an alias to /PSIA/System/Network/interfaces/*ID*/snmp/advanced. The <snmpAuthenticationPassword> and <snmpPrivacyPassword> tags are optionally used if the device implementation chooses to calculate the corresponding keys based on a password (as in RFC3414). In this case, the <snmpAuthenticationKey> and <snmpPrivacyKey> may or may not be provided. The <localEngineID> tag is used for “trap” messages and the <remoteEngineID> tag is used for “inform” messages.			

7.4.21. /PSIA/System/Network/interfaces/<ID>/qos

URI	/PSIA/System/Network/interfaces/*ID*/qos		Type	Resource
Function	This function is used to set the QoS setting for the device.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<QoS>	
PUT		<QoS>	<ResponseStatus>	
Notes	At least one of <CoSList> or <DSCPList> must be provided.			

QoS XML Block

```
<QoS version="1.0" xmlns="urn:psialliance-org">
   <CoSList/>     <!-- dep -->
   <DSCPList/>    <!-- dep -->
</QoS>
```

7.4.22. /PSIA/System/Network/interfaces/<ID>/qos/cos

URI	/PSIA/System/Network/interfaces/*ID*/qos/cos		Type	Resource
Function	Class of Service (CoS) settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<CoSList>	
PUT		<CoSList>	<ResponseStatus>	
POST		<CoS>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A list of class of service parameter blocks is specified for each type of traffic: device management, command and control, video and audio streaming. Devices may extend the set of traffic types.			

CoSList XML Block

```
<CoSList version="1.0" xmlns="urn:psialliance-org">
   <CoS/>     <!-- opt -->
</CoSList>
```

7.4.23. /PSIA/System/Network/interfaces/<ID>/qos/cos/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/qos/cos/*ID*		Type	Resource
Function	Class of service settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<CoS>	
PUT		<CoS>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<trafficType> determines which kind of traffic the settings apply to.			

CoS XML Block

```
<CoS version="1.0" xmlns="urn:psialliance-org">
   <id>           <!-- req, xs:string;id -->    </id>
   <enabled>      <!-- req, xs:boolean -->      </enabled>
   <priority>     <!-- req, xs:integer -->      </priority>
   <vlanID>       <!-- req, xs:string -->       </vlanID>
   <trafficType>
       <!-- req, xs:string, "devicemanagement,commandcontrol,video,audio" -->
   </trafficType>
</CoS>
```

7.4.24. /PSIA/System/Network/interfaces/<ID>/qos/dscp

URI	/PSIA/System/Network/interfaces/*ID*/qos/dscp		Type	Resource
Function	Differentiated Services (DiffServ) settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DSCPList>	
PUT		<DSCPList>	<ResponseStatus>	
POST		<DSCP>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A list of DSCP parameter blocks is specified for each type of traffic: device management, command and control, video and audio streaming. Devices may extend the set of traffic types.			

DSCPList XML Block

```
<DSCPList version="1.0" xmlns="urn:psialliance-org">
   <DSCP/>    <!-- opt -->
</DSCPList>
```

7.4.25. /PSIA/System/Network/interfaces/<ID>/qos/dscp/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/qos/dscp/*ID*		Type	Resource
Function	DSCP entry settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<DSCP>	
PUT		<DSCP>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<trafficType> determines which kind of traffic the settings apply to.			

DSCP XML Block

```
<DSCP version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->    </id>
   <enabled>         <!-- req, xs:boolean -->      </enabled>
   <priorityValue>   <!-- req, xs:integer, 6 bits - refer to RFC2474 --> </priorityValue>
   <trafficType>
       <!-- req, xs:string, "devicemanagement,commandcontrol,video,audio" -->
   </trafficType>
</DSCP>
```

7.4.26. /PSIA/System/Network/interfaces/<ID>/discovery

URI	/PSIA/System/Network/interfaces/*ID*/discovery		Type	Resource
Function	Device discovery settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<Discovery>	
PUT		<Discovery>	<ResponseStatus>	
Notes	Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress. <portNo> is the port number for the multicast discovery address. <ttl> is the time to live for multicast discovery packets.			

Discovery XML Block

```
<Discovery version="1.0" xmlns="urn:psialliance-org">
   <UPnP>            <!-- opt -->
      <enabled>      <!-- req, xs:boolean -->  </enabled>
   </UPnP>
   <Zeroconf>        <!-- opt -->
      <enabled>      <!-- req, xs:boolean -->  </enabled>
   </Zeroconf>
   <MulticastDiscovery>  <!-- opt -->
      <enabled>      <!-- req, xs:boolean -->  </enabled>
      <ipAddress>    <!-- dep, xs:string -->   </ipAddress>
      <ipv6Address>  <!-- dep, xs:string -->   </ipv6Address>
      <portNo>       <!-- req, xs:integer -->  </portNo>
      <ttl>          <!-- req, xs:integer -->  </ttl>
   </MulticastDiscovery>
</Discovery>
```

7.4.27. /PSIA/System/Network/interfaces/<ID>/syslog

URI	/PSIA/System/Network/interfaces/*ID*/syslog		Type	Resource
Function	Syslog settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<Syslog>	
PUT		<Syslog>	<ResponseStatus>	
Notes	Configure the system settings.			

Syslog XML Block

```
<Syslog version="1.0" xmlns="urn:psialliance-org">
   <enabled>            <!-- req, xs:boolean -->  </enabled>
   <SyslogServerList/>   <!-- opt -->
</Syslog>
```

7.4.28. /PSIA/System/Network/interfaces/<ID>/syslog/servers

URI	/PSIA/System/Network/interfaces/*ID*/syslog/servers		Type	Resource
Function	Syslog server list.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<SyslogServerList>	
PUT		<SyslogServerList>	<ResponseStatus>	
POST		<SyslogServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Manage a set of syslog servers that receive logging notifications.			

SyslogServerList XML Block

```
<SyslogServerList version="1.0" xmlns="urn:psialliance-org">
   <SyslogServer/>    <!-- opt -->
</SyslogServerList>
```

7.4.29. /PSIA/System/Network/interfaces/<ID>/syslog/servers/<ID>

URI	/PSIA/System/Network/interfaces/*ID*/syslog/servers/*ID*		Type	Resource
Function	Syslog server settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<SyslogSever>	
PUT		<SyslogServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress. <facilityType> indicates the facility to store syslog messages. See RFC3164. <severity> indicates the minimum log severity for which to send a syslog message. See RFC3164.			

SyslogServer XML Block

```
<SyslogServer version="1.0" xmlns="urn:psialliance-org">
   <id>           <!-- req, xs:string;id -->    </id>
   <addressingFormatType>
       <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>     <!-- dep, xs:string -->       </hostName>
   <ipAddress>    <!-- dep, xs:string -->       </ipAddress>
   <ipv6Address>  <!-- dep, xs:string -->       </ipv6Address>
   <portNo>       <!-- opt, xs:integer -->      </portNo>
   <facilityType> <!-- req, xs:string, see RFC3164 -->     </facilityType>
   <severity>     <!-- opt, xs:string, see RFC3164 -->     </severity>
</SyslogServer>
```

7.4.30. Examples

Example: Getting the Network Settings

```
GET /PSIA/System/Network 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="urn:psialliance-org">
   <NetworkInterface>
       <id>1</id>
       <IPAddress>
           <ipVersion>v4</ipVersion>
           <addressingType>static</addressingType>
           <ipAddress>3.137.217.220</ipAddress>
           <subnetMask>255.255.255.0</subnetMask>
           <DefaultGateway>
               <ipAddress>3.137.217.0</ipAddress>
           </DefaultGateway>
           <PrimaryDNS>
               <ipAddress>3.137.218.37</ipAddress>
           </PrimaryDNS>
           <SecondaryDNS>
               <ipAddress>3.137.217.15</ipAddress>
           </SecondaryDNS>
       </IPAddress>
   </NetworkInterface>
   <NetworkInterface>
       <id>2</id>
       <IPAddress>
           <ipVersion>v4</ipVersion>
           <addressingType>dynamic</addressingType>
       </IPAddress>
       <Wireless>
           <enabled>true</enabled>
           <wirelessNetworkMode>intrastructure</wirelessNetworkMode>
           <WirelessSecurity>
               <securityMode>WPA-personal</securityMode>
               <WPA>
                   <algorithmType>AES</algorithmType>
                   <sharedKey>ac34587bc8a8fff7a</sharedKey>
               </WPA>
           </WirelessSecurity>
       </Wireless>
   </NetworkInterface>
</NetworkInterfaceList>
```

Example: Setting the IP Address

```
PUT /PSIA/System/Network/interfaces/1/ipAddress 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"?>
```

```
<IPAddress version="1.0" xmlns="urn:psialliance-org">
   <ipVersion> v4</ipVersion>
   <addressingType>static </addressingType>
   <ipAddress>3.137.217.220</ipAddress>
   <subnetMask>255.255.255.0</subnetMask>
   <DefaultGateway>
       <ipAddress>3.137.217.0</ipAddress>
   </DefaultGateway>
   <PrimaryDNS>
       <ipAddress>3.137.218.37</ipAddress>
   </PrimaryDNS>
   <SecondaryDNS>
       <ipAddress>3.137.217.15</ipAddress>
   </SecondaryDNS>
</IPAddress>
```

7.5. /PSIA/System/IO

URI	/PSIA/System/IO		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IOPortList>	
Notes	The allocation of IDs between input and output ports must be unique.			

IOPortList XML Block

```
<IOPortList version="1.0" xmlns="urn:psialliance-org">
   <IOInputPortList/>    <!-- opt -->
   <IOOutputPortList/>   <!-- opt -->
</IOPortList>
```

7.5.1. /PSIA/System/IO/status

URI	/PSIA/System/IO/status		Type	Resource
Function	Query the IO status.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IOPortStatusList>	
Notes	<ioPortID> refers to /PSIA/System/IO/inputs/*ID* or /PSIA/System/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="urn:psialliance-org">
   <IOPortStatus>     <!-- opt -->
      <ioPortID>      <!-- req, xs:string;id -->                    </ioPortID>
      <ioPortType>    <!-- req, xs:string, "input,output" -->       </ioPortType>
      <ioState>       <!-- req, xs:string, "active,inactive" -->    </ioState>
   </IOPortStatus>
</IOPortStatusList>
```

7.5.2. /PSIA/System/IO/inputs

URI	/PSIA/System/IO/inputs		Type	Resource
Function	Access input ports.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<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="urn:psialliance-org">
   <IOInputPort/>     <!-- opt -->
```

```
</IOInputPort>
```

7.5.3. /PSIA/System/IO/inputs/<ID>

URI	/PSIA/System/IO/inputs/*ID*		Type	Resource
Function	Access a particular input port.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<IOInputPort>	
PUT		<IOInputPort>	<ResponseStatus>	
Notes	<triggeringType> indicates the signal conditions to trigger the input port. Rising/Falling refer to a rising/falling edge of a signal. High/Low will continuously trigger for the duration of the high/low input signal.			

IOInputPort XML Block

```
<IOInputPort version="1.0"   xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->                          </id>
   <triggeringType>  <!-- req, xs:string, "high,low,rising,falling" -->  </triggeringType>
</IOInputPort>
```

7.5.4. /PSIA/System/IO/inputs/<ID>/status

URI	/PSIA/System/IO/inputs/*ID*/status		Type	Resource
Function	Query the status of an input port.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<IOPortStatus>	
Notes	See /PSIA/System/IO/status for an explanation of the fields.			

7.5.5. /PSIA/System/IO/outputs

URI	/PSIA/System/IO/outputs		Type	Resource
Function	Access output ports.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<IOOutputPortList>	
Notes	IO outputs are hardwired, meaning that the inputs are statically allocated by the device and cannot be created or deleted.			

IOOutputPortList XML Block

```
<IOOutputPortList version="1.0" xmlns="urn:psialliance-org">
   <IOOutputPort/>    <!-- opt -->
</IOOutputPort>
```

7.5.6. /PSIA/System/IO/outputs/<ID>

URI	/PSIA/System/IO/outputs/*ID*		Type	Resource
Function	Access a particular output port.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IOOutputPort>	
PUT		<IOOutputPort>	<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 pulse output port signal when it is being triggered. It must be provided if the <outputState> is “pulse”. <actionMapping> is used in interfaces that allow configuration of “On” and “Off” for "High" and "Low" signals.			

IOOutputPort XML Block

```
<IOOutputPort version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->           </id>
   <PowerOnState>        <!-- req -->
       <defaultState>    <!-- req, xs:string, "high,low" -->          </defaultState>
       <outputState>     <!-- req, xs:string, "high,low,pulse" -->    </outputState>
       <pulseDuration>   <!-- dep, xs:integer, milliseconds -->   </pulseDuration>
   </PowerOnState>
   <ManualControl>       <!-- opt -->
       <actionMapping>
          <!-- req, xs:string, "high,low": ON maps to high / ON maps to low -->
       </actionMapping>
   </ManualControl>
</IOOutputPort>
```

7.5.7. /PSIA/System/IO/outputs/<ID>/trigger

URI	/PSIA/System/IO/outputs/*ID*/trigger		Type	Resource
Function	Manually trigger an output port.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	outputState pulseDuration	<IOPortData>	<ResponseStatus>	
Notes	Either the inbound data or query string values are used. The IO output port is toggled to a high or low signal accordingly. If the <outputState> refers to pulse, then the <pulseDuration> tag must be provided and the output port will be triggered to the specified state for the duration specified by <pulseDuration>.			

IOPortData XML Block

```
<IOPortData xmlns="urn:psialliance-org">
   <outputState>     <!-- req, xs:string, "high,low,pulse" -->    </outputState>
   <pulseDuration>   <!-- dep, xs:integer, milliseconds -->      </pulseDuration>
</IOPortData>
```

7.5.8. /PSIA/System/IO/outputs/<ID>/status

URI	/PSIA/System/IO/inputs/*ID*/status		Type	Resource
Function	Query the status of an output port.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<IOPortStatus>	
Notes	See /PSIA/System/IO/status for an explanation of the fields.			

7.5.9. IO Port Examples

Example: Set up IO Port Triggering

NOTE: The following example requires that input port event detection and output port triggering be enabled and scheduled with /PSIA/Custom/Event/triggers and /PSIA/Custom/Event/schedule beforehand.

The following commands set up one device input port and two device output ports (the number of IO ports is device-dependent) in the following manner:

- Input port 111 will continuously trigger an event (specified in the example in section 7.15.16) when the input signal is high. The input port should stop triggering this event when the input signal reverts back to low.
- Output port 222 will have a default low signal when not being triggered. When triggered, it will switch to a high signal. The port should automatically revert to a low signal when triggering stops, but in the case that a device cannot support this feature the port can be manually reset (see 7.15.16).
- Output port 333 will have a default low signal when not being triggered. When triggered, it will send a "pulse" of the opposite signal - high, in this case - for a duration of 3 seconds and then switch back to a low signal.

```
PUT /PSIA/System/IO/inputs/111 HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<IOInputPort>
   <triggeringType>high</triggeringType>
</IOInputPort>
```

```
PUT /PSIA/System/IO/outputs/222 HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<IOOutputPort>
   <PowerOnState>
       <defaultStateType>low</defaultStateType>
       <outputStateType>high</outputStateType>
   </PowerOnState>
</IOOutputPort>
```

```
PUT /PSIA/System/IO/outputs/333 HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<IOOutputPort>
   <PowerOnState>
       <defaultStateType>low</defaultStateType>
       <outputStateType>pulse</outputStateType>
       <pulseDuration>3000</pulseDuration>
   </PowerOnState>
</IOOutputPort>
```

Example: Manually Trigger and Reset an Output Port

Use the following command to manually set to a low signal. Note that this feature has no effect on future event detection and triggering – e.g. if output port 1 is automatically triggered in the future, it will override the behavior set here.

```
PUT /PSIA/System/IO/outputs/222/trigger HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<IOPortData xmlns="urn:psialliance-org">
   <outputState>low</outputState>
</IOPortData>
```

or, the same without the XML payload:

```
PUT /PSIA/System/IO/outputs/222/trigger?outputState=low HTTP/1.1
```

7.6. /PSIA/System/Audio

URI	/PSIA/System/Audio		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Audio service.			

7.6.1. /PSIA/System/Audio/channels

URI	/PSIA/System/Audio/channels		Type	Resource
Function	Access audio channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET		None	<AudioChannelList>	
Notes	Since inputs are resources that are defined by the hardware configuration of the device, audio channels cannot be created or deleted. ID numbering or values should be considered arbitrary and device-dependent.			

AudioChannelList XML Block

```
<AudioChannelList version="1.0" xmlns="urn:psialliance-org">
   <AudioChannel/>    <!-- opt -->
</AudioChannelList>
```

7.6.2. /PSIA/System/Audio/channels/<ID>

URI	/PSIA/System/Audio/channels/*ID*		Type	Resource
Function	Access a particular audio channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET		None	<AudioChannel>	
PUT		<AudioChannel>	<ResponseStatus>	
Notes	<audioMode> is the duplex mode for audio transmission between the client and media device. <microphoneSource> indicates whether the device microphone is internal or external. <microphoneVolume> Volume control percentage for device microphone. 0 is mute. <speakerVolume> Volume control percentage for device speaker. 0 is mute.			

AudioChannel XML Block

```
<AudioChannel version="1.0" xmlns="urn:psialliance-org">
   <id>           <!-- req, xs:string;id -->                              </id>
   <enabled>      <!-- req, xs:boolean -->                               </enabled>
   <audioMode>
       <!-- req, xs:string, "listenonly,talkonly,talkorlisten,talkandlisten" -->
   </audioMode>
   <microphoneEnabled>    <!-- opt, xs:boolean -->                       </microphoneEnabled>
   <microphoneSource>     <!-- opt, xs:string, "internal,external" -->    </microphoneSource>
   <microphoneVolume>     <!-- opt, xs:integer, 0..100 -->                </microphoneVolume>
   <speakerEnabled>       <!-- opt, xs:boolean -->                        </speakerEnabled>
```

```
   <speakerVolume>       <!-- opt, xs:integer, 0..100 -->               </speakerVolume>
</AudioChannel>
```

7.7. /PSIA/System/Video

URI	/PSIA/System/Video		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Video service. Video outputs (i.e. decoding) will be covered in a future IPMD specification.			

7.7.1. /PSIA/System/Video/overlayImages

URI	/PSIA/System/Video/overlayImages		Type	Resource
Function	Manage overlay images.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<OverlayImageList>	
POST		Raw Image data	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	These are the bitmaps used by the image overlays for video channels. The image repository is centralized so that the same image can be used for multiple channels. <imageType> is the mime type of the image, i.e. “image/jpeg”. <imageWidth> and <imageHeight> are the width and height of the image. When issuing a POST of the image data, the client must set the HTTP `Content-Type:` header field to the correct MIME type for the image.			

OverlayImageList XML Block

```
<OverlayImageList version="1.0" xmlns="urn:psialliance-org">
   <OverlayImage>     <!-- opt -->
      <id>            <!-- req, xs:string -->                  </id>
      <imageType>     <!-- req, xs:string, "JPEG,GIF,PNG" -->  </imageType>
      <imageWidth>    <!-- req, xs:integer;coordinate -->      </imageWidth>
      <imageHeight>   <!-- req, xs:integer;coordinate-->       </imageHeight>
   </OverlayImage>
</OverlayImageList>
```

7.7.2. /PSIA/System/Video/overlayImages/<ID>

URI	/PSIA/System/Video/overlayImages/*ID*		Type	Resource
Function	Access the overlay image for a particular channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Raw Image data	
PUT		Raw Image data	<ResponseStatus>	
DELET			<ResponseStatus>	
Notes	Overlay images can be updated using the PUT command and deleted using the DELETE command.			

7.7.3. /PSIA/System/Video/inputs

URI	/PSIA/System/Video/inputs		Type	Resource
Function	Access the video inputs on an IP media device.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<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. ID numbering or values should be considered arbitrary and device-dependent.			

VideoInput XML Block

```
<VideoInput version="1.0" xmlns="urn:psialliance-org">
   <VideoInputChannelList/>  <!-- opt -->
</VideoInput>
```

7.7.4. /PSIA/System/Video/inputs/channels

URI	/PSIA/System/Video/inputs/channels		Type	Resource
Function	Access video input channels.			
Methods	Query String(s)	Inbound Data	Return Result	
GET		None	<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="urn:psialliance-org">
   <VideoInputChannel/>      <!-- opt -->
</VideoInputChannelList>
```

7.7.5. /PSIA/System/Video/inputs/channels/<ID>

URI	/PSIA/System/Video/inputs/channels/*ID*		Type	Resource
Function	Access video input channel properties.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<VideoInputChannel>	
PUT		<VideoInputChannel>	<ResponseStatus>	
Notes	<powerLineFrequencyMode> is used to adjust/correct video image based on different power frequencies. <whiteBalanceMode> indicates the white balance operational mode. <whiteBalanceLevel> indicates the white balance percentage value when whiteBalanceMode refers to manual. 0 is 'cool', 100 is 'hot'. <exposureMode> indicates the exposure operational mode. <exposureTarget> the target exposure for manual or auto-exposure. <exposureAutoMin> minimum exposure when <exposureMode> is set to auto. <exposureAutoMax> maximum exposure when <exposureMode> is set to auto. <GainWindow> defines the coordinates of the window used to determine the auto-gain statistics, if smaller than the entire window. <gainLevel> indicates the gain level percentage value when <exposureMode> refers to Manual. 0 is low gain, 100 is high gain. <irisMode> indicates the iris operational mode. Only applicable for auto-iris lens modules. Override will put lens module into manual mode until the scene changes, at which point operation is switched to the auto mode. <focusMode> indicates the focus operational mode. Only applicable for auto-focus lens modules. Override will put lens module into manual mode until the scene changes, at which point operation is switched to the auto mode. In <DayNightFilter>, <beginTime> and <endTime> are only used if <switchScheduleEnabled> is true.			

VideoInputChannel XML Block

```
<VideoInputChannel version="1.0" xmlns="urn:psialliance-org">
   <id>                      <!-- req, xs:string;id -->              </id>
   <inputPort>               <!-- req, xs:string -->                 </inputPort>
   <powerLineFrequencyMode>  <!-- opt, xs:string "50hz, 60hz" --> </powerLineFrequencyMode>
   <whiteBalanceMode>
       <!-- opt, xs:string,
           "manual,auto,indoor/incandescent,fluorescent/white,
           fluorescent/yellow,outdoor,black&white"
       -->
   </whiteBalanceMode>
   <whiteBalanceLevel>       <!-- dep, xs:integer, 0..100 -->        </whiteBalanceLevel>
   <exposureMode>            <!-- opt, xs:string, "manual, auto" -->  </exposureMode>
   <Exposure>                <!-- opt -->
       <exposureTarget>      <!-- req, xs:integer, microseconds -->   </exposureTarget>
       <exposureAutoMin>     <!-- req, xs:integer, microseconds -->   </exposureAutoMin>
       <exposureAutoMax>     <!-- req, xs:integer, microseconds -->   </exposureAutoMax>
   </Exposure>
   <GainWindow>                  <!-- opt -->
       <RegionCoordinatesList>   <!-- opt -->
           <RegionCoordinates>   <!-- opt -->
               <positionX>       <!-- req, xs:integer;coordinate -->  </positionX>
               <positionY>       <!-- req, xs:integer;coordinate -->  </positionY>
           </RegionCoordinates>
```

```
        </RegionCoordinatesList>
    </GainWindow>
    <gainLevel>                <!-- dep, xs:integer, 0..100 -->      </gainLevel>
    <brightnessLevel>          <!-- opt, xs:integer, 0..100 -->      </brightnessLevel>
    <contrastLevel>            <!-- opt, xs:integer, 0..100 -->      </contrastLevel>
    <sharpnessLevel>           <!-- opt, xs:integer, 0..100 -->      </sharpnessLevel>
    <saturationLevel>          <!-- opt, xs:integer, 0..100 -->      </saturationLevel>
    <hueLevel>                 <!-- opt, xs:integer, 0..100 -->      </hueLevel>
    <gammaCorrectionEnabled>   <!-- opt, xs:boolean -->              </gammaCorrectionEnabled>
    <gammaCorrectionLevel>     <!-- opt, xs:integer, 0..100  -->     </gammaCorrectionLevel>
    <WDREnabled>               <!-- opt, xs:boolean -->              </WDREnabled>
    <WDRLevel>                 <!-- opt, xs:integer, 0..100 -->      </WDRLevel>
    <LensList>                 <!-- opt -->
        <Lens>                 <!-- opt -->
            <lensModuleName>   <!-- opt, xs:string -->               </lensModuleName>
            <irisMode>
                <!-- opt, xs:string, "manual,auto,override" -->
            </irisMode>
            <focusMode>
                <!-- opt, xs:string, "manual,auto,autobackfocus,override" -->
            </focusMode>
        </Lens>
    </LensList>
    <DayNightFilter>           <!-- opt -->
        <dayNightFilterType>
            <!-- req, xs:string, "day,night,auto" -->
        </dayNightFilterType>
        <switchScheduleEnabled><!-- opt, xs:boolean -->              </switchScheduleEnabled>
        <beginTime>            <!-- dep, xs:time -->                 </beginTime>
        <endTime>              <!-- dep, xs:time -->                 </endTime>
    </DayNightFilter>
<VideoInputChannel>
```

7.7.6. /PSIA/System/Video/inputs/channels/<ID>/focus

URI	/PSIA/System/Video/inputs/channels/*ID*/focus		Type	Resource
Function	Manually focus a video input channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	focus	<FocusData>	<ResponseStatus>	
Notes	<focus>: 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.			

FocusData XML Block

```
<FocusData version="1.0" xmlns="urn:psialliance-org">
    <focus>    <!-- req, xs:intger, -100..100 -->   </focus>
</FocusData>
```

7.7.7. /PSIA/System/Video/inputs/channels/<ID>/iris

URI	/PSIA/System/Video/inputs/channels/*ID*/iris		Type	Resource
Function	Manually adjust iris for a video input channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	iris	<IrisData>	<ResponseStatus>	
Notes	<iris> negative numbers close iris, positive numbers open iris. Numerical value is a percentage of the maximum iris speed of the lens module.			

IrisData XML Block

```
<IrisData version="1.0" xmlns="urn:psialliance-org">
   <iris>     <!-- req, xs:integer, -100..100 -->  </iris>
</IrisData>
```

7.7.8. /PSIA/System/Video/inputs/channels/<ID>/lens

URI	/PSIA/System/Video/inputs/channels/*ID*/lens		Type	Resource
Function	Query lens information.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<LensStatus>	
Notes	<absoluteFocus> indicates the current absolute focus position. 0 is focus near, 100 is focus far. <absoluteIris> indicates the current absolute iris position. 0 is completely closed, 100 is completely open.			

LensStatus XML Block

```
<LensStatus version="1.0" xmlns="urn:psialliance-org">
   <Absolute>
      <absoluteFocus>    <!-- req, xs:integer, 0..100 --> </absoluteFocus>
      <absoluteIris>     <!-- req, xs:integer, 0..100 --> </absoluteIris>
   </Absolute>
</LensStatus>
```

7.7.9. /PSIA/System/Video/inputs/channels/<ID>/overlays

URI	/PSIA/System/Video/channels/*ID*/overlays		Type	Resource
Function	Configure and access text and image overlays.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<VideoOverlay>	
PUT		<VideoOverlay>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	IP media devices can overlay additional information on the encoded video stream. These overlays can be either text information or a set of images. Overlays are composited together in ID-order when displayed in the video. Overlay images are managed with /PSIA/System/Video/overlayImages.			

VideoOverlay XML Block

```
<VideoOverlay version="1.0" xmlns="urn:psialliance-org">
   <TextOverlayList/>    <!-- opt -->
   <ImageOverlayList/>   <!-- opt -->
</VideoOverlay>
```

7.7.10. /PSIA/System/Video/inputs/channels/<ID>/overlays/text

URI	/PSIA/System/Video/channels/*ID*/overlays/text		Type	Resource
Function	Access and configure text overlays for a particular video channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<TextOverlayList>	
PUT		<TextOverlayList>	<ResponseStatus>	
POST		<TextOverlay>	<ResponseStatus>	
DELETE			<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="urn:psialliance-org">
   <TextOverlay/>    <!-- opt -->
</TextOverlayList>
```

7.7.11. /PSIA/System/Video/inputs/channels/<ID>/overlays/text/<ID>

URI	/PSIA/System/Video/channels/*ID*/overlays/text/*ID*		Type	Resource
Function	Access and configure a particular text overlay for a video channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			TextOverlay>	
PUT		<TextOverlay>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A text overlay can contain time information and static text with color and transparency information.			

TextOverlay XML Block

```
<TextOverlay version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->       </id>
   <enabled>             <!-- req, xs:boolean -->         </enabled>
   <timeStampEnabled>    <!-- opt, xs:boolean -->         </timeStampEnabled>
   <dateTimeFormat>      <!-- dep, xs:string -->          </dateTimeFormat>
   <backgroundColor>     <!-- opt, xs:hexBinary;color --> </backgroundColor>
   <fontColor>           <!-- opt, xs:hexBinary;color --> </fontColor>
   <fontSize>            <!-- opt, xs:integer, pixels --> </fontSize>
   <displayText>         <!-- req, xs:string -->          </displayText>
   <horizontalAlignType> <!-- opt, xs:string, "left,right,center" --> </horizontalAlignType>
   <verticalAlignType>   <!-- opt, xs:string, "top,bottom" -->       </verticalAlignType>
```

```
</TextOverlay>
```

7.7.12. /PSIA/System/Video/inputs/channels/<ID>/overlays/image

URI	/PSIA/System/Video/channels/*ID*/overlays/image		Type	Resource
Function	Access and configure image overlays for a particular video channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<ImageOverlayList>	
PUT		<ImageOverlayList>	<ResponseStatus>	
POST		<ImageOverlay>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A set of image overlays is managed. They are composited over the video signal in increasing ID-order.			

ImageOverlayList XML Block

```
<ImageOverlayList version="1.0" xmlns="urn:psialliance-org">
   <ImageOverlay/>    <!-- opt -->
</ImageOverlayList>
```

7.7.13. /PSIA/System/Video/inputs/channels/<ID>/overlays/image/<ID>

URI	/PSIA/System/Video/channels/*ID*/overlays/image/*ID*		Type	Resource
Function	Access and configure a particular image overlay for a video channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<ImageOverlay>	
PUT		<ImageOverlay>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	An image overlay can contain time information and static text with color and transparency information. In order to enable image overlay, an image must have been previously uploaded to the device using the /PSIA/System/Video/overlayImages command.			

ImageOverlay XML Block

```
<ImageOverlay version="1.0" xmlns="urn:psialliance-org">
   <id>                      <!-- req, xs:string;id -->           </id>
   <enabled>                 <!-- req, xs:boolean -->             </enabled>
   <imageName>               <!-- req, xs:string -->              </imageName>
   <positionX>               <!-- opt, xs:integer;coordinate -->  </positionX>
   <positionY>               <!-- opt, xs:integer;coordinate -->  </positionY>
   <transparentColorEnabled> <!-- opt, xs:boolean -->             </transparentColorEnabled>
   <transparentColor>        <!-- dep, xs:hexBinary;color -->     </transparentColor>
</ImageOverlay>
```

7.7.14. /PSIA/System/Video/inputs/channels/<ID>/privacyMask

URI	/PSIA/System/Video/channels/*ID*/privacyMask		Type	Resource
Function	Access and configure privacy masking.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PrivacyMask>	
PUT		<PrivacyMask>	<ResponseStatus>	
Notes	Privacy masking can be enabled and the region list configured per channel.			

PrivacyMask XML Block

```
<PrivacyMask version="1.0" xmlns="urn:psialliance-org">
   <enabled>                <!-- req, xs:boolean -->         </enabled>
   <PrivacyMaskRegionList/>  <!-- opt -->
</PrivacyMask>
```

7.7.15. /PSIA/System/Video/inputs/channels/<ID>/privacyMask/regions

URI	/PSIA/System/Video/channels/*ID*/privacyMask/regions		Type	Resource
Function	Access and configure privacy mask regions.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PrivacyMaskRegionList>	
PUT		<PrivacyMaskRegionList>	<ResponseStatus>	
POST		<PrivacyMaskRegion>	<ResponseStatus>	
DELETE			<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="urn:psialliance-org">
   <PrivacyMaskRegion/>  <!-- opt -->
</PrivacyMaskRegionList>
```

7.7.16. /PSIA/System/Video/inputs/channels/<ID>/privacyMask/regions/<ID>

URI	/PSIA/System/Video/channels/*ID*/privacyMask/regions/*ID*		Type	Resource
Function	Access and configure a particular privacy mask region.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PrivacyMaskRegion>	
PUT		<PrivacyMaskRegion>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Region coordinates are dependent on video resolution. Regions will be “drawn” from the coordinates provided in a top-down fashion. At least three <RegionCoordinates> blocks must be provided for a single <PrivacyMaskRegion> block. Ordering of <PrivacyMaskRegion> blocks is insignificant.			

PrivacyMaskRegion XML Block

```
<PrivacyMaskRegion version="1.0" xmlns="urn:psialliance-org">
   <id>                     <!-- req, xs:string;id -->              </id>
   <enabled>                <!-- req, xs:boolean -->                <enabled>
   <RegionCoordinatesList>  <!-- req -->
      <RegionCoordinates>   <!-- req, at least one if list is defined -->
         <positionX>        <!-- req, xs:integer;coordinate -->     </positionX>
         <positionY>        <!-- req, xs:integer;coordinate  -->    </positionY>
      </RegionCoordinates>
   </RegionCoordinatesList>
</PrivacyMaskRegion>
```

7.8. /PSIA/System/Serial

URI	/PSIA/System/Serial		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Serial line service.			

7.8.1. /PSIA/System/Serial/ports

URI	/PSIA/System/Serial/ports		Type	Resource
Function	List of serial ports supported by the device.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SerialPortList>	
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="urn:psialliance-org">
   <SerialPort/>  <!-- opt -->
</SerialPortList>
```

7.8.2. /PSIA/System/Serial/ports/<ID>

URI	/PSIA/System/Serial/ports/*ID*		Type	Resource
Function	Serial port			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<SerialPort>	
PUT		<SerialPort>	<ResponseStatus>	
Notes	Access to the serial port parameters. <serialPortType> set the type of port; RS232, RS485, etc. <direction> indicates whether the port is bidirectional. <duplexMode> indicates whether the serial port runs in full or half duplex mode.			

SerialPort XML Block

```
<SerialPort version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->                          </id>
   <enabled>         <!-- req, xs:boolean -->                           </enabled>
   <serialPortType>  <!-- req, xs:string, "RS485,RS422,RS232" -->       </serialPortType>
   <duplexMode>      <!-- req, xs:string, "half,full" -->               </duplexMode>
   <direction>       <!-- req, xs:string, "monodirectional,bdirectional" --> </direction>
   <baudRate>        <!-- req, xs:integer -->                           </baudRate>
   <dataBits>        <!-- req, xs:integer -->                           </dataBits>
   <parityType>      <!-- req, xs:string, "none,even,odd,mark,space" --> </parityType>
   <stopBits>        <!-- req, xs:string, "1,1.5,2" -->                 </stopBits>
</SerialPort>
```

7.8.3. /PSIA/System/Serial/ports/<ID>/command

URI	/PSIA/System/Serial/ports/*ID*/command		Type	Resource
Function	Send a command to a serial port.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	chainNo	<SerialCommand> Raw Data	<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`. In this case, the chainNo query string parameter should be used.			

SerialCommand XML Block

```
<SerialCommand version="1.0" xmlns="urn:psialliance-org">
   <chainNo>      <!-- opt, xs:string -->   </chainNo>
   <command>      <!-- req, xs:hexBinary --></command>
</SerialCommand>
```

Example

Send the command using an XML block:

```
PUT /PSIA/System/Serial/ports/999/command HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<SerialCommand>
   <chainNo>0</chainNo>
   <command>ab45be8778cd</command>
</SerialCommand>
```

Send the command using query strings and a binary payload:

```
PUT /PSIA/System/Serial/ports/999/command?chainNo=1 HTTP/1.1
Content-Type: application/octet-stream
Content-Length: xxx

(…Raw bytes of command follow here…)
```

7.9. /PSIA/Diagnostics

URI	/PSIA/Diagnostics		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Diagnostic services.			

7.9.1. /PSIA/Diagnostics/commands

URI	/PSIA/Diagnostics/commands		Type	Resource
Function	Access diagnostic commands.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<DiagnosticCommandList>	
POST		<DiagnosticCommand>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Diagnostic commands are device specific and run asynchronously. A client uses POST to issue a new command that runs in the background. During the time is it running, its status can be queried by issuing an HTTP GET on its URL: /PSIA/Diagnostics/commands/*ID*. <status> and <resultMessage> are read-only. DELETE removes all diagnostic commands that are running. Devices may chose to clear the list of completed commands at any reasonable time after they have completed, subject to available storage space. Command results may be cleared when the device is rebooted. The command itself is free-form and device-dependent. <status> indicates the status of the command: it passed, it failed or it is still running. <resultMessage> is a string that describes the outcome of the command more in detail.			

7.9.2. /PSIA/Diagnostics/commands/<ID>

URI	/PSIA/Diagnostics/commands/*ID*		Type	Resource
Function	Access a particular diagnostics command.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<DiagnosticCommand>	
DELETE			<ResponseStatus>	
Notes	DELETE can be used to remove an already-completed command or interrupt and delete a running diagnostic command.			

7.9.3. Diagnostics XML Data

DiagnosticCommandList XML Block

```
<DiagnosticCommandList version="1.0" xmlns="urn:psialliance-org">
   <DiagnosticCommand/>  <!-- opt -->
</DiagnosticCommandList>
```

DiagnosticCommand XML Block

```
<DiagnosticCommand version="1.0" xmlns="urn:psialliance-org">
   <command>          <!-- req, xs:string -->                              </command>
   <status>           <!-- ro, req, xs:string, "pass,fail,running" -->    </status>
   <resultMessage>    <!-- ro, req, xs:string -->                          </resultMessage>
</DiagnosticCommand>
```

7.10./PSIA/Security

URI	/PSIA/Security		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Security service.			

7.10.1. /PSIA/Security/srtpMasterKey

URI	/PSIA/Security/srtpMasterKey		Type	Resource
Function	Access the SRTP master key.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Data	
PUT		Data	<ResponseStatus>	
Notes	See RFC3711 for a description of SRTP.			

7.10.2. /PSIA/Security/deviceCertificate

URI	/PSIA/Security/deviceCertificate		Type	Resource
Function	This function is used to upload a user certificate to the device. The user certificate is used for 802.1x (radius) with various authentication mechanisms.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Data	
PUT		Data	<ResponseStatus>	
Notes	The format of the certificate is device-dependent.			

7.11./PSIA/Security/AAA

URI	/PSIA/Security/AAA		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Authentication, authorization and auditing service.			

7.11.1. /PSIA/Security/AAA/users

URI	/PSIA/Security/AAA/users		Type	Resource
Function	Access the device user list.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<UserList>	
PUT		<UserList>	<ResponseStatus>	
POST		<User>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes	It is possible to add, remove and update users entries in the list. Passwords can only be uploaded - they are never revealed during GET operations.

UserList XML Block

```
<UserList version="1.0" xmlns="urn:psialliance-org">
   <User/>    <!-- opt -->
</UserList>
```

7.11.2. /PSIA/Security/AAA/users/<ID>

URI	/PSIA/Security/users/*ID*		Type	Resource
Function	Authentication user settings.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<User>	
PUT		<User>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Each <protocolID> tag, if <ProtocolList> is provided, must match a corresponding <id> tag in /PSIA/Security/adminAccesses. **Note:** <password> is a write-only field.			

User XML Block

```
<User version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->       </id>
   <userName>            <!-- req, xs:string -->          </userName>
   <password>            <!-- wo, req, xs:string -->      </password>
   <ProtocolList>        <!-- opt -->
      <Protocol>         <!-- opt -->
         <protocolID>    <!-- req, xs:string;id -->       </protocolID>
      </Protocol>
   </ProtocolList>
</User>
```

7.11.3. /PSIA/Security/AAA/certificate

URI	/PSIA/Security/AAA/certificate		Type	Resource
Function	Access the device certificate.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			Data	
PUT		Data	<ResponseStatus>	
Notes	The certificate format is device-dependent.			

7.11.4. /PSIA/Security/AAA/adminAccesses

URI	/PSIA/Security/adminAccesses		Type	Resource
Function	Administrative access protocols for the device.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AdminAccessProtocolList>	
PUT		<AdminAccessProtocolList>	<ResponseStatus>	
POST		<AdminAccessProtocol>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Allows configuration of the set of protocols that allow administrative access.			

AdminAccessProtocolList XML Block

```
<AdminAccessProtocolList version="1.0" xmlns="urn:psialliance-org">
   <AdminAccessProtocol/>    <!-- opt -->
</AdminAccessProtocolList>
```

7.11.5. /PSIA/Security/AAA/adminAccesses/<ID>

URI	/PSIA/Security/adminAccesses/*ID*		Type	Resource
Function	Administrative access and protocol settings.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<AdminAccessProtocol>	
PUT		<AdminAccessProtocol>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<protocol> is the protocol name for admin access, i.e. “HTTP”, “HTTPS”, etc.			

AdminAccessProtocol XML Block

```
<AdminAccessProtocol version="1.0" xmlns="urn:psialliance-org">
   <id>          <!-- req, xs:string;id -->              </id>
   <enabled>     <!-- req, xs:boolean -->                </enabled>
   <protocol>    <!-- req, xs:string, "HTTP,HTTPS" -->   </protocol>
   <portNo>      <!-- req, xs:integer -->                </portNo>
</AdminAccessProtocol>
```

7.12./PSIA/Streaming

URI	/PSIA/Streaming		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	Streaming service			

7.12.1. /PSIA/Streaming/status

URI	/PSIA/Streaming/status		Type	Resource
Function	Query the device streaming status.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<StreamingStatus>	
Notes	This command accesses the status of all device streaming sessions.			

StreamingStatus XML Block

```
<StreamingStatus version="1.0" xmlns="urn:psialliance-org">
   <totalStreamingSessions>         <!-- req, xs:integer -->  </totalStreamingSessions>
   <StreamingSessionStatusList/>    <!-- dep, only if there are sessions -->
</StreamingStatus>
```

7.12.2. /PSIA/Streaming/channels

URI	/PSIA/Streaming/channels		Type	Resource
Function	Streaming channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<StreamingChannelList>	
PUT		<StreamingChannelList>	<ResponseStatus>	
POST		<StreamingChannel>	<ResponseStatus>	
DELETE			<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 /PSIA/Streaming/channels/description.			

StreamingChannelList XML Block

```
<StreamingChannelList version="1.0" xmlns="urn:psialliance-org">
   <StreamingChannel/>   <!-- opt -->
</StreamingChannelList>
```

7.12.3. /PSIA/Streaming/channels/<ID>

URI	/PSIA/Streaming/channels/*ID*		Type	Resource
Function	Access streaming channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<StreamingChannel>	
PUT		<StreamingChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<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. <videoSourcePortNo> and <audioSourcePortNo> are the source port numbers for the outbound video or audio streams. <videoInputChannelID> refers to /PSIA/System/Video/inputs/channel/*ID*. <audioInputChannelID> refers to /PSIA/System/Audio/channels/*ID*. It must be configured as an input channel. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress. <Security> determines whether SRTP is used for stream encryption. <audioResolution> is the resolution for the outbound audio stream in bits.			

StreamingChannel XML Block

```
<StreamingChannel version="1.0" xmlns="urn:psialliance-org">
   <id>           <!-- req, xs:string;id -->    </id>
   <channelName>  <!-- req, xs:string -->       </channelName>
   <enabled>      <!-- req, xs:boolean -->      </enabled>
   <Transport>    <!-- req -->
      <rtspPortNo>              <!-- opt, xs:integer -->      </rtspPortNo>
      <maxPacketSize>           <!-- opt, xs:integer -->      </maxPacketSize>
      <audioPacketLength>       <!-- opt, xs:integer -->      </audioPacketLength>
      <audioInboundPacketLength><!-- opt, xs:integer -->      </audioInboundPacketLength>
      <audioInboundPortNo>      <!-- opt, xs:integer -->      </audioInboundPortNo>
      <videoSourcePortNo>       <!-- opt, xs:integer -->      </videoSourcePortNo>
      <audioSourcePortNo>       <!-- opt, xs:integer -->      </audioSourcePortNo>
      <ControlProtocolList>     <!-- req -->
         <ControlProtocol>      <!-- req -->
            <streamingTransport>
               <!-- req, xs:string, "HTTP,RTSP" -->
            </streamingTransport>
         </ControlProtocol>
      </ControlProtocolList>
      <Unicast>                 <!-- opt -->
         <enabled>              <!-- req, xs:boolean -->         </enabled>
         <interfaceID>          <!-- opt, xs:string -->          </interfaceID>
         <rtpTransportType>
            <!-- opt, xs:string, "RTP/UDP,RTP/TCP" -->
         </rtpTransportType>
      </Unicast>
      <Multicast>               <!-- opt -->
         <enabled>              <!-- req, xs:boolean -->         </enabled>
         <userTriggerThreshold><!-- opt, xs:integer -->         </userTriggerThreshold>
         <destIPAddress>        <!-- dep, xs:string -->          </destIPAddress>
```

```
        <videoDestPortNo>     <!-- opt, xs:integer -->         </videoDestPortNo>
        <audioDestPortNo>     <!-- opt, xs:integer -->         </audioDestPortNo>
        <destIPv6Address>     <!-- dep, xs:string -->          </destIPv6Address>
        <ttl>                 <!-- opt, xs:integer -->         </ttl>
    </Multicast>
    <Security>                <!-- opt -->
        <enabled>             <!-- req, xs:boolean -->         </enabled>
    </Security>
  </Transport>
  <Video>                     <!-- opt -->
    <enabled>                 <!-- req, xs:boolean -->         </enabled>
    <videoInputChannelID>     <!-- req, xs:string;id -->       </videoInputChannelID>
    <videoCodecType>
        <!-- req, xs:string, "MPEG4,MJPEG,3GP,H.264,MPNG" -->
    </videoCodecType>
    <videoScanType>
        <!-- opt, xs:string, "progressive,interlaced" -->
    </videoScanType>
    <videoResolutionWidth>    <!-- req, xs:integer -->         </videoResolutionWidth>
    <videoResolutionHeight>   <!-- req, xs:integer -->         </videoResolutionHeight>
    <videoPositionX>          <!-- opt, xs:integer -->         </videoPositionX>
    <videoPositionY>          <!-- opt, xs:integer -->         </videoPositionY>
    <videoQualityControlType>
        <!-- opt, xs:string, "cbr,vbr" -->
    </videoQualityControlType>
    <constantBitRate>  <!-- dep, xs:integer, in kbps -->        </constantBitRate>
    <fixedQuality>     <!-- opt, xs:integer, percentage, 0..100 -->    </fixedQuality>
    <vbrUpperCap>      <!-- dep, xs:integer, in kbps -->        </vbrUpperCap>
    <vbrLowerCap>      <!-- dep, xs:integer, in kbps -->        </vbrLowerCap>
    <maxFrameRate>     <!-- req, xs:integer, maximum frame rate x100 -->   </maxFrameRate>
    <keyFrameInterval> <!-- opt, xs:integer, milliseconds -->   </keyFrameInterval>
    <rotationDegree>   <!-- opt, xs:integer, degrees, 0..360 --></rotationDegree>
    <mirrorEnabled>    <!-- opt, xs:boolean -->                 </mirrorEnabled>
    <snapShotImageType><!-- opt, xs:string, "JPEG,GIF,PNG" --> </snapShotImageType>
  </Video>
  <Audio>                     <!-- opt -->
    <enabled>                 <!-- req, xs:boolean -->         </enabled>
    <audioInputChannelID>     <!-- req, xs:string;id -->       </audioInputChannelID>
    <audioCompressionType>
        <!-- req, xs:string,
            "G.711alaw,G.711ulaw,G.726,G.729,G.729a,G.729b,PCM,MP3,AC3,AAC,ADPCM"
         -->
    </audioCompressionType>
    <audioInboundCompressionType>
        <!-- opt, xs:string,
            "G.711alaw,G.711ulaw,G.726,G.729,G.729a,G.729b,PCM,MP3,AC3,AAC,ADPCM"
         -->
    </audioInboundCompressionType>
    <audioBitRate>        <!-- opt, xs:integer, in kbps -->    </audioBitRate>
    <audioSamplingRate>   <!-- opt, xs:float, in kHz -->       </audioSamplingRate>
    <audioResolution>     <!-- opt, xs:integer, in bits -->    </audioResolution>
  </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 for the device while others may require that channels be manually configured before use.

```
GET /PSIA/Streaming/channels/444 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="urn:psialliance-org">
   <id>444</id>
   <channelName>Input 1 MPEG-4 ASP</channelName>
   <enabled>true<</enabled>
   <Transport>
       <rtspPortNo>554<</rtspPortNo>
       <maxPacketSize>1446<</maxPacketSize>
       <ControlProtocolList>
           <ControlProtocol>
               <streamingTransport>RTSP</streamingTransport>
           </ControlProtocol>
           <ControlProtocol>
               <streamingTransport>HTTP</streamingTransport>
           </ControlProtocol>
   </Transport>
   <Video>
       <enabled>true</enabled>
       <videoInputChannelID>2</videoInputChannelID>
       <videoCodecType>MPEG4</videoCodecType>
       <videoScanType>progressive</videoScanType>
       <videoResolutionWidth> 640</videoResolutionWidth>
       <videoResolutionHeight>480</videoResolutionHeight>
       <videoPositionX>0</videoPositionX>
       <videoPositionY>0</videoPositionY>
       <videoQualityControlType>CBR</videoQualityControlType>
       <constantBitRate>2000</constantBitRate>
       <maxFrameRate>2500</maxFrameRate>
       <keyFrameInterval>1000</keyFrameInterval>
       <rotationDegree>0</rotationDegree>
       <mirrorEnabled>false</mirrorEnabled>
       <snapShotImageType>JPEG</snapShotImageType>
   </Video>
   <Audio>
       <enabled>false</enabled>
       <audioInputChannelID>2</audioInputChannelID>
       <audioCompressionType> G.726</audioCompressionType>
       <audioBitRate>24</audioBitRate>
       <audioSamplingRate>8</audioSamplingRate>
   </Audio>
</StreamingChannel>
```

Example: Getting Streaming Capabilities

```
GET /PSIA/Streaming/channels/444/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="urn:psialliance-org">
```

```
<id opt="111,222,333,444">444</id>
<channelName min="0" max="64">Input 1 MPEG-4 ASP</channelName>
<enabled opt="true,false" def="true">true</enabled>
<Transport>
    <rtspPortNo min="0" max="65535" def="554">554</rtspPortNo>
    <maxPacketSize min="0" max="1500">1446</maxPacketSize>
    <audioPacketLength min="0" max="5000"/>
    <audioInboundPacketLength min="0" max="5000"/>
    <audioInboundPortNo min="0" max="65535"/>
    <videoSourcePortNo min="0" max="65535"/>
    <audioSourcePortNo min="0" max="65535"/>
    <ControlProtocolList>
        <ControlProtocol>
            <streamingTransport opt="RTSP/RTP,HTTP">RTSP</streamingTransport>
        </ControlProtocol>
        <ControlProtocol>
            <streamingTransport opt="RTSP/RTP,HTTP">HTTP</streamingTransport>
        </ControlProtocol>
    </ControlProtocolList>
    <Unicast>
        <enabled opt="true,false" def="false"/>
        <rtpTransportType opt="RTP/UDP,RTP/TCP"/>
    </Unicast>
    <Multicast>
        <enabled opt="true,false" def="false"/>
        <userTriggerThreshold/>
        <videoDestPortNo min="0" max="65535"/>
        <audioDestPortNo min="0" max="65535"/>
        <destIPAddress min="8" max="16"/>
        <destIPv6Address min="15" max="39"/>
        <ttl min="0" max="127" def="1"/>
    </Multicast>
    <Security>
        <enabled opt="true,false" def="false"/>
    </Security>
</Transport>
<Video>
    <enabled opt="true,false">true</enabled>
    <videoInputChannelID opt="1,2,3,4">2</videoInputChannelID>
    <videoCodecType opt="MJPEG,MPEG4">MPEG4</videoCodecType>
    <videoScanType opt="interlaced,progressive">progressive</videoScanType>
    <videoResolutionWidth min="0" max="640">640</videoResolutionWidth>
    <videoResolutionHeight min="0" max="480">480</videoResolutionHeight>
    <videoPositionX min="0" max="640">0</videoPositionX>
    <videoPositionY min="0" max="480">0</videoPositionY>
    <videoQualityControlType opt="CBR,VBR">CBR</videoQualityControlType>
    <constantBitRate min="50" max="4000" dynamic="true">2000</constantBitRate>
    <maxFrameRate opt="2500,1250,625,312,156,78" dynamic="true">2500</maxFrameRate>
    <keyFrameInterval min="0", max="10000">1000</keyFrameInterval>
    <rotationDegree opt="0,90,180,270" def="0">0</rotationDegree>
    <mirrorEnabled opt="true,false" def="false">false</mirrorEnabled>
    <snapShotImageType opt="JPEG" def="JPEG">JPEG</snapShotImageType>
</Video>
<Audio>
    <enabled opt="true,false" def="false">false</enabled>
    <audioInputChannelID opt="1,2,3,4">2</audioInputChannelID>
    <audioCompressionType opt="G.726,G.711ulaw" def="G.726">G.726</audioCompressionType>
    <audioBitRate opt="16,24,32,40" def="32" dynamic="true">24</audioBitRate>
    <audioSamplingRate opt="8" dynamic="true">8</audioSamplingRate>
    <audioResolution opt="3,4,5,6" dynamic="true"/>
</Audio>
```

```
</StreamingChannel>
```

Example: Setting Streaming Channel Properties

The following command sets the streaming parameters of an MJPEG stream with G.711 audio compression over RTSP and HTTP on streaming ID 555. The MJPEG codec is configured to encode a window of 640x480 positioned at 120,100 in the sensor field.

Some of the fields, such as <videoInputChannelID> and <audioInputChannelID> are already preconfigured for this streaming channel.

```
PUT /PSIA/Streaming/channels/333 HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<StreamingChannel version="1.0" xmlns="urn:psialliance-org">
   <channelName>Parking Garage Camera 1</channelName>
   <enabled>true</enabled>
   <Transport>
       <rtspPortNo>554</rtspPortNo>
       <ControlProtocolList>
           <ControlProtocol>
               <streamingTransport>RTSP</streamingTransport>
           </ControlProtocol>
           <ControlProtocol>
               <streamingTransport>HTTP</streamingTransport>
           </ControlProtocol>
       </ControlProtocolList>
   </Transport>
   <Video>
       <enabled>true</enabled>
       <videoCodecType>MJPEG</videoCodecType>
       <videoResolutionWidth>320</videoResolutionWidth>
       <videoResolutionHeight>240</videoResolutionHeight>
       <videoPositionX>100</videoPositionX>
       <videoPositionY>120</videoPositionY>
       <videoQualityControlType>VBR</videoQualityControlType>
       <fixedQuality>75</fixedQuality>
       <vbrUpperCap>10000</vbrUpperCap>
       <vbrLowerCap>2000</vbrLowerCap>
       <maxFrameRate>3000</maxFrameRate>
       <rotationDegree>90</rotationDegree>
       <mirrorEnabled>false</mirrorEnabled>
       <snapShotImageType>JPEG</snapShotImageType>
   </Video>
   <Audio>
       <enabled>true</enabled>
       <audioCompressionType>G711uaw</audioCompressionType>
       <audioBitRate>64</audioBitRate>
   </Audio>
</StreamingChannel>
```

7.12.4. /PSIA/Streaming/channels/<ID>/status

URI	/PSIA/Streaming/channels/*ID*/status		Type	Resource
Function	Get the list of streaming sessions associated with a particular channel.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<StreamingSessionStatusList>	
Notes	Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /PSIA/System/Network/interfaces/*ID*/ipAddress.			

StreamingSessionStatus XML Block

```
<StreamingSessionStatusList version="1.0" xmlns="urn:psialliance-org">
   <StreamingSessionStatus version="1.0" xmlns="urn:psialliance-org">
      <clientAddress>    <!-- req -->
         <ipAddress>     <!-- dep, xs:string -->              </ipAddress>
         <ipv6Address>   <!-- dep, xs:string -->              </ipv6Address>
      </clientAddress>
      <clientUserName>   <!-- opt, xs:string -->              </clientUserName>
      <startDateTime>    <!-- opt, xs:datetime -->            </startDateTime>
      <elapsedTime>      <!-- opt, xs:integer, seconds -->    </elapsedTime>
      <bandwidth>        <!-- opt, xs:integer, in kbps -->    </bandwidth>
   </StreamingSessionStatus>
</StreamingSessionStatusList>
```

7.12.5. /PSIA/Streaming/channels/<ID>/http

URI	/PSIA/Streaming/channels/*ID*/http		Type	Resource
Function	Access a live stream via http.			
Methods	Query String(s)	Inbound Data	Return Result	
GET	videoCodecType videoScanType videoResolutionWidth videoResolutionHeight videoPositionX videoPositionY videoQualityControlType constantBitRate fixedQuality vbrUpperCap vbrLowerCap maxFrameRate keyFrameInterval rotationDegree mirrorEnabled snapShotImageType		Stream over HTTP	
POST		<Video>		
Notes	This function is used to request a stream from the device using HTTP or HTTPS. This API uses HTTP server-push with the MIME type multipart/x-mixed-replace. HTTP streaming must be enabled on the channel. To determine the format of the video returned, either the parameters in <Video> or the query string values are used, depending on the capabilities of the encoder.			

	For supported values, query `/PSIA/Streaming/channels/ID/http/capabilities`.

Example

```
GET /PSIA/Streaming/channels/777/http?videoCodecType=MJPEG HTTP/1.1
…
HTTP/1.1 200 OK
Content-Type: multipart/x-mixed-replace; boundary=<boundary>
--<boundary>
Content-Type: image/jpeg
Content-Length: xxx

Image data for a single frame
--<boundary>
…
```

7.12.6. /PSIA/Streaming/channels/<ID>/picture

URI	`/PSIA/Streaming/channels/ID/picture`	**Type**	Resource
Function	Get a snapshot of the current image.		

Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET	videoResolutionWidth videoResolutionHeight videoPositionX videoPositionY rotationDegree mirrorEnabled snapShotImageType		Picture over HTTP
POST		<Video>	
Notes	All devices must support <snapShotImageType> of "JPEG". 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 `/PSIA/Streaming/channels/ID/picture/capabilities`. Examples: GET /PSIA/Streaming/channels/123456/picture?snapShotImageType=JPEG POST /PSIA/Streaming/channels/123456/picture … <?xml version="1.0" encoding="UTF-8"?> <Video>…</Video> GET /PSIA/Streaming/channels/123456/picture Accept: image/jpeg		

7.12.7. /PSIA/Streaming/channels/<ID>/requestKeyFrame

URI	`/PSIA/Streaming/channels/ID/requestKeyFrame`	**Type**	Resource
Function	Request that the device issue a key frame on a particular channel.		

Methods	**Query String(s)**	**Inbound Data**	**Return Result**

PUT			<ResponseStatus>
Notes	The key frame that is issued should include everything necessary to initialize a video decoder, i.e. parameter sets for H.264 or VOS for MPEG-4.		

7.13./PSIA/PTZ

URI	/PSIA/PTZ		Type	Service
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
Notes	PTZ control service.			

7.13.1. /PSIA/PTZ/channels

URI	/PSIA/PTZ/channels		Type	Resource
Function	Access the list of PTZ channels.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZChannelList>	
PUT		<PTZChannelList>	<ResponseStatus>	
POST		<PTZChannel>	<ResponseStatus>	
DELETE			<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 /PSIA/PTZ/channels/description.			

PTZChannelList XML Block

```
<PTZChannelList version="1.0" xmlns="urn:psialliance-org">
   <PTZChannel/>      <!-- opt -->
</PTZChannelList>
```

7.13.2. /PSIA/PTZ/channels/<ID>

URI	/PSIA/PTZ/channels/*ID*		Type	Resource
Function	Access or control a PTZ channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZChannel>	
PUT		<PTZChannel>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<videoInputID> links the PTZ channel to a video channel. <panMaxSpeed> defines or limits the maximum pan speed. <tiltMaxSpeed> defines or limits the maximum tilt speed. <autoPatrolSpeed> defines or limits the maximum patrol speed. <controlProtocol> indicates the control protocol to be used for PTZ. Supported protocols are device-dependent. <defaultPreset> identifies the default preset ID to be used with some interfaces.			

PTZChannel XML Block

```
<PTZChannel version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->                  </id>
   <enabled>         <!-- req, xs:boolean -->                    </enabled>
   <videoInputID>    <!-- req, xs:string;id -->                  </videoInputID>
   <panMaxSpeed>     <!-- opt, xs:integer, degrees/sec --></panMaxSpeed>
   <tiltMaxSpeed>    <!-- opt, xs:integer, degrees/sec --></tiltMaxSpeed>
   <autoPatrolSpeed> <!-- opt, xs:integer, 0..100 -->        </autoPatrolSpeed>
   <controlProtocol> <!-- opt, xs:string, "pelco-d,…" -->      </controlProtocol>
   <defaultPresetID> <!-- opt, xs:string;id -->                </defaultPresetID>
</PTZChannel>
```

7.13.3. /PSIA/PTZ/channels/<ID>/homePosition

URI	/PSIA/PTZ/channels/*ID*/homePosition		Type	Resource
Function	Set the home position of the PTZ camera to the current position			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT			<ResponseStatus>	
Notes	This function is used to set the current position as the absolute home position for a PTZ enabled device. After calling this API, the current position will act as the reference point for all absolute PTZ commands sent to the device (see 0).			

7.13.4. /PSIA/PTZ/channels/<ID>/continuous

URI	/PSIA/PTZ/channels/*ID*/continuous		Type	Resource
Function	Pans, tilts, and/or zooms the device in a continuous fashion.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	pan tilt zoom	<PTZData>	<ResponseStatus>	
Notes	The device shall not respond with a <ResponseStatus> until the PTZ command has been issued. The total round-trip time for this API should be less than 70 ms. Either the inbound data or query string values are used. <pan>: Negative numbers pan left, positive numbers pan right, 0 means stop. Numerical value is a percentage of panMaxSpeed. <tilt>: Negative numbers tilt down, positive numbers tilt up, 0 means stop. Numerical value is a percentage of tiltMaxSpeed. <zoom>: Negative numbers zoom out, positive numbers zoom in, 0 means stop. Numerical value is a percentage of the maximum zoom speed of the lens module. The auto patrol feature is stopped if it is running.			

Continuous PTZ Data XML Block

```
<PTZData version="1.0" xmlns="urn:psialliance-org">
   <pan>             <!-- opt, xs:integer, -100..100 -->      </pan>
   <tilt>            <!-- opt, xs:integer, -100..100 -->      </tilt>
   <zoom>            <!-- opt, xs:integer, -100..100 -->      </zoom>
```

```
</PTZData>
```

7.13.5. /PSIA/PTZ/channels/<ID>/momentary

URI	/PSIA/PTZ/channels/*ID*/momentary		Type	Resource
Function	Pans, tilts, and/or zooms the device in a momentary fashion.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	pan tilt zoom duration	<PTZData>	<ResponseStatus>	
Notes	The device shall not respond with a <ResponseStatus> until the PTZ command has been issued. The total round-trip time for this API should be less than 70 ms. Either the inbound data or query string values are used. <pan>: Negative numbers pan left, positive numbers pan right, 0 means stop. Numerical value is a percentage of panMaxSpeed. <tilt>: Negative numbers tilt down, positive numbers tilt up, 0 means stop. Numerical value is a percentage of tiltMaxSpeed. <zoom>: Negative numbers zoom out, positive numbers zoom in, 0 means stop. Numerical value is a percentage of the maximum zoom speed of the lens module.The device will move in the specified directions at the specified speeds for the amount of time specified in the <duration> tag, or until the device cannot move any longer in that particular direction. The auto patrol feature is stopped if it is running.			

Momentary PTZ Data XML Block

```
<PTZData version="1.0" xmlns="urn:psialliance-org">
   <pan>             <!-- opt, xs:integer, -100..100 -->      </pan>
   <tilt>            <!-- opt, xs:integer, -100..100 -->      </tilt>
   <zoom>            <!-- opt, xs:integer, -100..100 -->      </zoom>
   <Momentary>
      <duration>     <!-- req, xs:integer, milliseconds -->   </duration>
   </Momentary>
</PTZData>
```

7.13.6. /PSIA/PTZ/channels/<ID>/relative

URI	/PSIA/PTZ/channels/*ID*/relative		Type	Resource
Function	Pans, tilts, and/or zooms the device relative to the current position.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	positionX positionY relativeZoom	<PTZData>	<ResponseStatus>	

Notes	The device shall not respond with a <ResponseStatus> until the PTZ command has been issued. The total round-trip time for this API should be less than 70 ms. Either the inbound data or query string values are used. The <positionX> and <positionY> tags must be provided in relation to the currently set video resolution. The device will center on the provided coordinates. The <relativeZoom> tag roughly indicates what percentage to zoom in respect to the current image. The auto patrol feature is stopped if it is running.

Relative PTZ Data XML Block

```
<PTZData version="1.0" xmlns="urn:psialliance-org">
   <Relative>
      <positionX>    <!-- opt, xs:integer -->                </positionX>
      <positionY>    <!-- opt, xs:integer -->                </positionY>
      <relativeZoom> <!-- opt, xs:integer, -100..100 -->      </relativeZoom>
   </Relative>
</PTZData>
```

7.13.7. /PSIA/PTZ/channels/<ID>/absolute

URI	/PSIA/PTZ/channels/*ID*/absolute		**Type**	Resource
Function	Pans, tilts, and/or zooms the device relative to the absolute home position.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	elevation azimuth absoluteZoom	<PTZData>	<ResponseStatus>	
Notes	The device shall not respond with a <ResponseStatus> until the PTZ command has been issued. The total round-trip time for this API should be less than 70 ms. Either the inbound data or query string values are used. All parameters in the <Absolute> block must be provided. The device will pan/tilt to the provided elevation and azimuth degrees in respect to the device's "home" position. The device will also zoom to the position specified by <absoluteZoom>. The "homePosition" URI should be called first to configure the device's "home" or "zero" position. The auto patrol feature is stopped if it is running.			

Absolute PTZ Data XML Block

```
<PTZData version="1.0" xmlns="urn:psialliance-org">
   <Absolute>
      <elevation>    <!-- opt, xs:integer, -90..90 -->        </elevation>
      <azimuth>      <!-- opt, xs:integer, 0..360 -->         </azimuth>
      <absoluteZoom> <!-- opt, xs:integer, 0..100 -->         </absoluteZoom>
   </Absolute>
</PTZData>
```

7.13.8. /PSIA/PTZ/channels/<ID>/digital

URI	/PSIA/PTZ/channels/*ID*/digital	**Type**	Resource
Function	Digitally pans, tilts, and/or zooms the image.		

Methods	Query String(s)	Inbound Data	Return Result
PUT	positionX positionY digitalZoomLevel	<PTZData>	<ResponseStatus>
Notes	This function is used to digitally “pan/tilt/zoom” the device in relation to the current position. This function does not physically move the device. The XML content is returned with the <ResponseStatus> block. Either the inbound data or query string values are used. The <digitalZoomLevel> tag can be provided by itself to digitally zoom the image (a value of 0 indicates no zoom). If the <positionX> and <positionY> tags are provided, the <digitalZoomLevel> tag must be provided with a value greater than 0 (meaning the image must be zoomed). The <positionX> and <positionY> tags, if provided, must be in relation to the “un-zoomed”, currently set video resolution. The device will center on the provided coordinates. The <positionX> and <positionY> tags, if provided, must be within the boundaries of the current video resolution in respect to the zoom factor specified by the <digitalZoomLevel> tag. The auto patrol feature is stopped if it is running.		

Digital PTZ Data XML Block

```
<PTZData version="1.0" xmlns="urn:psialliance-org">
   <Digital>
      <positionX>          <!-- opt, xs:integer -->         </positionX>
      <positionY>          <!-- opt, xs:integer -->         </positionY>
      <digitalZoomLevel>   <!-- opt, xs:integer, 0..100 --> </digitalZoomLevel>
   </Digital>
</PTZData>
```

7.13.9. /PSIA/PTZ/channels/<ID>/status

URI	/PSIA/PTZ/channels/*ID*/status	Type	Resource
Function	Get current PTZ camera position information.		

Methods	Query String(s)	Inbound Data	Return Result
GET			<PTZStatus>
Notes	Currently only querying the absolute coordinates, elevation, azimuth and zoom, is supported.		

PTZStatus XML Block

```
<PTZStatus version="1.0" xmlns="urn:psialliance-org">
   <Absolute>
      <elevation>    <!-- opt, xs:integer, -90..90 -->    </elevation>
      <azimuth>      <!-- opt, xs:integer, 0..360 -->     </azimuth>
      <absoluteZoom> <!-- opt, xs:integer, 0..100 -->     </absoluteZoom>
   </Absolute>
</PTZStatus>
```

7.13.10. /PSIA/PTZ/channels/<ID>/presets

URI	/PSIA/PTZ/channels/*ID*/presets		Type	Resource
Function	Access the list of PTZ presets.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZPresetList>	
PUT		<PTZPresetList>	<ResponseStatus>	
POST		<PTZPreset>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A single preset corresponding to the current position created upon POST operation. Any currently running/paused patrols must be restarted if their presets are modified. If a patrol has no presets after this API is called, the patrol should be removed.			

```
<PTZPresetList version="1.0" xmlns="urn:psialliance-org">
   <PTZPreset/>   <!-- opt -->
</PTZPresetList>
```

7.13.11. /PSIA/PTZ/channels/<ID>/presets/<ID>

URI	/PSIA/PTZ/channels/ID/presets/*ID*		Type	Resource
Function	Get the preset for a particular PTZ channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZPreset>	
PUT		<PTZPreset>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	The <presetName> tag must be unique across the entire device. Any currently running/paused patrols with modified presets must be restarted. If a patrol has no presets after this API is called, the patrol should be removed. The <TextOverlayList> can optionally be provided. In this case, the text overlay is displayed when the device is navigated to said preset. <dateTimeFormat> specifies the date/time format for the timestamp, if included in the text overlay. Formatted according to Unix strptime() standard C library function. Ex. "%d %b %Y %H:%M:%S" could have a timestamp as "7 Dec 2008 12:33:45". Formatting support is device-dependent. Colors are expressed in RGB triplets in hexadecimal format (3 bytes) without the leading "0x".			

PTZPreset XML Block

```
<PTZPreset version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->    </id>
   <presetName>      <!-- req, xs:string -->       </presetName>
   <TextOverlayList> <!-- opt -->
      <TextOverlay>  <!-- req -->
         <id>                  <!-- req, xs:string;id -->          </id>
         <enabled>             <!-- req, xs:boolean -->           </enabled>
         <timeStampEnabled>    <!-- opt, xs:boolean -->           </timeStampEnabled>
         <dateTimeFormat>      <!-- opt, xs:string -->            </dateTimeFormat>
```

```
          <backgroundColor>     <!-- opt, xs:hexBinary;color -->     </backgroundColor>
          <fontColor>           <!-- opt, xs:hexBinary;color -->     </fontColor>
          <fontSize>            <!-- opt, xs:string, in pixels -->   </fontSize>
          <displayText>         <!-- opt, xs:string -->              </displayText>
          <horizontalAlignType>
              <!-- opt, xs:string, "left,right,center" -->
          </horizontalAlignType>
          <verticalAlignType>
              <!-- opt, xs:string, "top,bottom" -->
          </verticalAlignType>
       </TextOverlay>
   </TextOverlayList>
</PTZPreset>
```

7.13.12. /PSIA/PTZ/channels/<ID>/presets/<ID>/goto

URI	/PSIA/PTZ/channels/*ID*/presets/*ID*/goto		Type	Resource
Function	Go to a preset position on a particular PTZ channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT			<ResponseStatus>	
Notes	The auto patrol feature is stopped if it is running.			

7.13.13. /PSIA/PTZ/channels/<ID>/patrols

URI	/PSIA/PTZ/channels/*ID*/patrols		Type	Resource
Function	Access and configure PTZ patrols.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZPatrolList>	
PUT		<PTZPatrolList>	<ResponseStatus>	
POST		<PTZPatrol>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A PTZ patrol is composed of a set of presets and dwell times and runs continually in a loop.			

PTZPatrolList XML Block

```
<PTZPatrolList version="1.0" xmlns="urn:psialliance-org">
   <PTZPatrol/>   <!-- opt -->
</PTZPatrolList>
```

7.13.14. /PSIA/PTZ/channels/<ID>/patrols/status

URI	/PSIA/PTZ/channels/*ID*/patrols/status		Type	Resource
Function	Get the status of all PTZ patrols on a particular PTZ channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZPatrolStatusList>	
Notes	The status should be given for every configured patrol on the device. <patrolID> is defined in /PSIA/PTZ/channels/*ID*/patrols/*ID*.			

PTZPatrolStatus XML Block

```
<PTZPatrolStatusList version="1.0" xmlns="urn:psialliance-org">
   <PTZPatrolStatus>     <!-- opt -->
      <patrolID>         <!-- req, xs:string;id -->                          </patrolID>
      <patrolStatus>     <!-- req, xs:string, "running,stopped,paused" -->   </patrolStatus>
   </PTZPatrolStatus>
</PTZPatrolStatusList>
```

7.13.15. /PSIA/PTZ/channels/<ID>/patrols/<ID>

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*		Type	Resource
Function	Access and configure a particular PTZ patrol.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<PTZPatrol>	
PUT		<PTZPatrol>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	A "patrol" is defined as a specific sequence of presets in fixed rotation, with a specified dwell time per each preset. The <presetID> must correspond to a valid ID in /PSIA/PTZ/channels/<ID>/presets. The <SequenceList> entries are order-specific. The presets will be addressed from the top-down. The auto patrol feature is restarted if it is currently running for a patrol entry that has changed settings. The auto patrol feature is stopped if it is currently paused for a patrol entry that has changed settings. Patrol schedules should not overlap unless the device is capable of running multiple patrols at the same time (i.e. an analog-to-digital encoding device). Manual patrol APIs (startPatrol, stopPatrol, pausePatrol) override patrol scheduling.			

PTZPatrol XML Block

```
<PTZPatrol version="1.0" xmlns="urn:psialliance-org">
   <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 -->
         <presetID>      <!-- req, xs:string;id -->                      </presetID>
         <delay>         <!-- req, xs:integer, milliseconds -->          </delay>
      </PatrolSequence>
```

```
    </PatrolSequenceList>
</PTZPatrol>
```

7.13.16. /PSIA/PTZ/channels/<ID>/patrols/<ID>/start

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*/start		Type	Resource
Function	Manually start a patrol.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	A patrol is not initialized if there are less than two presets in the sequence list. The auto patrol feature is restarted if running for a particular patrol. If the auto patrol feature is paused for the particular patrol, it is resumed based on the <resumeType> tag – if <resumeType> refers to 'Relative', the patrol is resumed where it left off. If <resumeType> refers to 'Absolute', the patrol is resumed at the position where it would have been had it not been paused.			

7.13.17. /PSIA/PTZ/channels/<ID>/patrols/<ID>/stop

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*/stop		Type	Resource
Function	Manually stop a patrol.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	The specified patrol sequence is stopped if it is currently running or paused.			

7.13.18. /PSIA/PTZ/channels/<ID>/patrols/<ID>/pause

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*/pause		Type	Resource
Function	Manually pause a patrol.			
Methods	Query String(s)	Inbound Data	Return Result	
PUT			<ResponseStatus>	
Notes	A patrol can be resumed by calling /PSIA/PTZ/channels/*ID*/patrols/*ID*/start.			

7.13.19. /PSIA/PTZ/channels/<ID>/patrols/<ID>/status

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*/status		Type	Resource
Function	Query a patrol status.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<PTZPatrolStatus>	
Notes	Returns the status of a particular patrol; whether it is running, stopped or paused.			

7.13.20. /PSIA/PTZ/channels/<ID>/patrols/<ID>/schedule

URI	/PSIA/PTZ/channels/*ID*/patrols/*ID*/schedule	Type	Resource

Function	Access the schedule for a particular PTZ patrol.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<TimeBlockList>
PUT		<TimeBlockList>	<ResponseStatus>
Notes	The <TimeBlockList> in the schedule defines when the patrol should be active.		

7.13.21. Patrol Examples

Example: Create a patrol

The following commands are two examples of setting up patrols. Here is the list of PTZ channel 1 presets used for the settings.

```
GET /PSIA/PTZ/channels/1/presets 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”?>
<PTZPresetList version=“1.0” xmlns=“urn:psialliance-org”>
   <PTZPreset>
       <id>1</id>
       <presetName>Left Wing</presetName>
   </PTZPreset>
   <PTZPreset>
       <id>2</id>
       <presetName>Right Wing</presetName>
   </PTZPreset>
   <PTZPreset>
       <id>3</id>
       <presetName>Gate</presetName>
   </PTZPreset>
   <PTZPreset>
       <id>4</id>
       <presetName>Alley</presetName>
   </PTZPreset>
   <PTZPreset>
       <id>5</id>
       <presetName>North Entrance</presetName>
   </PTZPreset>
   <PTZPreset>
       <id>6</id>
       <presetName>East Entrance</presetName>
   </PTZPreset>
</PTZPresetList>
```

Use the following command to create a “Parking Garage” patrol has the behavior “Left wing” @ 5 sec, “Right wing” @ 5 sec, “Gate” @ 10 sec, “Alley” @ 3 sec, and repeat:

```
POST /PSIA/PTZ/channels/1/patrols HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<PTZPatrol version="1.0" xmlns="urn:psialliance-org">
   <patrolName>Parking Garage</patrolName>
   <resumeType>relative</resumeType>
   <PatrolSequenceList>
       <PatrolSequence>
           <presetID>1</presetID>
           <delay>5000</delay>
       </PatrolSequence>
       <PatrolSequence>
           <presetID>2</presetID>
           <delay>5000</delay>
       </PatrolSequence>
       <PatrolSequence>
           <presetID>3</presetID>
           <delay>10000</delay>
       </PatrolSequence>
       <PatrolSequence>
           <presetID>4</presetID>
           <delay>3000</delay>
       </PatrolSequence>
   </PatrolSequenceList>
</PTZPatrol>
```

Use the following command to create a "Perimeter Scan" patrol has the behavior "North entrance" @ 7 sec, "East entrance" @ 7 sec, "Alley" @ 7 sec, and repeat.

```
POST /PSIA/PTZ/channels/1/patrols HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<PTZPatrol version="1.0" xmlns="urn:psialliance-org">
   <patrolName>Perimeter Scan</patrolName>
   <resumeType>relative</resumeType>
   <PatrolSequenceList>
       <PatrolSequence>
           <presetID>5</presetID>
           <delay>7000</delay>
       </PatrolSequence>
       <PatrolSequence>
           <presetID>7</presetID>
           <delay>7000</delay>
       </PatrolSequence>
       <PatrolSequence>
           <presetID>4</presetID>
           <delay>7000</delay>
       </PatrolSequence>
   </PatrolSequenceList>
</PTZPatrol>
```

Example: Schedule a Patrol

Assume that the "Parking Garage" patrol has been assigned to ID 7. The following command schedules the patrol to operate from 9:00 am to 7:00 pm Monday, Wednesday, Friday, and 9:00 am to 11:00 pm on the weekends:

```
PUT /PSIA/PTZ/channels/1/patrols/7/schedule HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<TimeBlockList>
   <TimeBlock>
       <dayOfWeek>1</dayOfWeek>
       <TimeRange>
           <beginTime>09:00:00</beginTime>
           <endTime>19:00:00</endTime>
       </TimeRange>
   </TimeBlock>
   <TimeBlock>
       <dayOfWeek>3</dayOfWeek>
       <TimeRange>
           <beginTime>09:00:00</beginTime>
           <endTime>19:00:00</endTime>
       </TimeRange>
   </TimeBlock>
   <TimeBlock>
       <dayOfWeek>5</dayOfWeek>
       <TimeRange>
           <beginTime>09:00:00</beginTime>
               <endTime>19:00:00</endTime>
       </TimeRange>
   </TimeBlock>
<TimeBlockList>
```

Assume that the "Perimeter Scan" patrol has been assigned to ID 8. The following command schedules the patrol to operate from 11:00 pm to 6:00 am everyday:

```
PUT /PSIA/PTZ/channels/1/patrols/8/schedule HTTP/1.1
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>
<TimeBlockList>
   <TimeBlock>
       <dayOfWeek>6</dayOfWeek>
       <TimeRange>
           <beginTime>23:00:00</beginTime>
           <endTime>06:00:00</endTime>
       </TimeRange>
   </TimeBlock>
   <TimeBlock>
       <dayOfWeek>7</dayOfWeek>
       <TimeRange>
           <beginTime>23:00:00</beginTime>
           <endTime>06:00:00</endTime>
       </TimeRange>
   </TimeBlock>
</TimeBlockList>
```

7.14./PSIA/Custom/MotionDetection

URI	/PSIA/Custom/MotionDetection		Type	Service
Function	Motion detection configuration for all video input channels.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<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="urn:psialliance-org">
   <MotionDetection/>    <!-- opt -->
</MotionDetectionList>
```

7.14.1. /PSIA/Custom/MotionDetection/<ID>

URI	/PSIA/Custom/MotionDetection/*ID*		Type	Resource
Function	Motion detection configuration for a video input channel.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<MotionDetection>	
PUT		<MotionDetection>	<ResponseStatus>	
Notes	Note that the ID used here MUST correspond to the video input ID. The interface supports both grid-based and region-based motion detection. The actual types supported can be determined by looking at the result of a GET of /PSIA/Custom/MotionDetection/*ID*/capabilities and looking at the options available for the <regionType> field. Grid-based motion detect divides the image into a set of fixed "bins" that delimit the motion detection area boundaries. ROI-based motion detection allows motion areas or regions of interest to be defined based on pixel coordinates.			

MotionDetection XML Block

```
<MotionDetection version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->                  </id>
   <enabled>             <!-- req, xs:boolean -->                    </enabled>
   <samplingInterval>    <!-- req, xs:integer, number of frames -->   </samplingInterval>
   <startTriggerTime>    <!-- req, xs:integer, milliseconds -->       </startTriggerTime>
   <endTriggerTime>      <!-- req, xs:integer, milliseconds -->       </endTriggerTime>
   <directionSensitivity>
       <!-- opt, xs:string, "left-right,right-left,up-down,down-up" -->
</directionSensitivity>
   <regionType>          <!-- req, xs:string, "grid,roi" -->          </regionType>
   <minObjectSize>
       <!-- opt, xs:integer, min number of pixels per object -->
   </minObjectSize>
   <maxObjectSize>
       <!-- opt, xs:integer, max number of pixels per object -->
   </maxObjectSize>
```

```
   <Grid>            <!-- dep, required if <motionType> is "grid" -->
      <rowGranularity>      <!-- req, xs:integer -->  </rowGranularity>
      <columnGranularity>   <!-- req, xs:integer -->  </columnGranularity>
   </Grid>
   <ROI>             <!-- dep, required if <motionType> is "roi" -->
      <minHorizontalResolution> <!-- req, xs:integer -->  </minHorizontalResolution>
      <minVerticalResolution>   <!-- req, xs:integer -->  </minVerticalResolution>
   </ROI>
   <MotionDetectionRegionList/>  <!-- req -->
</MotionDetection>
```

7.14.2. /PSIA/Custom/MotionDetection/<ID>/regions

URI	/PSIA/Custom/MotionDetection/*ID*/regions		Type	Resource
Function	Access the list of regions for motion detection on a particular video input channel.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<MotionDetectionRegionList>	
PUT		<MotionDetectionRegionList>	<ResponseStatus>	
POST		<MotionDetectionRegion>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Each motion detection region has its own detection threshold and sensitivity level. It is possible to define mask regions that are subtracted from other regions, allowing non-rectangular motion areas to be configured.			

MotionDetectionRegionList XML Block

```
<MotionDetectionRegionList version="1.0" xmlns="urn:psialliance-org">
   <MotionDetectionRegion/>  <!-- opt -->
</MotionDetectionRegionList>
```

7.14.3. /PSIA/Custom/MotionDetection/<ID>regions/<ID>

URI	/PSIA/Custom/MotionDetection/*ID*/regions/*ID*		Type	Resource
Function	Access the list of regions for motion detection			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<MotionDetectionRegion>	
PUT		<MotionDetectionRegion>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	The region detection coordinate space depends on the value of <motionType>.			

MotionDetectionRegion XML Block

```
<MotionDetectionRegion version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id --></id>
   <enabled>             <!-- req, xs:boolean -->  </enabled>
   <maskEnabled>         <!-- opt, xs:boolean -->  </maskEnabled>
   <sensitivityLevel>    <!-- req -->
      <!-- req, xs:integer, 0..100, 0 is least sensitive -->
   </sensitivityLevel>
```

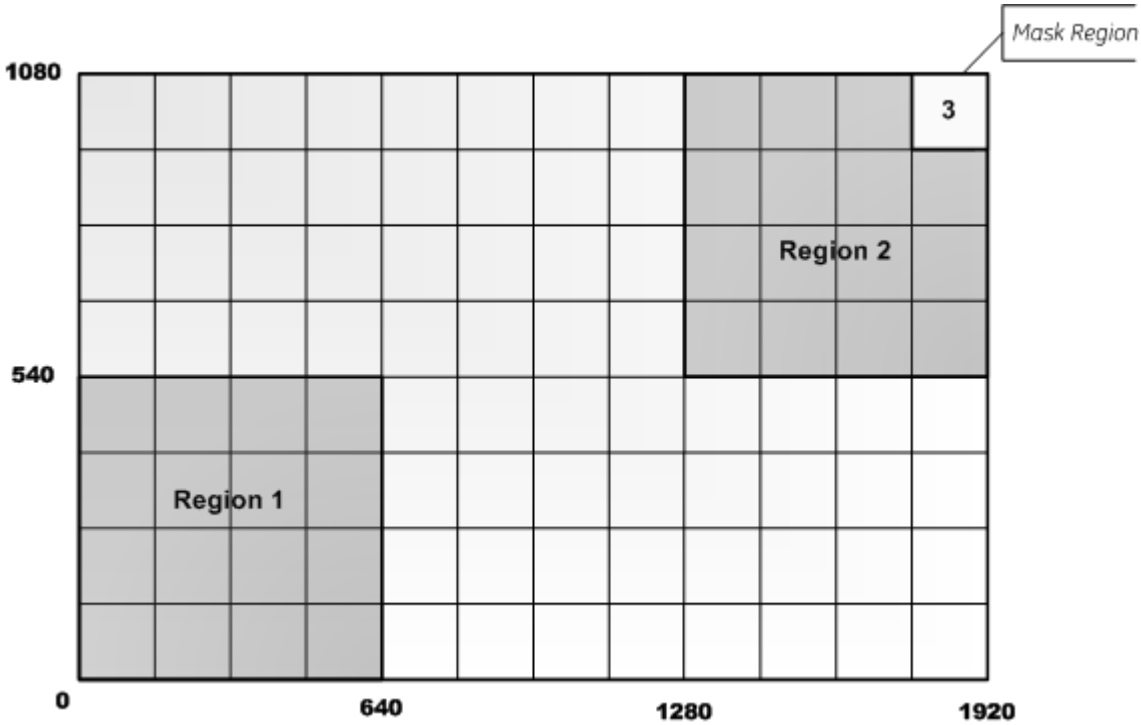
```
   <detectionThreshold>      <!-- req -->
       <!-- req, xs:integer, 0..100, percentage-->
   </detectionThreshold>
   <RegionCoordinatesList>   <!-- req -->
       <RegionCoordinates>   <!-- Note: at least two coordinates are required -->
           <positionX>       <!-- req, xs:integer -->  </positionX>
           <positionY>       <!-- req, xs:integer -->  </positionY>
       </RegionCoordinates>
   </RegionCoordinatesList>
</MotionDetectionRegion>
```

7.14.4. Motion Detection Example

Set up Motion Detection

The following command configures two rectangular detection regions, with one “masked” region on video input channel ID 777. Example assumes a resolution of 1920x1080 and a grid motion detection algorithm:

- Motion detection is enabled with a granularity of a 12x8 grid – this means the detection region coordinates will ultimately be defined by a grid of 96 regions. For a resolution of 1920x1080, this means that each “granule” will be 160x135 pixels (1920/12 x 1080/8). (If a coordinate doesn’t exactly match the configured granularity, it should be mapped internally to the nearest possible point)
- A sample will be taken every 2 frames for motion detection and motion must be detected for at least one second before triggering an event notification (motion must be stopped for at least one second to stop the triggering).
- Two detection regions are defined, the second containing an inner/overlapping region that is disabled. Region 1 occupies the bottom-left 8 granules. Region 2 occupies the top-right 8 granules, with the top-right-most corner granule (region 3) disabled by use of the <maskEnabled> tag.



```
PUT /PSIA/Custom/Analytics/motionDetection/777 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="urn:psialliance-org">
   <enabled>true</enabled>
   <samplingInterval>2</samplingInterval>
   <startTriggerTime>1000</startTriggerTime>
   <endTriggerTime>1000</endTriggerTime>
   <regionType>grid</regionType>
   <Grid>
       <rowGranularity>8</rowGranularity>
       <columnGranularity>12</columnGranularity>
   </Grid>
   <MotionDetectionRegionList>
       <MotionDetectionRegion>
           <enabled>true</enabled>
           <sensitivityLevel>50</sensitivityLevel>
           <detectionThreshold>80</detectionThreshold>
           <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>
           <enabled>true</enabled>
           <sensitivityLevel>20</sensitivityLevel>
           <detectionThreshold>50</detectionThreshold>
           <RegionCoordinatesList>
               <RegionCoordinates>
                   <positionX>8</positionX>
                   <positionY>4</positionY>
               </RegionCoordinates>
               <RegionCoordinates>
                   <positionX>8</positionX>
                   <positionY>8</positionY>
               </RegionCoordinates>
               <RegionCoordinates>
                   <positionX>12</positionX>
                   <positionY>8</positionY>
               </RegionCoordinates>
               <RegionCoordinates>
                   <positionX>12</positionX>
                   <positionY>4</positionY>
               </RegionCoordinates>
           </RegionCoordinatesList>
```

```
      </MotionDetectionRegion>
      <MotionDetectionRegion>
          <maskEnabled>true</maskEnabled>
          <RegionCoordinatesList>
              <RegionCoordinates>
                  <positionX>11</positionX>
                  <positionY>7</positionY>
              </RegionCoordinates>
              <RegionCoordinates>
                  <positionX>11</positionX>
                  <positionY>8</positionY>
              </RegionCoordinates>
              <RegionCoordinates>
                  <positionX>12</positionX>
                  <positionY>8</positionY>
              </RegionCoordinates>
              <RegionCoordinates>
                  <positionX>12</positionX>
                  <positionY>7</positionY>
              </RegionCoordinates>
          </RegionCoordinatesList>
      </MotionDetectionRegion>
   </MotionDetectionRegionList>
</MotionDetection>
```

7.15./PSIA/Custom/Event

URI	/PSIA/Custom/Event		Type	Service
Function	Access and configure the device event behavior, scheduling and notifications.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<EventNotification>	
PUT		<EventNotification>	<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 (HTTP, FTP, e-mail) are supported.			

EventNotification XML Block

```
<EventNotification version="1.0" xmlns="urn:psialliance-org">
   <EventTriggerList/>          <!-- opt -->
   <EventSchedule/>             <!-- opt -->
   <EventNotificationMethods/>  <!-- opt -->
</EventNotification>
```

7.15.1. /PSIA/Custom/Event/triggers

URI	/PSIA/Custom/Event/triggers		Type	Resource
Function	Access the list of event triggers.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<EventTriggerList>	

PUT		<EventTriggerList>	<ResponseStatus>
POST		<EventTrigger>	<ResponseStatus>
DELETE			<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="urn:psialliance-org">
   <EventTrigger/>    <!-- opt -->
</EventTriggerList>
```

7.15.2. /PSIA/Custom/Event/triggers/<ID>

URI	/PSIA/Custom/Event/triggers/*ID*	**Type**	Resource
Function	Access a particular event trigger.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<EventTrigger>
PUT		<EventTrigger>	<ResponseStatus>
DELETE			<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. Disk failure: trigger when a disk fails. Recording failure: trigger when recording fails: either there is a problem with the disk, or the storage volume is full, or the volume is corrupt. Bad video: trigger when the input video is bad. POS: trigger when a point-of-sale event is detected. Analytics: trigger on a general analytics event. Currently analytics events apart from VMD, which has its own event trigger, are not supported. Fan failure: trigger when a fan fails. Overheat: trigger when the temperate threshold of a particular sensor is exceeded. Device vendors can add additional event types and advertise these using the capabilities query on /PSIA/Custom/Event/triggers. <inputIOPortID> is only required if the <eventType> is "IO".		

EventTrigger XML Block

```
<EventTrigger version="1.0" xmlns="urn:psialliance-org">
   <id>                    <!-- req, xs:string;id -->          </id>
   <eventType>             <!-- req -->
      <!-- req, xs:string,
          "IO,VMD,videoloss,raidfailure,recordingfailure,
           badvideo,POS,analytics,fanfailure,overheat"
      -->
   </eventType>
   <eventDescription>      <!-- opt, xs:string -->            </eventDescription>
   <inputIOPortID>         <!-- dep, xs:string;id -->         </inputIOPortID>
```

```
   <intervalBetweenEvents>   <!-- opt, xs:integer, seconds -->    </intervalBetweenEvents>
   <EventTriggerNotificationList/>  <!-- opt -->
</EventTrigger>
```

7.15.3. /PSIA/Custom/Event/triggers/<ID>/notifications

URI	/PSIA/Custom/Event/triggers/*ID*/notifications		Type	Resource
Function	List of notification methods and behaviors.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<EventTriggerNotificationList>	
PUT		<EventTriggerNotificationList>	<ResponseStatus>	
POST		<EventTriggerNotification>	<ResponseStatus>	
DELETE			<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="urn:psialliance-org">
   <EventTriggerNotification/>   <!-- opt -->
</EventTriggerNotificationList>
```

7.15.4. /PSIA/Custom/Event/triggers/<ID>/notifications/<ID>

URI	/PSIA/Custom/Event/triggers/*ID*/notifications/*ID*		Type	Resource
Function	Access and configure a particular notification trigger.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<EventTriggerNotification>	
PUT		<EventTriggerNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<outputIOPortID> is only required if the <notifiocationMethod> is "IO".			

EventTriggerNotification XML Block

```
<EventTriggerNotification version="1.0" xmlns="urn:psialliance-org">
   <id>                     <!-- req, xs:string;id -->              </id>
   <notificationMethod>
       <!-- req, xs:string, "email,IM,IO,syslog,HTTP,FTP" -->
   </notificationMethod>
   <notificationRecurrence>
       <!-- opt, xs:string, "beginning,beginningandend,recurring" -->
   </notificationRecurrence>
   <notificationInterval>    <!-- dep, xs:integer, milliseconds -->   </notificationInterval>
   <outputIOPortID>          <!-- dep, xs:string;id -->               </outputIOPortID>
</EventTriggerNotification>
```

7.15.5. /PSIA/Custom/Event/schedule

URI	/PSIA/Custom/Event/schedule		Type	Resource
Function	Event schedules.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<EventSchedule>	
PUT		<EventSchedule>	<ResponseStatus>	
Notes	Defines the schedule. The schedule is defined as a date-time range and a set of time blocks that define when the events are active. If <DateTimeRange> is not present, the schedule is always valid.			

EventSchedule XML Block

```
<EventSchedule version="1.0" xmlns="urn:psialliance-org">
   <DateTimeRange>       <!-- opt -->
      <beginDateTime>    <!-- req, xs:datetime -->     </beginDateTime>
      <endDateTime>      <!-- req, xs:datetime -->     </endDateTime>
   </DateTimeRange>
   <TimeBlockList/>      <!-- req -->
</EventSchedule>
```

7.15.6. /PSIA/Custom/Event/notification

URI	/PSIA/Custom/Event/notification		Type	Resource
Function	Configure notifications.			
Methods	Query String(s)	Inbound Data	Return Result	
GET			<EventNotificationMethods>	
PUT		<EventNotificationMethods>	<ResponseStatus>	
Notes	The following notification types are supported: HTTP: the device connects to a given address and port and issues an HTTP GET/POST with the given parameters. FTP: a video clip or snapshot is uploaded to an FTP server. E-mail: a mail with the video clip or snapshot is sent in an e-mail to a list of servers. <MediaFormat> determines the type of snapshot, video clip and the video clip pre and post recording times.			

EventNotificationMethods XML Block

```
<EventNotificationMethods version="1.0" xmlns="urn:psialliance-org">
   <MailingNotificationList/>    <!-- opt -->
   <FTPNotificationList/>        <!-- opt -->
   <HttpHostNotificationList/>   <!-- opt -->
   <FTPFormat>                   <!-- opt -->
      <uploadSnapShotEnabled>    <!-- req, xs:boolean -->  </uploadSnapShotEnabled>
      <uploadVideoClipEnabled>   <!-- req, xs:boolean -->  </uploadVideoClipEnabled>
   </FTPFormat>
   <EmailFormat>                 <!-- opt -->
      <senderEmailAddress>       <!-- req, xs:string -->   </senderEmailAddress>
      <receiverEmailAddress>     <!-- req, xs:string -->   </receiverEmailAddress>
      <subject>                  <!-- req, xs:string -->   </subject>
      <BodySetting>                 <!-- opt -->
```

```
          <attachedVideoURLEnabled> <!-- req, xs:boolean -->  </attachedVideoURLEnabled>
          <attachedSnapShotEnabled> <!-- req, xs:boolean -->  </attachedSnapShotEnabled>
          <attachedVideoClipEnabled><!-- req, xs:boolean -->  </attachedVideoClipEnabled>
       </BodySetting>
   </EmailFormat>
   <MediaFormat>             <!-- opt -->
       <snapShotImageType>   <!-- opt, xs:string, "JPEG,GIF,PNG" -->  </snapShotImageType>
       <videoClipFormatType> <!-- opt, xs:string, "ASF,MP4,3GP,264" --></videoClipFormatType>
       <preCaptureLength>    <!-- opt, xs:integer, milliseconds -->   </preCaptureLength>
       <postCaptureLength>   <!-- opt, xs:integer, milliseconds -->   </postCaptureLength>
   </MediaFormat>
</EvenNotificationMethods>
```

7.15.7. /PSIA/Custom/Event/notification/mailing

URI	/PSIA/Custom/Event/notification/mailing		Type	Resource
Function	E-mail notifications.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<MailingNotificationList>	
PUT		<MailingNotificationList>	<ResponseStatus>	
POST		<MailingNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	When the notification is triggered, an e-mail with a snapshot or video clip is mailed to the each of the addresses in the mailing list.			

MailingNotificationList XML Block

```
<MailingNotificationList version="1.0" xmlns="urn:psialliance-org">
   <MailingNotification/>    <!-- opt -->
</MailingNotificationList>
```

7.15.8. /PSIA/Custom/Event/notification/mailing/<ID>

URI	/PSIA/Custom/Event/notification/mailing/*ID*		Type	Resource
Function	Access a particular e-mail notification.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<MailingNotification>	
PUT		<MailingNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server. <authenticationMode> determines the authentication requirements for sending an email from the device. <portNo> is the port number of the SMTP server entry. <popAddressingFormatType> indicates whether an IP address or hostname is used for the POP server. <accountName> is the user account name for the SMTP server			

MailingNotification XML Block

```
<MailingNotification version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->    </id>
   <authenticationMode>
       <!-- req, xs:string, "none,SMTP,POP/SMTP" -->
   </authenticationMode>
   <addressingFormatType>
       <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>            <!-- dep, xs:string -->       </hostName>
   <ipAddress>           <!-- dep, xs:string -->       </ipAddress>
   <ipv6Address>         <!-- dep, xs:string -->       </ipv6Address>
   <portNo>              <!-- opt, xs:integer -->      </portNo>
   <popAddressingFormatType>
       <!-- xs:string, "ipaddress,hostname" -->
   </popAddressingFormatType>
   <popServerHostName>   <!-- dep, xs:string -->   </popServerHostName>
   <popServerIPAddress>  <!-- dep, xs:string -->   </popServerIPAddress>
   <popServerIPv6Address><!-- dep, xs:string -->   </popServerIPv6Address>
   <accountName>         <!-- dep, xs:string -->   </accountName>
   <password>            <!-- dep, xs:string -->   </password>
</MailingNotification>
```

7.15.9. /PSIA/Custom/Event/notification/ftp

URI	/PSIA/Custom/Event/notification/ftp		Type	Resource
Function	FTP notifications.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<FTPNotificationList>	
PUT		<FTPNotificationList>	<ResponseStatus>	
POST		<FTPNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	FTP notifications involve posting a particular video clip or snapshot to an FTP server.			

FTPNotificationList XML Block

```
<FTPNotificationList version="1.0" xmlns="urn:psialliance-org">
   <FTPNotification/>    <!-- opt -->
</FTPNotificationList>
```

7.15.10. /PSIA/Custom/Event/notification/ftp/<ID>

URI	/PSIA/Custom/Event/notification/ftp/*ID*		Type	Resource
Function	Access a particular FTP transfer notification.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<FTPNotification>	
PUT		<FTPNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes	Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server. Note: FTP transfers are always in binary mode.

FTPNotification XML Block

```
<FTPNotification version="1.0" xmlns="urn:psialliance-org">
   <id>                  <!-- req, xs:string;id -->       </id>
   <addressingFormatType>
       <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>            <!-- dep, xs:string -->          </hostName>
   <ipAddress>           <!-- dep, xs:string -->          </ipAddress>
   <ipv6Address>         <!-- dep, xs:string -->          </ipv6Address>
   <portNo>              <!-- opt, xs:integer -->         </portNo>
   <userName>            <!-- req, xs:string -->          </userName>
   <password>            <!-- req, xs:string -->          </password>
   <passiveModeEnabled>  <!-- opt, xs:boolean -->         </passiveModeEnabled>
   <uploadPath>          <!-- opt, xs:string -->          </uploadPath>
   <baseFileName>        <!-- opt, xs:string -->          </baseFileName>
</FTPNotification>
```

7.15.11. /PSIA/Custom/Event/notification/httpHost

URI	/PSIA/Custom/Event/notification/httpHost		**Type**	Resource
Function	Access the list of HTTP notification hosts.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<HttpHostNotificationList>	
PUT		<HttpHostNotificationList>	<ResponseStatus>	
POST		<HttpHostNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	HTTP notification involves the device connecting to a particular URL and delivering an HTTP message whenever the event triggers.			

HttpHostNotificationList XML Block

```
<HttpHostNotificationList version="1.0" xmlns="urn:psialliance-org">
   <HttpHostNotification/>   <!-- opt -->
</HttpHostNotificationList>
```

7.15.12. /PSIA/Custom/Event/notification/httpHost/<ID>

URI	/PSIA/Custom/Event/notification/httpHost/*ID*		Type	Resource
Function	Access a particular HTTP notification host.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<HttpHostNotification>	
PUT		<HttpHostNotification>	<ResponseStatus>	
DELETE			<ResponseStatus>	

Notes	Depending on the value of <addressingFormatType>, either the <hostName> or the IP address fields will be used to locate the NTP server. If <parameterFormatType> is “XML”, HTTP POST is used. If <parameterFormatType> is “querystring”, HTTP GET is used.

HttpHostNotification XML Block

```
<HttpHostNotification version="1.0" xmlns="urn:psialliance-org">
   <id>              <!-- req, xs:string;id -->              </id>
   <url>             <!-- req, xs:string -->                 </url>
   <protocolType>    <!-- req, xs:string, "HTTP,HTTPS" -->   </protocolType>
   <parameterFormatType>
       <!-- req, xs:string, "XML,querystring" -->
   </parameterFormatType>
   <addressingFormatType>
       <!-- req, xs:string, "ipaddress,hostname" -->
   </addressingFormatType>
   <hostName>        <!-- dep, xs:string -->                 </hostName>
   <ipAddress>       <!-- dep, xs:string -->                 </ipAddress>
   <ipv6Address>     <!-- dep, xs:string -->                 </ipv6Address>
   <portNo>          <!-- opt, xs:integer -->                </portNo>
   <userName>        <!-- dep, xs:string -->                 </userName>
   <password>        <!-- dep, xs:string -->                 </password>
   <httpAuthenticationMethod>
       <!-- req, xs:string, "MD5digest,none" -->
   </httpAuthenticationMethod>
</HttpHostNotification>
```

7.15.13. /PSIA/Custom/Event/notification/alertStream

URI	/PSIA/Custom/Event/notification/alertStream		**Type**	Resource
Function	Access the event notification data stream through HTTP server push.			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			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="urn:psialliance-org">
   <ipAddress>       <!-- dep, xs:string -->       </ipAddress>
   <ipv6Address>     <!-- dep, xs:string -->       </ipv6Address>
   <portNo>          <!-- opt, xs:integer -->      </portNo>
```

```
   <protocol>        <!-- opt, xs:string, "HTTP,HTTPS" -->    </protocol>
   <macAddress>      <!-- opt, xs:string;MAC -->   </macAddress>
   <channelID>       <!-- dep, xs:string;id -->    </channelID>
   <dateTime>        <!-- req, xs:datetime -->     </dateTime>
   <activePostCount> <!-- req, xs:integer -->      </activePostCount>
   <eventType>
       <!-- req, xs:string,
           "IO,VMD,videoloss,raidfailure,recordingfailure,
            badvideo,POS,analytics,fanfailure,overheat"
       -->
   </eventType>
   <eventState>      <!-- req, xs:string, "active,inactive" -->   </eventState>
   <eventDescription><!-- opt, xs:string -->                      </eventDescription>
   <inputIOPortID>   <!-- dep, xs:string;id -->                   </inputIOPortID>
   <DetectionRegionList>         <!-- dep, if <eventType> is "vmd" -->
       <DetectionRegionEntry>    <!-- req -->
           <regionID>            <!-- req, xs:string;id -->       </regionID>
           <sensitivityLevel>    <!-- req, xs:integer, 0..100 --> </sensitivityLevel>
           <detectionThreshold>  <!-- req, xs:integer, 0..100 --> </detectionThreshold>
       </DetectionRegionEntry>
   </DetectionRegionList>
</EventNotificationAlert>
```

Example

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

```
GET /PSIA/Custom/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="urn:psialliance-org">
   <ipAddress>3.137.217.220</ipAddress>
   <portNo>80</portNo>
   <protocol>HTTP</protocol>
   <macAddress>00:14:22:43:D5:D4</macAddress>
   <channelID>1</channelID>
   <dateTime>2009-03-11T15:27Z</dateTime>
   <activePostCount>1</activePostCount>
   <eventType>VMD</eventType>
   <eventState>active</eventState>
   <eventDescription>Motion alarm</eventDescription>
   <DetectionRegionList>
       <DetectionRegionEntry>
           <regionID>2</regionID>
           <sensitivityLevel>67</sensitivityLevel>
           <detectionThreshold>43</detectionThreshold>
           <detectionLevel>49</detectionLevel>
       </DetectionRegionEntry>
   </DetectionRegionList>
</EventNotificationAlert>
--<boundary>
…
```

7.15.14. HTTP Notification Alert

This function is used to send an event notification from the media device to the monitoring web server or management system via HTTP or HTTPS. A connection will be established with the client only when an event occurs. The destination address is determined by the <HttpHostNotificationList> block.

URI	`https://<ipAddress>:<portNo>/<url>`		
Function	HTTP notification alert request.		
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
POST		Notification Alert	
Notes	Either GET or POST can be used. If GET is used, the corresponding query string parameters are provided in place of the inbound XML. If Post is used, the inbound XML is provided in place of the corresponding query string parameters. The "DeviceID=" and "DeviceName=" fields are taken from the <DeviceInfo> settings for the device. The <parameterFormatType> tag indicates whether XML or query string parameters should be used for this API. The <protocolType> tag under <HttpHostList> determines whether HTTP or HTTPS is used for this API. The <portNo> tag under <HttpHostList> determines the port number to be used for the notification alert. The <portNo> and <protocolType> tags in the alert are provided for a client application to connect/manage the device after it sends out this notification. The <addressingFormatType> tag under <HttpHostList> determines whether <ipAddress>/IPAddress or <ipv6Address>/IPv6Address is used. The <url> tag under <HttpHostList> indicates the URL base to be used for the alert. If <eventType>/EventType refers to an input-port-related event, the <inputIOPortID> tag or InputIOPortID parameter must be provided. If <eventType>/EventType refers to a motion-related event, the <DetectionRegionList> block or RegionIndexX parameter(s) must be provided if detection regions have been defined. If the motion event is for a full-screen configuration, these region indexes should not be provided. The <sensitivityLevel>/SensitivityLevelX and <detectionThreshold>/DetectionThresholdX parameters are used to indicate the current values of the activity detection at the time that the notification is sent out. If the alert is for a motion-related event, multiple region indexes may be provided per single API. If query string parameters are used, the format "RegionIndexX" is used where "X" is a number starting with "1" and incrementing by one for every subsequent region index provided. If the <httpAuthenticationMethod> tag under <HttpHostList> is configured for "MD5 Digest Authentication", the corresponding security values must be stored in the header fields of the HTTP(S) request. The <activePostCount>/ActivePostCount parameter is a sequence number starting at 1 and incrementing by one for every event notification sent.		

Notification Alert

```
version=1.0
DeviceID=
DeviceName=
IPAddress=
IPv6Address=
PortNo=
Protocol=
MacAddress=
```

```
ChannelID=
DateTime=
ActivePostCount=
EventType=
EventState=
EventDescription=
InputIOPortID=
RegionIndex1=
SensitivityLevel1=
DetectionThreshold1=
RegionIndex2=
SensitivityLevel2=
DetectionThreshold2=
…
```

7.15.15. E-mail Notification Alert

Function	Send e-mail on alert
Notes	1. The <MailingList> XML block determines how the email is sent. 2. The “From” email address is determined by the <senderEmailAddress> tag in the <EmailFormat> block. 3. The “To” email address is determined by the <receiverEmailAddress> tag in the <EmailFormat> block. 4. The “Subject” of the email is determined by the <subject> tag in the <EmailFormat> block. 5. The <EventNotificationAlert> XML follows the same rules as the “HTTPS Event Notification Alert” API. 6. The <BodySetting> block in the <EmailFormat> block determines what media (if any) is included in the email. 7. If a video/audio clip is attached to the email body, the <preCaptureLength> and <postCaptureLength> tags in <EventNotificationSetting> will determine the length of the clip.

E-mail Format

```
From: …
To: …
Subject: …

<?xml version=“1.0” encoding=“UTF-8”?>
<EventNotificationAlert version="1.0" xmlns="urn:psialliance-org">
   …
</EventNotificationAlert>

VideoURL=…

(Picture Snap Shot)
```

7.15.16. 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 HTTP server will be notified with the “HTTPS Event Notification Alert”. The behavior of this notification is as follows:

- An HTTP(S) notification is sent at detection time, and every 5 seconds after while the signal is present. This is denoted by the <notificationRecurrence> and <notificationInterval> tags. These APIs will have an <eventState> of “active”.
- When the input port 1 signal detection stops, one last HTTP(S) notification is sent to the server (again, 5 seconds from the last notification) with an <eventState> of “active”.
- After the signal detection stops for input port 1, the device will wait 1 second before starting to detect the signal again for this port (indicated by <intervalBetweenEvents>).

```
POST /PSIA/Custom/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=“urn:psialliance-org”>
   <eventType>IO</eventType>
   <eventDescription>Input port 1 event detection</eventDescription>
   <inputIOPortID>111</inputIOPortID>
   <intervalBetweenEvents>1</intervalBetweenEvents>
   <EventTriggerNotificationList>
       <EventTriggerNotification>
           <notificationMethod>IO</notificationMethod>
           <outputIOPortID>222</outputIOPortID>
       </EventTriggerNotification>
       <EventTriggerNotification>
           <notificationMethod>HTTP</notificationMethod>
           <notificationRecurrence>recurring</notificationRecurrence>
           <notificationInterval>5000</notificationInterval>
       </EventTriggerNotification>
   </EventTriggerNotificationList>
</EventTrigger>
```

Example: Trigger Syslog from Motion Detection

The command below enables motion detection. When motion is detected, syslog notification is used. The behavior of this notification is as follows:

- A syslog message is sent once at detection time
- A syslog message is sent once when detection stops

The above behavior is result of the <notificationRecurrence> tag. When detection stops the device will immediately start motion detection again, as denoted by the <intervalBetweenEvents> tag.

```
POST /PSIA/Custom/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=“urn:psialliance-org”>
   <eventType>VMD</eventType>
   <eventDescription>Motion detection</eventDescription>
   <intervalBetweenEvents>0</intervalBetweenEvents>
   <EventTriggerNotificationList>
       <EventTriggerNotification>
           <notificationMethod>syslog</notificationMethod>
           <notificationRecurrence>beginningandend</notificationRecurrence>
       </EventTriggerNotification>
   </EventTriggerNotificationList>
</EventTrigger>
```

Example: Schedule event detection and triggering

The command below schedules event detection and triggering from 8:00 am to 6:00 pm and 10:00 pm to 11:00 pm every Monday, Wednesday, and Friday. On Tuesday and Thursday, event detection and triggering is scheduled from 7:00 am to 5:00 pm.

```
PUT /PSIA/Custom/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="urn:psialliance-org">
   <TimeBlockList>
       <TimeBlock>
           <dayOfWeek>1</dayOfWeek>
           <bitString>000000001111111111000010</bitString>
       </TimeBlock>
       <TimeBlock>
           <dayOfWeek>2</dayOfWeek>
           <TimeRange>
               <beginTime>07:00:00</beginTime>
               <endTime>17:00:00</endTime>
           </TimeRange>
       </TimeBlock>
       <TimeBlock>
           <dayOfWeek>3</dayOfWeek>
           <bitString>000000001111111111000010</bitString>
       </TimeBlock>
       <TimeBlock>
           <dayOfWeek>4</dayOfWeek>
           <TimeRange>
               <beginTime>07:00:00</beginTime>
               <endTime>17:00:00</endTime>
           </TimeRange>
       </TimeBlock>
       <TimeBlock>
           <dayOfWeek>5</dayOfWeek>
           <TimeRange>
               <beginTime>08:00:00</beginTime>
               <endTime>18:00:00</endTime>
           </TimeRange>
       </TimeBlock>
       <TimeBlock>
           <dayOfWeek>5</dayOfWeek>
           <TimeRange>
               <beginTime>22:00:00</beginTime>
               <endTime>23:00:00</endTime>
           </TimeRange>
       </TimeBlock>
   </TimeBlockList>
</EventSchedule>
```