BS EN 62676-2-2:2014

BSI Standards Publication

Video surveillance systems for use in security applications

Part 2-2: Video transmission protocols — IP interoperability implementation based on HTTP and REST services

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62676-2-2:2014. It is identical to IEC 62676-2-2:2013.

The UK participation in its preparation was entrusted by Technical Committee GW/1, Electronic security systems, to Subcommittee GW/1/10, Closed circuit television (CCTV).

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 79913 6
ICS 13.320

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 May 2014.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62676-2-2

January 2014

ICS 13.320

English version

Video surveillance systems for use in security applications - Part 2-2: Video transmission protocols - IP interoperability implementation based on HTTP and REST services (IEC 62676-2-2:2013)

Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité -
Partie 2-2: Protocoles de transmission vidéo -
Mise en oeuvre de l'interopérabilité IP en fonction des services HTPP et REST
(CEI 62676-2-2:2013)

Videoüberwachungsanlagen für Sicherungsanwendungen -
Teil 2-2: Videoübertragungsprotokolle -
IP-Interoperabilität auf Basis von HTTP- und REST-Diensten
(IEC 62676-2-2:2013)

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B - 1000 Brussels

Foreword

The text of document 79/436/FDIS, future edition 1 of IEC 62676-2-2, prepared by IEC TC 79 "Alarm and electronic security systems" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62676-2-2:2014.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2014-09-12
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2016-12-12

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 62676-2-2:2013 was approved by CENELEC as a European Standard without any modification.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
ISO 10918-1	-	Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines	-	-
ISO/IEC 11172-3	1993	Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 3: Audio	-	-
ISO/IEC 13818-2	-	Information technology - Generic coding of moving pictures and associated audio information - Part 2: Video	-	-
ISO/IEC 14496-2	2004	Information Technology – Coding of audio-visual objects - Part 2: Visual	-	-
ISO/IEC 14496-3	-	Information technology - Coding of audio-visual objects - Part 3: Audio	-	-
ISO/IEC 14496-10	2012	Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding	-	-
IETF RFC 1213	-	Management Information Base for Network Management of TCP/IP-based Internets: MIB-II	-	-
IETF RFC 1945	-	Hypertext Transfer Protocol – HTTP/1.0, T. Berners-Lee, MIT/LCS, R. Fielding, UC Irvine, H. Frystyk	-	-
IETF RFC 2046	-	Multipurpose Internet Mail Extensions (MIME) - Part 2: Media Types	-	-
IETF RFC 2250	-	RTP Payload Format for MPEG1/MPEG2 Video	-	-
IETF RFC 2326	-	Real time Streaming protocol (RTSP)	-	-
IETF RFC 2435	-	RTP Payload Format for JPEG-compressed Video	-	-
IETF RFC 2616	-	Hypertext Transfer Protocol HTTP/1.1.	-	-
IETF RFC 2617	-	HTTP Authentication: Basic and Digest Access Authentication	-	-

Publication	Year	Title	EN/HD	Year
IETF RFC 2818	-	HTTP Over TLS	-	-
IETF RFC 3016	-	RTP Payload Format for MPEG-4 Audio/Visual Streams	-	-
IETF RFC 3550	-	A Transport Protocol for Real-Time Applications	-	-
IETF RFC 3551	-	RTP Profile for Audio and Video Conferences with Minimal Control	-	-
IETF RFC 3629	-	UTF-8, a transformation format of ISO 10646	-	-
IETF RFC 3640	-	RTP Payload Format for Transport of MPEG-4 Elementary Streams	-	-
IETF RFC 3984	-	RTP Payload Format for H.264 Video	-	-
IETF RFC 4566	-	SDP: Session Description Protocol	-	-
ITU-T Recommendation G.726	-	40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)	-	-
ITU-T Recommendation H.264	-	Advanced video coding for generic audiovisual services	-	-
ITU-T Recommendation T.81	-	Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines	-	-

CONTENTS

INTRODUCTION 6
1 Scope 7
2 Normative references 7
3 Abbreviations 8
4 Overview 10
5 Design considerations 10
5.1 General 10
5.2 REST overview 11
5.3 Conformance 11
5.3.1 General 11
5.3.2 Minimum API set 11
5.3.3 XML requirements 11
5.3.4 Protocol requirements 12
5.4 HTTP methods and REST 12
5.5 HTTP status codes and REST 12
5.6 Unique identifiers 14
5.7 ID encoding 14
6 Architecture and namespace 15
7 System flow 17
7.1 General 17
7.2 Service discovery 18
7.3 Persistent connections 18
7.4 Authentication 19
7.5 Access restrictions 19
7.6 Setting configurations 20
7.7 Getting configurations 20
7.8 Getting capabilities 21
7.9 Uploading data 22
7.10 Receiving data 22
7.11 Operations 22
7.12 Diagnostics 23
7.13 Response status 23
7.13.1 General 23
7.13.2 Status code 23
7.13.3 Status string 24
7.13.4 ID 24
7.14 Processing rules 24
8 XML modeling 24
8.1 File format 24
8.2 Data structures 24
8.3 Lists 24
8.4 Capabilities 24
9 Custom services and resources 26
10 Interface design 26
10.1 General 26

10.2 Protocol 26
10.3 Hostname 27
10.4 Port 27
10.5 URI 27
10.6 Query string 27
10.7 Resource description 27
11 Standard resource descriptions 28
11.1 General 28
11.2 index 28
11.3 indexr 28
11.4 description 29
11.5 capabilities 29
11.6 Schemas 29
11.6.1 General 29
11.6.2 ResourceDescription 30
11.6.3 ResourceList 30
11.6.4 QueryStringParameterList 30
11.6.5 responseStatus 30
11.6.6 service.xsd 31
Annex A (normative) IP Media Device API Specification Version 1.0 34
Bibliography 122

Figure 1 – PSIA service architecture example 15
Figure A.1 – Motion detection grid with two detection regions 108

Table 1 – HTTP methods 12
Table 2 – HTTP status codes and REST 13
Table 3 – Resource names 16
Table 4 – Service URLs 16
Table 5 – HTTP requests 23
Table 6 – Capability attributes 25

INTRODUCTION

The IEC Technical Committee 79 in charge of alarm and electronic security systems together with many governmental organisations, test houses and equipment manufacturers have defined a common framework for video surveillance transmission in order to achieve interoperability between products.

The IEC 62676 series of standards on video surveillance system is divided into 4 independent parts:

Part 1	System requirements
Part 2:	Video transmission protocols
Part 3:	Analog and digital video interfaces
Part 4 :	Application guidelines (to be published)

Each part has its own clauses on scope, references, definitions and requirements

This IEC 62676-2 series consists of 3 subparts, numbered parts 2-1, 2-2 and 2-3 respectively:

IEC 62676-2-1, *Video transmission protocols – General requirements*

IEC 62676-2-2, *Video transmission protocols – IP interoperability implementation based on HTTP and REST services*

IEC 62676-2-3, *Video transmission protocols – IP interoperability implementation based on Web services*

This second subpart of this IEC 62676-2 series covers IP interoperability implementation based on HTTP and REST services. It is based on the requirements for IP video transmission protocols covered in IEC 62676-2-1, which defines protocol requirements to be fulfilled by any high-level IP video device interface.

VIDEO SURVEILLANCE SYSTEMS FOR USE IN SECURITY APPLICATIONS –

Part 2-2: Video transmission protocols – IP interoperability implementation based on HTTP and REST services

1 Scope

This part of IEC 62676 specifies a compliant IP video protocol based on HTTP and REST services.

Video transmission devices are often equipped with web servers that respond to HTTP requests. The HTTP response may contain XML content (for GET actions), XML response information (for SET actions), or various text/binary content (for retrieval of configuration data, etc.). REST is an approach to creating services that expose all information as resources in a uniform way. The ease of using REST is its uniform interface for operations. Since everything is represented as a resource, create, retrieve, update, and delete (CRUD) operations use the same URI. This specification leverages the features of HTTP and REST for IP video transmission.

A video transmission device supporting compliance to the requirements of this standard based on HTTP and REST Services as described in this document is declared as compatible to ´IEC 62676-2 HTTP and REST interoperability.´

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10918-1, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*

ISO/IEC 13818-2, *Information technology – Generic coding of moving pictures and associated audio information: Video*

ISO/IEC 14496-2:2004, *Information technology – Coding of audio-visual objects – Part 2: Visual*

ISO/IEC 14496-3, *Information technology – Coding of audio-visual objects – Part 3: Audio*

ISO/IEC 14496-10:2012, *Information technology – Coding of audio-visual objects – Part 10: Advanced video coding*

IETF RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*

IETF RFC 1945, *Hypertext Transfer Protocol – HTTP/1.0*

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*

IETF RFC 2250*, Format de charge utile RTP pour la video MPEG1/MPEG2*

IETF RFC 2326, *Real Time Streaming Protocol (RTSP)*

IETF RFC 2435*, Format de charge utile RTP pour l video JPEG*

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*

IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*

IETF RFC 2818, *HTTP Over TLS*

IETF RFC 3016*, Format de charge utile RTP pour flux audio/video MPEG-4*

IETF RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*

IETF RFC 3551, *RTP Profile for Audio and Video Conferences with Minimal Control*

IETF RFC 3629*, UTF-8 un format de transformation de l'ISO 10646*

IETF RFC 3640*, Format de charge utile RTP pour le transport de flux élémentaires MPEG-4*

IETF RFC 3984*, Format de charge utile RTP pour video H.264*

IETF RFC 4566, *SDP: Session Description Protocol*

ITU-T Recommendation G.726, *40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*

ITU-T Recommendation H.264, *Advanced video coding for generic audiovisual services*

ITU-T Recommendation T.81, *Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines*

3 Abbreviations

For the purposes of this document, the following abbreviations apply.

AAC	Advanced Audio Coding
API	Application Program Interface
AVP	Audio/Video Profile
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IETF	Internet Engineering Task Force
IO	I/O Input/Output
IP	Internet Protocol

IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISO	International Standards Organization
ITU	International telecommunications Union
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
MPEG	Moving Pictures Experts Group
NTP	Network Time Protocol
NVS	Network Video Storage Device
POSIX	Portable Operating System Interface
PTZ	Pan / Tilt / Zoom
QoS	Quality of Service
REST	Representational State Transfer
RFC	(Request for comment) IETF Standards Draft
RTCP	Real Time Control Protocol.
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
SRTP	Secure Real-time Transport Protocol
SSID	Service Set ID
SSL	Secure Sockets Layer SAML Security Assertion Markup Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TTL	Time-to-live
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF	Unicode Transformation Format
UTF-8	8-bit Unicode Transformation Format URN Uniform Resource Name
UUID	Universally Unique Identifier
VMS	Video management system
VT	Video Transmission
VTD	Video Transmission device
W3C	World Wide Web Consortium
WPA	Wi-Fi Protected Access
XML	eXtensible Markup Language
Zeroconf	Zero Configuration Networking

4 Overview

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

The REST Service Model Version 1.1 is intended to assist the IEC working groups in creating new protocols or converting contributed protocols to a standard service model that will be common to all endorsed specifications. Adherence to this service model will ensure interoperability between compliant protocols.

This model is similar in nature to Web services but is geared towards lightweight computing requirements on devices. As such, these protocols will not use Simple Object Access Protocol (SOAP) as defined by the W3C-defined Web services but instead will use a simplified XML schema and/or xml schema documents (.xsd’s).

Unless otherwise noted, all specifications of this clause should treat all configuration and management aspects as resources utilizing the REpresentational State Transfer (REST) architecture.

The Service Model is based on a REST architecture. While REST specifies that all interfaces are defined as resources, in the Model of this standard these resources are grouped by service. This architecture provides a convenient way to group related resources within a hierarchical namespace and lends itself to service discovery and future expansion.

Anybody is welcome to add services at any time provided said services adhere to the service model as defined herein. Every effort should be taken to maintain full backward compatibility when adding new services. The Service Model is designed to support expansion with backwards compatibility.

5 Design considerations

5.1 General

Network-attached devices are often equipped with a web server to maintain various web pages. These pages allow the devices to be configured through an internet browser. It is natural to reuse this web server and the HTTP protocol in order for external applications to configure and control the device. Thus, all resources will use a standard HTTP request which will be processed by the device’s web server.

When possible, IP devices should implement HTTPS to support privacy of data. It is assumed that the network infrastructure is configured properly with firewall, 802.1x, etc. and other features to provide basic network level security. Additionally, because IP devices are typically endpoint devices, HTTPS is assumed to provide sufficient safeguard in combination with the other features mentioned above.

Some devices are not capable of implementing HTTPS and in certain deployments it may not be necessary (i.e. closed networks). Additionally, SSL/TLS implies certificate management on an endpoint which could pose other problems. Embedded devices may not have a “trusted” certificate without a client explicitly trusting the certificate or uploading a trusted certificate. Furthermore, certificates may need to be regenerated upon configuration changes (IP address, etc.).

As such, the protocols use the HTTP Get and Post methods as described in “Hypertext Transfer Protocol -- HTTP/1.0” (RFC 1945) and “Hypertext Transfer Protocol -- HTTP/1.1” (RFC 2616).

5.2 REST overview

REST is an approach to creating services that expose all information as resources in a uniform way. This approach is quite different from the traditional Remote Procedure Call (RPC) mechanism which identifies the functions that an application can call. Put simply, a REST Web application is noun-driven while an RPC Web application is verb-driven. For example, if a Web application were to define an RPC API for user management, it might be written as follows:

```
GET http://webserver/getUserList
GET http://webserver/getUser?userid=100
POST http://webserver/addUser
POST http://webserver/updateUser
GET http://webserver/deleteUser?userid=100
```

On the other hand, a REST API for the same operations would appear as follows:

```
GET http://webserver/users
GET http://webserver/users/user100
POST http://webserver/users
PUT http://webserver/users/user100
DELETE http://webserver/users/user100
```

Part of the simplicity of REST is its uniform interface for operations. Since everything is represented as a resource, create, retrieve, update, and delete (CRUD) operations use the same URI.

5.3 Conformance

5.3.1 General

Every protocol will define one or more compliant REST services. To ensure interoperability, the following conformance requirements are also implied in each protocol.

5.3.2 Minimum API set

In addition to the service specific mandatory requirements, a system/device shall support all of the mandatory services.

Each specification may define one or more compliant services. Each service and each contained resource shall be assigned a scope of either mandatory or optional. Within each implemented service type, all mandatory resources shall be implemented.

5.3.3 XML requirements

A system/device shall support the syntax as defined by the W3C XML 1.0 specification.

A system/device shall support the UTF-8 character set as described by http://www.w3.org/International/O-charset

Additionally, XML content shall correspond to the following Schemas as defined in Annex A:

- “ResourceList XML Schema”
- “ResourceDescription XML Schema”
- “QueryStringParameterList XML Schema”
- “ResponseStatus XML Schema”

Vendors may optionally extend this standard to include proprietary XML content as long as it does not conflict with the minimum set of APIs. In this case, it is recommended to use a vendor-specific XML namespace to avoid conflicting names that may arise with future revisions.

For example, if vendor XYZ123 Inc intends to extend the XML standard to include a <configOption> parameter, it is recommended to use <configOption xmlns=”urn:XYZ123-com:configuration:options”> to avoid future namespace conflicts.

5.3.4 Protocol requirements

A system/device shall support transport of XML via either the HTTP/1.0 or HTTP/1.1 protocol as specified in RFC 1945 and RFC 2616, respectively. It is highly recommended that HTTP/1.1 is used in order to support key features (persistent connections, HTTPS, etc.). When HTTP 1.0 is implemented, the client applications shall not issue multiple messages to the target systems/devices.

5.4 HTTP methods and REST

The CRUD operations are defined by the HTTP method as shown in the table1 below.

Table 1 – HTTP methods

HTTP method	Operation
POST	Create the resource
GET	Retrieve the resource
PUT	Update the resource
DELETE	Delete the resource

Rules of thumb.

GET calls should never change the system state. They are meant to only return data to the requestor and not to have any side effects

POST calls should only be used to ADD something that did not already exist.

PUT calls are expected to update an existing resource but if the resource specified does not already exist, it can be created as well. This will be the assumed default behavior of PUT calls. If any resource wishes to deviate from this behavior, it should be considered an exception and this should be noted in the implementation notes of the resource.

5.5 HTTP status codes and REST

The following Table 2 shows how the HTTP status codes map to REST operations along with the general use case for response headers and bodies. For more information, please see the table under each REST API.

Table 2 – HTTP status codes and REST

HTTP status codes	REST meaning	POST	GET	PUT	DEL
200	“OK” - The request has succeeded. Header Notes: None Body notes: The requested resource will be returned in the body.		X	X	
201	“Created” - The request has created a new resource. Header notes: The *Location* header contains the URI of the newly created resource. Body notes: The response returns an entity describing the newly created resource.	X			
204	“No Content” - The request succeeded, but there is no data to return. Header notes: None Body notes: No body is allowed.			X	X
301	“Moved Permanently” - The requested resource has moved permanently. Header notes: The *Location* header contains the URI of the new location. Body notes: The body may contain the new resource location.		X		
302	“Found” - The requested resource should be accessed through this location, but the resource actually lives at another location. This is typically used to set up an alias. Header notes: The *Location* header contains the URI of the resource. Body notes: The body may contain the new resource location.		X		
400	“Bad Request” - The request was badly formed. This is commonly used for creating or updating a resource, but the data was incomplete or incorrect. Header notes: The Reason-Phrase sent with the HTTP status header may contain information on the error. Body notes: The response may contain more information of the underlying error that occurred in addition to the Reason-Phrase.	X		X	
401	“Unauthorized” - The request requires user authentication to access this resource. If the request contains invalid authentication data, this code is sent. Header notes: At least one authentication mechanism shall be specified in the *WWW-Authenticate* header. The Reason-Phrase sent with the HTTP status header may contain information on the error. Body notes: The response may contain more information of the underlying error that occurred in addition to the Reason-Phrase.	X	X	X	X
403	“Forbidden” - The request is not allowed because the server is refusing to fill the request. A common reason for this is that the device does not support the requested functionality. Header notes: The Reason-Phrase sent with the HTTP status header may contain information on	X	X	X	X

HTTP status codes	REST meaning	POST	GET	PUT	DEL
	the error. Body notes: The response may contain more information of the underlying error that occurred in addition to the Reason-Phrase.				
404	“Not Found” - The requested resource does not exist. Header notes: None Body notes: None	X	X	X	X
405	“Method Not Allowed” – The request used an HTTP method that is not supported for the resource because the {API Protocol} specification does not allow this method. If the device does not support the functionality but it is a valid {API Protocol} operation, then a 403 is returned. Header notes: The *Allow* header lists the supported HTTP methods for this resource. Body notes: None	X	X	X	X
500	“Internal Server Error” - An internal server error has occurred. Header notes: None Body notes: None	X	X	X	X
503	“Service Unavailable” – The HTTP server is up, but the REST service is not available. Typically this is caused by too many client requests. Header notes: The *Retry-After* header suggests to the client when to try resubmitting the request. Body notes: None	X	X	X	X

5.6 Unique identifiers

IDs are defined as URL-Valid Strings, as required by REST. The device will create an ID for all resources that add a resource.

IDs within each type should be unique at least on the channel level, but there is no requirement for uniqueness across devices. If globally unique IDs are desired, a globally unique ID should be derived using the method described in RFC 4122.

5.7 ID encoding

Because IDs will occur as part of a URI, there are two ways to encode an ID: either following RFC 3986 or, for pure binary IDs, as a hex string.

RFC 3986 first converts the URI to UTF and then prints the following unreserved characters in the URI without any encoding:

- A-Z
- a-z
- 0-9
- -
- .
- _
- ~

All non-printable or reserved characters will be encoded as a two digit hex value prefixed by a %. For example, a space (ASCII value of 32) will be encoded as %20.

Because a pure binary ID can contain values that might interfere with the operation of browsers and web servers, protocols support hex encoding of the ID. The ID shall begin with 0x (0X is also acceptable) followed by pairs of hex values. Each hex pair represents a single byte in the ID. For example: 0x3F431245DE67FAC46F9D034CA23AEFD4. The hexadecimal characters A-F can also be represented by a-f. So 0x3f431245de67fac46f9d034ca23aefd4 is equivalent to the previous ID.

If readable IDs are desired, it is recommended that IDs are created with unreserved, printable ASCII characters.

6 Architecture and namespace

In a typical REST-based namespace, every node or object in the tree-structured hierarchy is considered a resource.

The service model adds a resource sub-class called “Service”. Services are simply nodes which can contain other nodes. Nodes that do not contain other nodes (other than the standard node resources of the model) will continue to be called resources, while the term node will be used to refer to both services and resources.

Viewed as a tree, services are analogous to branches and resources are analogous to leaves.

Figure 1 below provides an example of PSIA service architecture.

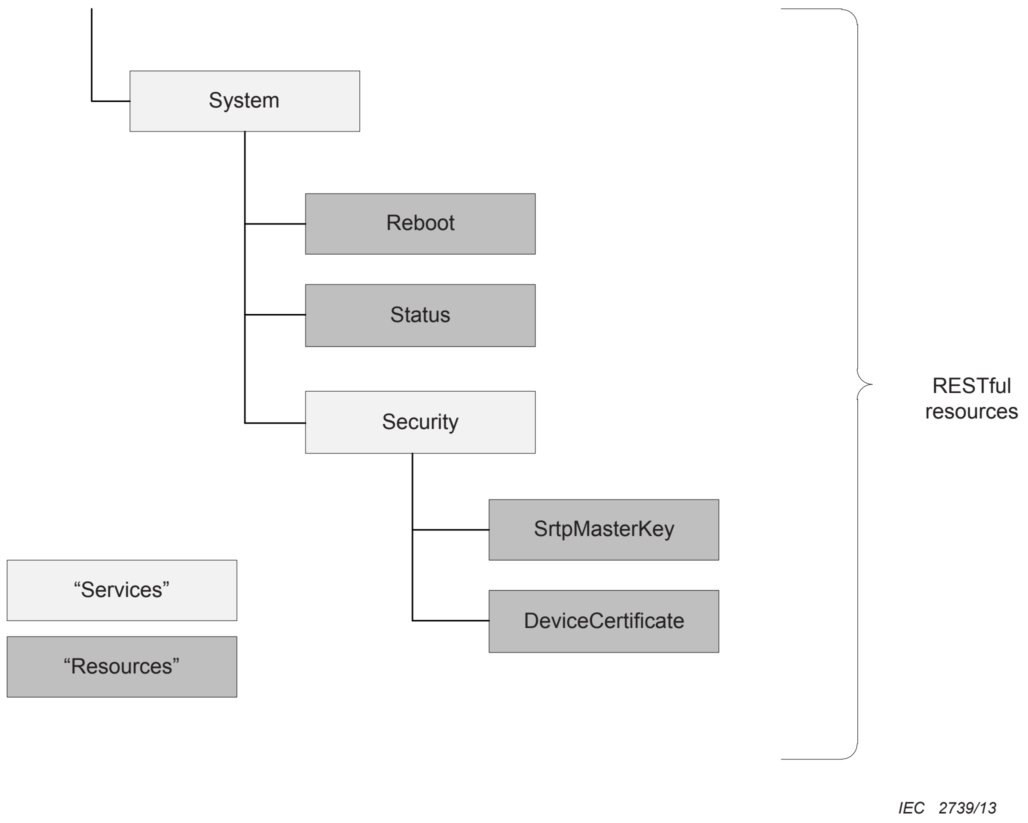

Figure 1 – PSIA service architecture example

Each node shall contain the following standard resources (see Table 3):

description	which will respond to an HTTP GET with a ResourceDescription datablock
index	which will respond to an HTTP GET with a ResourceList datablock

Each node may contain the following standard resources:

indexr	which will respond to an HTTP GET with a ResourceList datablock
capabilities	which will respond to an HTTP GET with a resource-specific XML Document

The index resource will return a list of all the immediate "children" of a node. For services, this list could contain other services as well as resources (see Table 4). For resources, this list should only indicate which standard resources (IE description, index, and optionally indexr and capabilities) are contained. The optional indexr resource will return a recursive listing that descends through the namespace hierarchy.

Table 3 – Resource names

Resource name	Description	Mandatory/optional
description	will respond to an HTTP GET with a <ResourceDescription> datablock	Mandatory
capabilities	will respond to an HTTP GET with a resource-specific XML Document	Optional
index	will respond to an HTTP GET with a <ResourceList> datablock	Mandatory
indexr	will respond to an HTTP GET with <ResourceList> datablock	Optional

For all protocols of this clause, the root namespace of "PSIA" is mandated, meaning it has to be included in the URL. Therefore, the root of any service's namespace will be "PSIA". Each service will be mandatory or optional, indicating to implementers which services they shall implement at a minimum. Within each service, resources will also be mandatory or optional. This scope will be hierarchical so that any resource of an optional service is, by definition, optional but if an optional service type is deployed, then every mandatory resource within that service then becomes mandatory. Table 4 lists the possible service types.

Table 4 – Service URLs

Service URL	Description	Mandatory/optional
/System	Resources related to general system configuration and operation	Mandatory
/System/Storage	Resources related to local storage	Optional
/System/Network	Resource related to network settings	Mandatory
/Security	Resources related to security of the device	Mandatory
/Security/AAA	Resources related to AAA functions	Mandatory
/Streaming	Resources related to streaming media content	Optional
/PTZ	Resources related to Pan/Tilt/Zoom	Optional
/Archive	Resources related to storage of content	Optional
/Diagnostics	Resources related to diagnostics	Optional
/Custom	Resources that are specific to a protocol or vendor specific	Optional

Multiple channels and versions

To provide for multi-channel support, a service shall insert the implied “Channels” service as a child-node which should then contain an ID resource for each channel. Each ID resource will then respond to each of the resources applicable to the service.

For Single-Channel Devices, the Channels service shall still be included to maintain consistency between single and multi-channel devices and to provide for the case where a multichannel device has only created a single channel.

Note that Channel IDs are arbitrarily assigned by the device.

```
(EG.For a single channel device:
        /Streaming/Channels/0/keyFrame
For a multi-channel device
        /Streaming/Channels/0/keyFrame
        /Streaming/Channels/1/keyFrame)
```

Devices may either pre-define this multichannel structure or support dynamic additions and deletions of channels (using HTTP POST and DELETE) as applicable.

In order to differentiate services that essentially provide for multiple instances of something within the hierarchy, it is recommended that services at the root level be referred to as “Root Services” while the term service continue to be used to describe any node that contains other nodes (EG Streaming is a Root Service, Channels is not).

Each node, be it a resource or service, will be able to return a description of itself within the service model. This description will include a version attribute to support versioning within the Service model. While this practice will allow resources with different versions to exist within the same services, it is mandatory that all resources within a service container are fully backward compatible.

If a new service version is introduced that does not maintain backwards compatibility with previous versions, then a new service shall be created for the new, incompatible version (EG /Streaming and /StreamingV2). IE it is acceptable to add resources to a Service but not to replace them with new versions that are not backward compatible. If new resource versions shall be added, the Root Service name should be changed to indicate a new Service version.

7 System flow

7.1 General

Before any protocol can be used to manage a device, it shall first be discovered. It is required that the Zeroconf (Zero Configuration Networking) technology be supported to discover/locate the device. Once this step is accomplished, transactions can commence.

While devices shall support ZeroConf, this does not preclude devices from using DHCP or manual IP Addressing. Devices should check for manually assigned IP addresses and DHCP assigned IP Addresses before attempting to assign an address using IPv4 Link-Local Addressing (which is the method for Ip Address discovery for ZeroConf).

ZeroConf is normally expected to operate in a local area network. Where discovery shall be supported in a routed or Wide Area Network, part 2 of ZeroConf (Multicat DNS) becomes superfluous and part 3 (DNS-SD) shall be supported by configuration of actual DNS servers.

HTTP requests are made through the device’s web server. The HTTP response may contain XML content (for GET actions), XML response information (for PUT or POST actions), or various text/binary content (for retrieval of configuration data, etc.). Edge devices should be able to handle overlapping/simultaneous HTTP requests, as well as persistent connections to handle multiple HTTP transactions.

The XML content should be described by .xsd documents. Relevant XML data structures shall be documented in an Appendix section of each Specification.

7.2 Service discovery

Zeroconf (Zero Configuration Networking) technology specifies DNS-SD (DNS Service Discovery as described in http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt) to discover/locate a device.

All protocols of this subclause will require DNS-SD for device discovery. To support this discovery model, the PSIA is registering a DNS SRV (RFC 2782) service type to be used to discover all IP video protocols via DNS–SD (DNS Service Discovery).

DNS-SD discoveries for the PSIA’s public DNS service type should be used to discover the device according to DNS Service Discovery (http://www.dns-sd.org/ServiceTypes.html). Once a device is established as a compliant device, its services and resources can be discovered using standard HTTP GETs using the standard, mandatory resources.

The following information should be advertised:

A path of “/index/” – can be obtained from the “path” key in the TXT record

The {host} – can be obtained from the service’s SRV record

The {port} – can be obtained from the service’s SRV record

The version of the DNS SVR record in “txtvers”

The IP video protocol version in “protovers”

Once a compliant device has been discovered, an HTTP GET of its mandatory index resource will return a list of the services that it supports. At this point, the standard methods can be used to “walk” the namespace tree and discover the supported services and resources.

It should be noted that the index resource returns only the first level resources of a node, but the indexr resource will return a recursive tree structured list with the current resource as root.

7.3 Persistent connections

Devices that implement HTTP/1.1 should support persistent connections in order to support video management systems or client applications that issue multiple HTTP(S) transactions. This standard assumes that HTTP/1.1 is implemented and utilized according to RFC 2616. For persistent connections with devices that support HTTP/1.0 RFC 2068 section 19.7.1 should be referenced.

A video management system or client application should, when using a persistent connection for multiple transactions, implement the “Connection: Keep-Alive” HTTP header. The management system should also use the “Connection: close” HTTP header field for the last transaction made within this persistent connection. This process assumes that the application is aware of the last request in a sequence of multiple requests.

7.4 Authentication

When an application sends any request to the device, it shall be authenticated by means of Basic Access or Digest authentication according to RFC 2617. This means the user access credentials are sent along with each request. If a user is authenticated, the request will follow the normal execution flow. Basic Access and/or Digest authentication are mandatory for all device implementations. It is up to the client to determine which method to use for different deployment scenarios.

A default user account, “admin”, shall be provided by the IP device. This account should have a default privilege level of “administrator”, and shall not be deleted. The default password of the “admin” account should be null in factory default configuration.

Example client HTTP request header and body with no authentication credentials:

```
GET /index
…
```

Example unauthorized HTTP response header and body:

```
HTTP/1.1 401 Unauthorized
…
WWW-Authenticate: Digest realm="testrealm@host.com",
                         qop="auth,auth-int",
                         nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                         opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: application/xml; charset=”UTF-8”
Content-Length: xxx    (note: xxx = size of XML block)

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

Example client HTTP request header and body with authentication credentials (username “Mufasa” and password “Circle of Life”):

```
GET /index
…
Authorization: Digest username="Mufasa",
                      realm="testrealm@host.com",
                      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                      uri="/dir/index.html",
                      qop=auth,
                      nc=00000001,
                      cnonce="0a4f113b",
                      response="6629fae49393a05397450978507c4ef1",
                      opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Example authorized HTTP response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/xml; charset=”UTF-8”
Content-Length: xxx    (note: xxx = size of XML block)

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

7.5 Access restrictions

All supported resources on a device shall be fully accessible to users with the “Administrator” privilege level. This means that in order to use the full suite of resources a device offers, authentication shall be granted with a user account having a privilege level corresponding to "Administrator".

It is required that at least one account with Administrator privileges be active at all times.

There are no restrictions as to which resources are accessible to users with other privilege levels. A vendor may choose to limit, for example, the allowable resources for user accounts with lower privileges. However, since user-specific authorization is not a function of the protocol, it is often assumed that full administrative rights will be available via the protocol. User-specific authorization functions are expected to be handled by the calling application.

While "Administrator" privilege levels shall be provided for, there is no requirement that any specific users be assigned an administrative privilege level. In cases where external requirements preclude a single user having all privileges, more granular authorization can be performed using user and role assignments which do not have full administrative privileges.

7.6 Setting configurations

Resources to set device configurations will use the HTTP PUT method if there is an XML block parameter for the request, and the HTTP GET method if there is no XML block parameter. The inbound XML format is defined according to a resource-specific XML schema For PUT operations, the request status will be indicated by the XML response information returned from the device, and can be used to indicate the status of the set operation. This XML format is defined according to "XML Response Schema" (see 7.13 for details). After successfully updating the repository, the device returns an XML response with status code "OK". A separate status code is used for unsuccessful operations. In either case, the device will not return a response until it is ready to continue normal operation – this includes accepting streaming requests, receiving behavioral control commands, etc.

Example HTTP request header and body:

```
POST /System/deviceInfo HTTP/1.1
…
Content-Type: application/xml; charset=”UTF-8”
Content-Length:xxx    (note: xxx = size of XML block)

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

Example HTTP response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/xml; charset=”UTF-8”
Content-Length: xxx    (note: xxx = size of XML block)

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

7.7 Getting configurations

Resources to get device configurations or status information will use the HTTP GET method. If successful, the result will be returned in XML format according to the resource description. If the request is unsuccessful for any reason (i.e. not authenticated), the result will be returned in XML format according to "ResponseStatus XML Schema". The Content-Type and Content-Length will be set in the headers of the HTTP response containing the XML data. The Content-Type is: application/xml; charset="UTF-8".

Example HTTP request header and body:

```
GET /System/deviceInfo HTTP/1.1
…
```

Example HTTP response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/xml; charset=”UTF-8”
Content-Length:  xxx    (note: xxx = size of XML block)

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

7.8 Getting capabilities

Capabilities can also be retrieved by any resources node that specifies an XML payload for inbound data with an HTTP GET of its “capabilities” resource. In other words, a client application can query a device for its capabilities in order to understand what XML tags are supported, the acceptable data ranges, etc. See 5.5 for more details on the returned capabilities.

Example HTTP request header and body:

```
GET /PTZ/channels/ID/0/absolute/capabilities HTTP/1.1
…
```

Example HTTP response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/xml; charset=”UTF-8”Content-Length: xxx (note: xxx = size of
XML block)
<?xml version="1.0" encoding="UTF-8" ?>
<PTZData version=“1.0” xmlns=“urn:psialliance-org”>
<pan min=”-100” max=”100”/>
<tilt min=”-100” max=”100”/>
<zoom min=”-100” max=”100”/>
<Momentary>
        <duration min=”0”/>
</Momentary>
<Relative>
    <positionX min=”0” max=”1024”/>
    <positionY min=”0” max=”1024”/>
    <relativeZoom min=”-100” max=”100”/>
</Relative>
<Absolute>
    <elevation min=”-90” max=”90”/>
    <azimuth min=”0” max=”360”/>
    <absoluteZoom min=”0” max=”100”/>
</Absolute>
<Digital>
    <positionX min=”0” max=”1024”/>
    <positionY min=”0” max=”1024”/>
    <digitalZoomLevel min=”0” max=”100”/>
</Digital>
</PTZData>
```

7.9 Uploading data

Resources to upload data (i.e. firmware, configuration file, etc.) to the device will use the HTTP PUT method. The content of the data will be stored in the body of the HTTP request. The Content-Type and Content-Length will be set in the headers of the HTTP request. The Content-Type is "application/octet-stream". In addition, each resource may optionally specify a different inputXML structure.

Example HTTP upload request header and body:

```
POST /System/configurationData HTTP/1.1
…
Content-Type: application/ xml; charset="UTF-8"

[proprietary configuration data content]
```

Example HTTP upload response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/xml; charset="UTF-8"
Content-Length: xxx    (note: xxx = size of XML block)

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

7.10 Receiving data

Resources to receive data (i.e. configuration file, etc.) from the device will use the HTTP GET method. The content of the data will be stored in the body of the HTTP response. The Content-Type and Content-Length will be set in the headers of the HTTP response, according to the type of data being returned.

The client may use the Accept: header string to tell the server what formats it accepts. Depending on what the client accepts, the server may transcode, transform or even compress the data to match the client's expectations.

Example HTTP download request header and body:

```
GET /System/configurationData HTTP/1.1
…
```

Example HTTP download response header and body:

```
HTTP/1.1 200 OK
…
Content-Type: application/octet-stream
Content-Length: xxx    (note: xxx = size of XML block)

[proprietary configuration data content]
```

7.11 Operations

For stateless operations (i.e. function calls) the formula is:

PUT /Service/<Operation>

Resources shall indicate in their descriptions which XML payload is required or the query string parameters to be used in the operation.

7.12 Diagnostics

Diagnostics (and other stateful operations) run in the background on the device, so it shall be possible to create them asynchronously and be able to query their status. Table 5 lists various HTTP requests available to manage diagnostic operations.

The REST model works well here:

Table 5 – HTTP requests

Request	Result
POST /Diagnostics/<command>	Returns diagnostic ID
GET /Diagnostics/<command>/<ID>	Get information on this ID
DELETE /Diagnostics/<command>/<ID>	Delete command in progress
GET /Diagnostics/commands	Get information on all commands running

7.13 Response status

7.13.1 General

Responses to many resource calls contain data in the form of the ResponseStatus XML document.

Within each specification, separate services and resources may each have their own data structures. The only provision of the model is that each ResourceDescription shall indicate which structures are used and each structure shall be defined in an XML schema document within the specification document. If resources do not define their own response structures, they may use the ResponseStatus structure of this standard as defined in Clause 10.

7.13.2 Status code

A ResponseStatus with statusCode=OK will be sent after the command has been completely processed on the device. Even if the request contains some parameters that are not supported, the device will ignore those parameters and return statusCode=OK.

A device will send a Device Busy response to a command which cannot be processed at that time (eg. receiving a reboot command while the flash is being updated)

If the device fails to perform the request - possibly due to a hardware error - it will return a Device Error statusCode and a fault message in the statusString.

An Invalid Operation statusCode is returned in response to a command that has not been implemented. Invalid Operation is also returned if an authentication attempt fails or the logged in user has insufficient privileges to execute the command.

An Invalid XML Format statusCode is returned if the XML is badly formed and causes the parser to fail. The statusString should indicate the fault.

An incomplete message or a message containing an out-of-range parameter will return an Invalid XML Content statusCode and associated statusString.

For settings that require a reboot to take effect, such as changing the network address or a firmware update, the Reboot Required statusCode is returned.

7.13.3 Status string

It is recommended that for all responses where the returned statusCode is not OK, a descriptive statusString be returned indicating the reason the command was not completed.

7.13.4 ID

In POST operations where the device will return an ID of the resource created, this attribute will be used to pass back the created ID.

7.14 Processing rules

Any field (particularly in the inbound XML parameters) that is not supported by the device should be ignored. For any given resource there may be some special processing rules. These rules are documented in the column associated with the heading “Implementation Note”.

8 XML modeling

8.1 File format

All XML files shall use UTF-8 (8-bit UCS/Unicode Transoformation Format) encoding according to RFC 3629. A BOM (byte-oder mark) can optionally be used. Thus, a media device should support- UTF-8 encoding with or without a BOM.

8.2 Data structures

Any Resource can specify separate input and output XML Documents. If a specific data structure is defined, these shall be specified as XML Schema Documents (xsd) within the specification. The xsd’s created for specifications based on this HTTP and REST service standard are to be included in the appendix section of the relevant specification. In addition, the PSIA will be posting xsd documents of relevant schemas at http://www.psialliance.org to support online reference of the schemas. However, there is no guarantee that the schemas will be posted at the same time the documents are published. For this reason, the schema definitions within the specification documents themselves are the minimal requirement.

8.3 Lists

Many of the XML blocks contain lists. The syntax of these lists is <XXXList>, where XXX is a name referring to the XML setting. Inside of the <XXXList> tag is one or more <XXX> nodes. As an example, the <ChannelList> block may contain content as such:

```
<ChannelList>
	<Channel>
		<id>1</id>
		…
	</Channel>
	<Channel>
		<id>2</id>
		…
	</Channel>
	…
</ChannelList>
```

8.4 Capabilities

Capabilities for any resource that defines an XML block for input will be returned as an XML document that is essentially an XML instance of the resource-specific input XML block. This XML document shall contain the acceptable values for each attribute (see Table 6).

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

Table 6 – Capability attributes

Capability attribute	Description	Syntax	Applicable XML data types

Min	The minimum character length for a string, or the minimum numerical value of a number	Examples:
		min="0"
		min="64"
		min="-100" (numerical only)
		min="1,2"
			All except fixed data types [a]

max	The maximum character length for a string, or the maximum numerical value of a number	Examples:
		max="5"
		max="64"
		max="4 096"
		max="10,50"
			All except fixed data types [a]

range	Indicates the possible range of numerical values within the "min" and "max" attributes of an element. This attribute should only be used if the possible values for an XML element does not include the entire numerical range between "min" and "max" attributes	Ranges are listed in numerical order separated by a "," character. A range has the form "x~y" where x is the range floor and y is the range ceiling. Single numbers may also be used. *Example*: if an XML element supports values 0, 123, 1024 to 2000, and 2003, the syntax would be: range="0,123,1024~2000,2003"	All numerical data types

opt	Lists the supported options for a CodeID data type. Required for XML elements with a CodeID data type. This attribute should *not* be used for any other data type	If all options are supported, the syntax is "all". Otherwise, supported options are listed separated by a "," character.
		Examples:
		opt="all"
		opt="1,2,3"
		opt="1,2,5,8,9,10,11"
			CodeID

Def	Indicates the default value of the XML element. If the element has no default value, this attribute should *not* be used	Examples:
		def="1234"
		def="Device ABC"
		def="3"
			All data types

reqReboot	Indicates if configuration of this XML element requires a device reboot before taking effect. If an element doesn't require a reboot, this attribute should *not* be used	reqReboot="true"	All data types
dynamic	Indicates if an XML element has dynamic	dynamic="true"	All data types

Capability attribute	Description	Syntax	Applicable XML data types
	capabilities dependent on other XML configurations. For example, if an element's data range changes based on another element's configured value, this attribute shall be used. In this case, the element's capability attributes shall always reflect the current device configuration		

Size	Indicates the maximum number of entries in an XML list. This attribute is only applicable to XML list elements. This attribute should not be used for any other type of element (see 8.3 for details)	*Example*: If a device supports 5 users the example would be
		
		<UserSetting>
		  <UserList size="5">
		    …
			Only supported for list elements (see section 8.3)

a Fixed, pre-defined data types do not need certain capability attributes because their formats/data ranges are already defined. Where pre-defined data types are used, each protocol document shall include an enumeration of these formats in an appendix.			

9 Custom services and resources

In order to support system/device specific resources that are not common to the public service definitions, the CUSTOM service type is provided. An HTTP GET of the index resource of the CUSTOM service returns a list of the custom services and resources supported by the system/device.

For each custom resource, an implicit mandatory resource named "Description" shall be supported. An HTTP GET of any custom resource's Description resource shall return a ResourceDescriptionBlock similar to the Resource Description information described in 7.7.

Custom services and resource can be used to support protocol-specific resources that are thought to be of an interim nature (i.e. a forthcoming protocol will most probably deprecate these resources) or vendor-specific proprietary resources. As long as all custom services and resources are implemented according to this Service Model, they can be discovered and called by clients and applications compliant to this clause.

10 Interface design

10.1 General

The HTTP URL format is of the general form

```
<protocol>://<hostname>:<port>/<URI>? P1=v1&P2=v2....&pn=vn
```

All requests will follow this format. A brief description of these components follows:

10.2 Protocol

The protocol field refers to the URL scheme that will be used for the particular request. Note that the current specification allows the following schemes:

- http
- https

10.3 Hostname

The hostname field refers to the hostname, IP address, or the FQDN (fully qualified domain name) of an IP device.

10.4 Port

The port field indicates the port number to be used for the HTTP request. The default port number for HTTP is 80. For HTTPS, the default port is 443. For RTSP, the default port is 554. If neglected in the URL, these default port numbers will be used for the request (as defined in RFC 2616, RFC 2818, and RFC 2326 respectively).

The HTTP and HTTPS port number is configurable for IP devices. The standard HTTP and HTTPS ports (80 and 443) will be assumed unless otherwise specified.

10.5 URI

The URI absolute path is most often of the form "<SERVICE>/<resource>" where <resource> corresponds to one of the resources defined in the specification. For example, <SERVICE> could refer to "System" or "Security". This is true for resources that update or retrieve device configurations.

10.6 Query string

Resources specify required and optional query string parameters. In either case, these query string parameters shall be listed in name-value pair syntax (p1=v1&p2=v2...&pn=vn) following the URI.

Example GET HTTP request with query string parameters:

```
GET /Streaming/Channels/1/picture?snapShotImageType=jpeg
…
```

Example POST HTTP request with query string parameters:

```
PUT /System/time?localTime=2009-02-16%2013:30:00
…
```

Each resource may define a set of parameters, in the form of name-value pairs, which exist in the query string. If resources define input data specific to the resource, this takes precedence over the use of query string parameters for input.

10.7 Resource description

For each resource in this document, the following components are defined:

Format – indicates the URL format of the HTTP request

Type – indicates whether this is a service or resource

Method specific (GET, PUT, POST, DELETE)

Query string parameters – indicates the name/value pairs (P1,P2,P3,...Pn) for the resource.

Inbound data – indicates inbound data for the resource as follows:

- **NONE** – indicates no input data
- **DataBlock** – the name of a Data Block defined within the specification. Datablocks used here shall be defined within the specification document. In addition, it is strongly recommended that .xml schema documents be created for each referenced datablock.
- **MIME type** – indicates that the input data is in the HTTP payload with the indicated MIME type. NOTE a type of "application/xml" is not considered valid.

If a device does not support particular XML tags or blocks, they need not be used in the resource operations.

Generally, if fields are not provided in the inbound XML, the current values for these fields should remain unchanged in the device's repository.

If required fields do not already exist in the device's repository, they shall be provided in applicable resource operations.

Function – describes the general function behavior

Return result – describes the response from the HTTP request

Implementation note – describes the implementation behavior and any special processing rules for the resource.

For example,

URI	`index`	Version	1.0	Type	Resource
Function	Enumerate child nodes				
Methods	Query String(s)	Inbound Data		Return Result	
GET	None	None		<ResourceList>	
Notes	Returns a flat (non-recursive) listing of all child nodes				

In order to support discovery of CUSTOM service resources, this resource description data structure is also captured as a data block of type ResourceDescription. Whenever an HTTP GET of a device's CUSTOM/Index resource is executed, a list of the device's custom resources is returned. For each custom resource, an HTTP GET of the mandatory resource "Description" will return a ResourceDescription Block indicating what the resource does and how it should be used.

11 Standard resource descriptions

11.1 General

This clause describes the standardized resources.

11.2 index

URI	`index`	Version	1.0	Type	Resource
Function	Enumerate child nodes				
Methods	Query String(s)	Inbound Data		Return Result	
GET	None	None		<ResourceList>	
Notes	Returns a flat (non-recursive) listing of all child nodes				

11.3 indexr

URI	`indexr`	Version	1.0	Type	Resource
Function	Enumerate child nodes				

Methods	Query String(s)	Inbound Data	Return Result
GET	None	None	<ResourceList>
Notes	Returns a recursive listing of all child nodes		

11.4 description

URI	`Description`	Version	1.0	Type	Resource
Function	Describe Current Resource				
Methods	Query String(s)	Inbound Data		Return Result	
GET	None	None		<ResourceDescription>	
Notes	Returns a description of the resource				

11.5 capabilities

URI	`capabilities`	Version	1.0	Type	Resource
Function	Return capabilities of Current Resource				
Methods	Query String(s)	Inbound Data		Return Result	
GET	None	None		Resource-Specific	
Notes	Returns a Capabilities description of the resource				

11.6 Schemas

11.6.1 General

The following data structures are defined for use with the Service Model of this subclause. The formats used in this subclause are basic samples intended to quickly demonstrate the structure of the data blocks. Note that the actual video IP protocols of this subclause are to include their documented data structures as .xsd files.

11.6.2 ResourceDescription

```
<ResourceDescription version="1.0" xmlns="urn:psialliance-org:resourcedescription">
   <name>index</name>
   <version>1.0</version>
   <type>resource</type>
   <get>
         <queryStringParameterList>none</queryStringParameterList>
         <inboundXML>none</inboundXML>
         <function>enumerate 1st level children</function>
         <returnResult>ResourceList</returnResult>
         <notes>non-recursive</notes>
   </get>
   <put></put>
   <post></post>
   <delete></delete>
</resourceDescription>
```

11.6.3 ResourceList

```
  <?xml version="1.0" encoding="utf-8" ?>
- <ResourceList version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="urn:psialliance-org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:psialliance-org
http://www.psialliance.org/XMLSchemas/service.xsd">
-  <Resource version="1.0" xmlns="urn:psialliance-org" xlink:href="/index">
         <name>index</name>
         <type>resource</type>
  </Resource>
- <Resource xlink:href="/System">
   <name>System</name>
   <type>service</type>
-  <ResourceList>
-        <Resource xlink:href="/System/Network">
               <name>Network</name>
               <type>service</type>
-              <ResourceList>
-                    <Resource xlink:href="/System/Network/ipAddress">
                           <name>ipAddress</name>
                           <type>resource</type>
                     </Resource>
               </ResourceList>
         </Resource>
   </ResourceList>
  </Resource>
  </ResourceList>
```

11.6.4 QueryStringParameterList

```
  <?xml version="1.0" encoding="utf-8" ?>
- <QueryStringParameterList version="1.0" xmlns="urn:psialliance-org">
-  <QueryStringParameter>
         <name>positionX</name>
         <type>integer</type>
         <description>X position of scaling window</description>
   </QueryStringParameter>
  </QueryStringParameterList>
```

11.6.5 responseStatus

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

11.6.6 service.xsd

The following XML Schema Document contains XML schema definitions for all of the Service Model data structures. All specifications of this subclause are to use this schema document to maintain consistency of the Service Model data structures.

This document and all subsequent XML Schema Documents will be posted at http://www.psialliance.org.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema version="1.0"
           targetNamespace="urn:psialliance-org"
           xmlns="urn:psialliance-org"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xlink="http://www.w3.org/1999/xlink"
                 elementFormDefault="qualified">
  <xs:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="xlink.xsd"/>

  <xs:annotation>
    <xs:documentation xml:lang="en">
      PSIA Core Service Schema
    </xs:documentation>
  </xs:annotation>

  <!-- ID -->
  <xs:simpleType name="Id">
    <xs:restriction base="xs:string">
      <!-- TODO -->
    </xs:restriction>
  </xs:simpleType>

  <!-- StatusCode -->
  <xs:simpleType name="StatusCode">
    <xs:restriction base="xs:int">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="7"/>
    </xs:restriction>
       <!-- O=1-OK, 2-Device Busy, 3-Device Error, 4-Invalid Operation, 5-
Invalid XML Format, 6-Invalid XML Content; 7-Reboot Required-->
  </xs:simpleType>

  <!-- ResourceType -->
  <xs:simpleType name="ResourceType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="service"/>
      <xs:enumeration value="resource"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- QueryStringParameter -->
  <xs:complexType name="QueryStringParameter">
    <xs:sequence>
              <xs:element name="name" type="xs:string" />
       <xs:element name="type" type="xs:string" />
       <xs:element name="description" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- QueryStringParameterList -->
  <xs:complexType name="QueryStringParameterList">
    <xs:sequence>
      <xs:element name="QueryStringParameter" type="QueryStringParameter"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- URLParameters -->
  <xs:complexType name="URLParameters">
    <xs:sequence>
      <xs:element name="queryStringParameterList"
type="QueryStringParameterList" />
         <xs:element name="inboundData" type="xs:string" />
         <xs:element name="returnResult" type="xs:string" />
         <xs:element name="function" type="xs:string" />
         <xs:element name="notes" type="xs:string" />
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

<!-- ResponseStatus -->
  <xs:complexType name="ResponseStatus">
    <xs:sequence>
      <xs:element name="requestURL" type="xs:anyURI" />
      <xs:element name="statusCode" type="StatusCode" />
      <xs:element name="statusString" type="xs:string" />
      <xs:element name="id" type="Id" minOccurs="0" maxOccurs="1" />
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
       <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- ResourceDescription -->
  <xs:complexType name="ResourceDescription">
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
         <xs:element name="version" type="xs:string" />
         <xs:element name="type" type="ResourceType" />
         <xs:element name="description" type="xs:string" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="notes" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="get" type="URLParameters" />
         <xs:element name="put" type="URLParameters" />
         <xs:element name="post" type="URLParameters" />
         <xs:element name="delete" type="URLParameters" />
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Resource -->
  <xs:complexType name="Resource">
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
         <xs:element name="version" type="xs:string" />
         <xs:element name="type" type="ResourceType" />
         <xs:element name="description" type="xs:string" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="ResourceList" type="ResourceList" minOccurs="0"
maxOccurs="1"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>
  <!-- ResourceList -->
  <xs:complexType name="ResourceList">
    <xs:sequence>
      <xs:element name="Resource" type="Resource" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>

</xs:schema>
```

Annex A
(normative)

IP Media Device API Specification Version 1.0

A.1 Overview

This annex specifies an interface that enables 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 video surveillance products from different vendors.

A.2 Scope

As the first standard 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 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 specification addressing events is published.

Suggestions for corrections to this version of the specification and additions to the protocol should be submitted to IEC or directly to the PSIA Forum’s IP Media Specification area. A Forum thread used to track issues for the next version is located at:

http://www.psiaforums.org

Please post a reply to this thread to submit your suggested correction or addition.

A.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.

A.4 Conformance

A.4.1 General

This subclause conforms to the Service model introduced in the previous subclause, 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 IP video HTTP and REST specifications and are intended to be referenced by other specifications.

The optional services defined are specific to IP media devices.

A.4.2 Service requirements

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

REQ	Serivce URL	Notes
✓	/	
✓	/System	
	/System/Storage	Not all IP media devices support storage.
✓	/System/Network	
	/System/IO	
	/System/Audio	
	/Sysytem/Video	
	/System/Serial	
	/Diagnostics	
✓	/Security	
	/Security/AAA	
	/Streaming	
	/PTZ	
	/Custom/MotionDetectio n	
	/Custom/Event	

A.4.3 Resource requirements

A.4.3.1 General

The following resources are required for the implemented services.

A.4.3.2 Root service

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

A.4.3.3 /System

REQ	Command	GET	PUT	POST	DEL
✓	reboot		✓		
✓	updateFirmware		✓		
✓	configurationData	✓	✓		
✓	factoryReset		✓		
✓	deviceInfo	✓	✓		
✓	supportReport	✓			
✓	status	✓			
✓	time	✓	✓		
✓	time/localTime	✓	✓		
✓	time/timeZone	✓	✓		
✓	time/ntpServers	✓	✓	✓	✓
✓	time/ntpServers/<ID>	✓	✓		✓
	logging	✓	✓		
	logging/messages	✓			

A.4.3.3.1 /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	✓			

A.4.3.3.2 /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	✓	✓	✓	✓
	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>	✓	✓		✓

A.4.3.3.3 /System/IO

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

A.4.3.3.4 /System/Audio

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

A.4.3.3.5 /System/Video

REQ	Command	GET	PUT	POST	DEL

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

A.4.3.3.6 /System/Serial

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

A.4.3.4 /Diagnostics

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

A.4.3.5 /Security

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

A.4.3.5.1 /Security/AAA

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

A.4.3.6 /Streaming

REQ	Command	GET	PUT	POST	DEL
✓	status	✓			
✓	channels	✓	✓	✓?	✓?

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

A.4.3.7 /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		✓		
	channels/<ID>/patrols/<ID>/status	✓			
	channels/<ID>/patrols/<ID>/schedule	✓	✓		

A.4.3.8 /Custom/MotionDetection

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

A.4.3.9 /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	✓			

A.5 Media streaming

A.5.1 General

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

A.5.2 Streaming with RTP and RTSP

A.5.2.1 General

An IP media device shall support streaming of video and audio content using RTSP [RFC 2326], SDP [RFC 4566] and RTP [RFC 3550, RFC 3551].

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 data.

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.

A.5.2.2 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 shall contain an absolute NTP timestamp relative to Jan 1, 1900 with its corresponding RTP timestamp.

It is highly recommended that send a 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.

A.5.2.3 Use of RTSP and SDP

A.5.2.3.1 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>/Streaming/channels/ID?<parameters>
```

Where the path corresponds to the REST resource for a given streaming channel as defined in 7.11.3. The set of valid parameters corresponds to the query strings and their types in 7.11.3. 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/Streaming/channels/123456?videoCodecType=mjpeg&rotationDe
gree=90
```

A.5.2.3.2 Minimal implementation

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

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

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

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

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

A.5.2.3.3 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 shall 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.

A.5.2.3.4 Keep alive

The RTSP server shall 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 shall 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.

A.5.2.3.5 RTSP authentication

IP media device RTSP servers shall support HTTP basic authentication. It is highly recommended that RTSP servers support HTTP digest authentication as specified in 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.

A.5.2.4 RTP packetization rules for codecs

Rules for the description and packetization of codecs are 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 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
Audio			

G.726 [a]	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

a 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 shall 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 shall 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.

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

A.5.3 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 `/Streaming/channels/`*`ID`*`/http` in A.7.10.5 for a description of HTTP streaming.

A.6 Common data types

A.6.1 General

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.

Annotation	Description
`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.

A.6.2 Built-in types

Type	Description
BaudRate	A positive numerical value indicating the data transmission rate in symbols per second. Value is ≥ 0. Example: 9 600
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: 2 500 [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.

A.6.3 ReceiverAddress

```
<ReceiverAddress>
       <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>
</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`.

A.6.4 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>
```

```
                    <!-- opt, xs:integer, ISO8601 weekday number, 1=Monday, … -->
             </dayOfWeek>
             <TimeRange>                  <!-- opt -->
                    <beginTime>          <!--   req,   xs:time,   ISO8601   time   -->
       </beginTime>
                    <endTime>            <!--   req,   xs:time,   ISO8601   time   -->
       </endTime>
             </TimeRange>
             <bitString>           <!-- opt,  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 provide the same <dayOfWeek>.
- If the <bitString> tag in is provided, <TimeRange> is not 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 are 24 bits in length.

A.7 Service command details

A.7.1 /System

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

A.7.1.1 /System/reboot

URI	`/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.			

A.7.1.2 /System/updateFirmware

URI	`/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.			

A.7.1.3 /System/configurationData

URI	`/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-dependant – 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.		

A.7.1.4 /System/factoryReset

URI	`/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 `/System/Network` and `/System/Security`. The default mode is “full”. The device may be rebooted after it is reset.		

A.7.1.5 /System/deviceInfo

URI	`/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 RFC 1213. <deviceLocation> is the location of the device as defined in RFC 1213 <systemContact> is the contact information for the device as defined in RFC 1213. <systemObjectID> is the System Object Identifier defined in RFC 1213.		

A.7.1.5.1 DeviceInfo XML Block

```
<DeviceInfo version=“1.0” xmlns=“urn:psialliance-org”>
       <deviceName>                <!-- req, xs:string -->
       </deviceName>
       <deviceID>                         <!-- req, xs:string -->
       </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>
```

A.7.1.6 /System/supportReport

URI	`/System/supportReport`		**Type**	Resource
Function	This function is used to get a compressed archive of support information for the device. The archive shall 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.			

A.7.1.7 /System/status

URI	`/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.			

A.7.1.7.1 DeviceStatus XML Block

```
<DeviceStatus version="1.0" xmlns="urn:psialliance-org">
       <currentDeviceTime>  <!-- req, xs:datetime -->
       </currentDeviceTime>
       <deviceUpTime>                    <!--    req,    xs:integer,    seconds    -->
       </deviceUpTime>
       <TemperatureList>          <!-- req -->
              <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:int
-->    </pressure>
              </Pressure>
       </PressureList>
       <TamperList>              <!-- opt -->
              <Tamper>
                    <tamperSensorDescription>         <!--    req,    xs:string    -->
       </tamperSensorDescription>
                    <tamper>                                        <!--          req,
xs:boolean -->      </tamper>
              </Tamper>
       </TamperList>
       <CPUList>                        <!-- req -->
              <CPU>
                    <cpuDescription>     <!-- req, xs:string -->
       </cpuDescription>
                    <cpuUtilization>     <!--  req,  xs:integer,  percentage  0..100  -->
       </cpuUtilization>
              </CPU>
       </CPUList>
       <MemoryList>              <!-- req -->
              <Memory>
                    <memoryDescription>  <!-- req, xs:string -->
       </memoryDescription>
                    <memoryUsage>              <!--   req,   xs:float,   in   MB   -->
       </memoryUsage>
                    <memoryAvailable>          <!--    req,    xs:float,    in    MB-->
       </memoryAvailable>
              </Memory>
       </MemoryList>
       <openFileHandles>          <!-- opt, xs:integer -->    </openFileHandles>
</DeviceStatus>
```

A.7.1.8 /System/time

URI	/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 “local” 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.			

A.7.1.8.1 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>
```

A.7.1.9 /System/time/localTime

URI	/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.		

A.7.1.10 /System/time/timeZone

URI	`/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 Clause 8 of ISO/IEC 9945-1:2003 time zone notations. Note that the value following the +/- is the amount of time that shall 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.			

A.7.1.11 /System/time/ntpServers

URI	`/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.			

A.7.1.11.1 NtpServerList XML Block

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

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

URI	`/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 `/System/Network/interfaces/`*`ID`*`/ipAddress`.		

A.7.1.12.1 NtpServer XML Block

```
<NTPServer version="1.0" xmlns="urn:psialliance-org">
       <id>   <!-- req, xs:string;id -->         </id>
       <addressingFormatType>
              <!-- xs:string, "ipaddress,hostname" -->
```

A.7.1.13 </addressingFormatType>

```
       <hostName>                  <!-- dep, xs:string -->            </hostName>
       <ipAddress>                 <!-- dep, xs:string -->            </ipAddress>
       <ipv6Address>        <!-- dep, xs:string -->            </ipv6Address>
       <portNo>                    <!-- opt, xs:integer -->           </portNo>
</NTPServer>
```

A.7.1.14 /System/logging

URI	`/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.			

A.7.1.14.1 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>
```

A.7.1.15 /System/logging/messages

URI	`/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.			

A.7.1.15.1 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 -->
                     </eventID>
              <message>            <!-- req, xs:string -->
                     </message>
       </LogMessage>
</LogMessageList>
```

A.7.2 /System/Storage

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

A.7.2.1 /System/Storage/volumes

URI	`/System/Storage/volumes`		**Type**	Service
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.			

A.7.2.1.1 StorageVolumeList XML Block

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

A.7.2.2 /System/Storage/volumes/<ID>

URI	`/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.			

A.7.2.2.1 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>
```

A.7.2.3 /System/Storage/volumes/<ID>/status

URI	/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.			

A.7.2.3.1 StorageVolumeStatus XML Block

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

A.7.2.4 /System/Storage/volumes/<ID>/format

URI	/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.			

A.7.2.5 System/Storage/volumes/<ID>/files

URI	/System/Storage/volumes/*ID*/files		**Type**	Resource
Function	Get the list of files stored on a particular storage.			
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.			

A.7.2.5.1 StorageFileList XML Block

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

A.7.2.6 /System/Storage/volumes/<ID>/files/<ID>

URI	/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.			

A.7.2.6.1 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>
```

A.7.2.7 /System/Storage/volumes/<ID>/files/<ID>/data

URI	/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 Accept: http header to negotiate the data format.			

A.7.3 /System/Network

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

A.7.3.1 /System/Network/interfaces

URI	/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.			

A.7.3.1.1 NetworkInterfaceList XML Block

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

A.7.3.2 /System/Network/interfaces/<ID>

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

A.7.3.2.1 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>
```

A.7.3.3 /System/Network/interfaces/<ID>/ipAddress

URI	/System/Network/interfaces/*ID*/ipAddress		Type	Resource
Function				
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. <subnetMask> notation is "xxx.xxx.xxx.xxx". <IPV6Address> is "xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx" using CIDR notation.			

A.7.3.3.1 IPAddress XML Block

```
<IPAddress version="1.0" xmlns="urn:psialliance-org">
       <ipVersion>                <!-- req, xs:string, "v4,v6" -->   </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>
```

A.7.3.4 /System/Network/interfaces/<ID>/wireless

URI	`/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 shall be provided. If the <securityMode> field is "WPA” or “WPA2-personal", the <WPA> block shall be provided. If the "WPA” or “WPA2-enterprise" security mode is used, the <WPA> block shall be used and settings related to 802.1x shall be set using the `/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			

A.7.3.4.1 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:string,   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>
```

A.7.3.5 /System/Network/interfaces/<ID>/ieee802.1x

URI	/System/Network/interfaces/*ID*/ieee802.1x		Type	Resource
Function				
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 shall be provided. If the <authenticationProtocolType> corresponds to "EAP-PEAP" or "EAP-FAST", then the <innerEAPProtocolType> tag shall be provided. The <anonymousID> tag is optional. If the <authenticationProtocolType> corresponds to "EAP-FAST", then the <autoPACProvisioningEnabled> tag shall be provided. <anonymousID> is the optional anonymous ID to be used in place of the <userName>.			

A.7.3.5.1 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>
              <!-- req, xs:string, "MS-CHAP,MS-CHAPv2,PAP,EAP-MD5" -->
       </innerTTLSAuthenticationMethod>
       <innerEAPProtocolType>
              <!-- req, xs:string, "EAP-POTP,MS-CHAPv2" -->
       </innerEAPProtocolType>
       <validateServerEnabled>     <!-- req, xs:boolean -->    </validateServerEnabled>
       <userName>                  <!-- req, xs:string -->     </userName>
       <password>                  <!-- req, xs:string -->     </password>
       <anonymousID>        <!-- req, xs:string -->     </anonymousID>
       <autoPACProvisioningEnabled>      <!--        req,        xs:boolean        -->
       </autoPACProvisioningEnabled>
</IEEE802_1x>
```

A.7.3.6 /System/Network/interfaces/<ID>/ipFilter

URI	/System/Network/interfaces/*ID*/ipFilter		Type	Resource
Function				
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.		

A.7.3.6.1 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>
```

A.7.3.7 /System/Network/interfaces/<ID>/ipFilter/filterAddresses

URI	/System/Network/interfaces/*ID*/ipFilter/filterAddresses		**Type**	Resource
Function				
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.			

A.7.3.7.1 IPFilterAddressList XML Block

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

A.7.3.8 /System/Network/interfaces/<ID>/ipFilter/filterAddresses/<ID>

URI	/System/Network/interfaces/*ID*/ipFilter/filterAddresses/ *ID*		**Type**	Resource
Function				
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
GET			<IPFilterAddress>	
PUT		<IPFilterAddress>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	If the <permissionType> tag is not provided as a direct child of <IPFilter>, the <permissionType> tag shall 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 shall be provided in place of the <AddressRange> block. If it refers to "range", the <Range> block shall be provided in place of the <AddressMask> block. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in			

	/System/Network/interfaces/ID/ipAddress.

A.7.3.8.1 IPFilanterAddress XML block

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

A.7.3.9 /System/Network/interfaces/<ID>/snmp

URI	/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 shall be provided.			

A.7.3.9.1 SNMP XML block

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

A.7.3.10 /System/Network/interfaces/<ID>/snmp/v2c

URI	/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

A.7.3.10.1 SNMPv2c XML Block

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

A.7.3.11 /System/Network/interfaces/<ID>/snmp/v2c/trapReceivers

URI	/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.			

A.7.3.11.1 SNMPTrapReceiverList XML Block

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

A.7.3.12 /System/Network/interfaces/<ID>/snmp/v2c/trapReceivers/<ID>

URI	/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> the format shall conform to the SNMPv2c standard.			

A.7.3.12.1 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>
```

A.7.3.13 /System/Network/interfaces/<ID>/snmp/advanced

URI	/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.		

A.7.3.13.1 SNMPadvanced XML block

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

A.7.3.14 /System/Network/interfaces/<ID>/snmp/advanced/users

URI	/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.		

A.7.3.14.1 SNMPUserList XML Block

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

A.7.3.15 /System/Network/interfaces/<ID>/snmp/advanced/users/<ID>

URI	/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>.

A.7.3.15.1 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:string -->            </remoteEngineID>
       <snmpAuthenticationMethod>
              <!-- req, xs:string, "MD5,SHA,none" -->
       </snmpAuthenticationMethod>
       <snmpAuthenticationKey>     <!-- req, xs:string -->
       </snmpAuthenticationKey>
       <snmpAuthenticationPassword>
              <!-- req, xs:string, see RFC3414 -->
       </snmpAuthenticationPassword>
       <snmpPrivacyMethod>         <!--      req,      xs:string,      "DES,none"      -->
       </snmpPrivacyMethod>
       <snmpPrivacyKey>                   <!-- req, xs:string -->
       </snmpPrivacyKey>
       <snmpPrivacyPassword>              <!--   req,   xs:string,   see   RFC3414   -->
       </snmpPrivacyPassword>
</SNMPUser>
```

A.7.3.16 /System/Network/interfaces/<ID>/snmp/advanced/notificationFilters

URI	/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.			

A.7.3.16.1 SNMPNotificationFilterList XML Block

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

A.7.3.17 /System/Network/interfaces/<ID>/snmp/advanced/notificationFilters/<ID>

URI	/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.		

A.7.3.17.1 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>                           <!-- opt -->
                     <oidSubtree>        <!-- req, xs:string -->
                            </oidSubtree>
                     <filterAction>             <!--          req,          xs:string,
"included,excluded" -->           </filterAction>
              </OID>
       </OIDSubtreeList>
</SNMPNotificationFilter>
```

A.7.3.18 /System/Network/interfaces/<ID>/snmp/advanced/notificationReceivers

URI	/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.		

A.7.3.18.1 SNMPNotificationReceiverList XML block

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

A.7.3.19 /System/Network/interfaces/<ID>/snmp/advanced/notificationReceivers/<ID>

URI	/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> shall correspond to a user ID in /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.			

A.7.3.19.1 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>                      <!-- opt -->
              <!-- req, xs:string, "noauthentication,authentication,privacy" -->
       </securityType>
       <filterEnabled>       <!-- req, xs:boolean -->
       </filterEnabled>
       <filterName>          <!-- req, xs:integer -->
       </filterName>
       <timeout>                    <!-- req, xs:integer, seconds -->
       </timeout>
       <retries>                    <!-- req, xs:integer -->
       </retries>
</SNMPNotificationReceiver>
```

A.7.3.20 /System/Network/interfaces/<ID>/snmp/v3

URI	/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 /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 RFC 3414). 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.			

A.7.3.21 /System/Network/interfaces/<ID>/qos

URI	/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> shall be provided.			

A.7.3.21.1 QoS XML block

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

A.7.3.22 /System/Network/interfaces/<ID>/qos/cos

URI	/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.			

A.7.3.22.1 CoSList XML block

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

A.7.3.23 /System/Network/interfaces/<ID>/qos/cos/<ID>

URI	/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.			

A.7.3.23.1 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>
```

A.7.3.24 /System/Network/interfaces/<ID>/qos/dscp

URI	/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.			

A.7.3.24.1 DSCPList XML block

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

A.7.3.25 /System/Network/interfaces/<ID>/qos/dscp/<ID>

URI	/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.			

A.7.3.25.1 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>
```

A.7.3.26 /System/Network/interfaces/<ID>/discovery

URI	/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 /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.			

A.7.3.26.1 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>         <!-- req, xs:string -->     </ipAddress>
              <ipv6Address> <!-- req, xs:string -->     </ipv6Address>
              <portNo>            <!-- req, xs:integer -->    </portNo>
              <ttl>               <!-- req, xs:integer -->    </ttl>
       </MulticastDiscovery>
</Discovery>
```

A.7.3.27 /System/Network/interfaces/<ID>/syslog

URI	/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.			

A.7.3.27.1 Syslog XML block

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

A.7.3.28 /System/Network/interfaces/<ID>/syslog/servers

URI	/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.			

A.7.3.28.1 SyslogServerList XML block

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

A.7.3.29 /System/Network/interfaces/<ID>/syslog/servers/<ID>

URI	`/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 `/System/Network/interfaces/ID/ipAddress`. <facilityType> indicates the facility to store syslog messages. See RFC 3164. <severity> indicates the minimum log severity for which to send a syslog message. See RFC 3164.			

A.7.3.29.1 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>           <!-- req, xs:string -->            </hostName>
       <ipAddress>          <!-- req, xs:string -->            </ipAddress>
       <ipv6Address> <!-- req, xs:string -->            </ipv6Address>
       <portNo>             <!-- req, xs:integer -->           </portNo>
       <facilityType>       <!-- req, xs:string, see RFC3164 -->
       </facilityType>
       <severity>           <!-- req, xs:string, see RFC3164 -->              </severity>
</SyslogServer>
```

A.7.3.30 Examples

A.7.3.30.1 Example: getting the network settings

```
GET /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>
```

A.7.3.30.2 Example: setting the IP address

```
PUT /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>
```

A.7.3.31 /System/IO

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

A.7.3.31.1 IOPortList XML block

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

A.7.3.32 /System/IO/status

URI	/System/IO/status		**Type**	Resource
Function				
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	

GET			<IOPortStatusList>
Notes	<ioPortID> refers to `/System/IO/inputs/ID` or `/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.		

A.7.3.32.1 IOPortStatus XML block

```
<IOPortStatusList version="1.0" xmlns="urn:psialliance-org">
       <IOPortStatus>             <!-- req -->
              <ioPortID>          <!-- req, xs:string;id -->
                     </ioPortID>
              <ioPortType>  <!-- req, xs:string, "input,output" -->
       </ioPortType>
              <ioState>           <!-- req, xs:string, "active,inactive" -->
       </ioState>
       </IOPortStatus>
</IOPortStatusList>
```

A.7.3.33 /System/IO/inputs

URI	`/System/IO/inputs`	**Type**	Resource
Function			
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.		

A.7.3.33.1 IOInputPortList XML Block

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

A.7.3.34 /System/IO/inputs/<ID>

URI	`/System/IO/inputs/`*`ID`*	**Type**	Resource
Function			
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.		

A.7.3.34.1 IOInputPort XML block

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

A.7.3.35 /System/IO/inputs/<ID>/status

URI	`/System/IO/inputs/`*`ID`*`/status`	**Type**	Resource

Function			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**
GET			<IOPortStatus>
Notes	See `/System/IO/status` for an explanation of the fields.		

A.7.3.36 /System/IO/outputs

URI	`/System/IO/outputs`	**Type**	Resource
Function			
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.		

A.7.3.36.1 IOOutputPortList XML block

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

A.7.3.37 /System/IO/outputs/<ID>

URI	`/System/IO/outputs/`*`ID`*	**Type**	Resource
Function			
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 shall be provided if the <outputState> is “pulse”. <actionMapping> is used in interfaces that allow configuration of “On” and “Off” for "High" and "Low" signals.		

A.7.3.37.1 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>     <!-- opt, xs:integer, milliseconds -->
       </pulseDuration>
       </PowerOnState>
       <ManualControl>            <!-- opt -->
              <actionMapping>
                     <!-- req, xs:string, "high,low": ON maps to high / ON maps to low
-->
              </actionMapping>
       </ManualControl>
```

```
</IOOutputPort>
```

A.7.3.38 /System/IO/outputs/<ID>/trigger

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

A.7.3.38.1 IOPortData XML block

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

A.7.3.39 /System/IO/outputs/<ID>/status

URI	`/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 `/System/IO/status` for an explanation of the fields.			

A.7.3.40 IO port examples

A.7.3.40.1 Example: set up IO port triggering

NOTE The following example shows that input port event detection and output port triggering be enabled and scheduled with `/Custom/Event/triggers` and `/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 A.7.13.17) 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 A.7.13.17).
- 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 three seconds and then switch back to a low signal.

```
PUT /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 /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 /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>
```

A.7.3.40.2 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 behaviour set here.

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

A.7.4 /System/Audio

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

A.7.4.1 /System/Audio/channels

URI	/System/Audio/channels		**Type**	Resource
Function				
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.

A.7.4.1.1 AudioChannelList XML block

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

A.7.4.2 /System/Audio/channels/<ID>

URI	/System/Audio/channels/*ID*		**Type**	Resource
Function				
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.			

A.7.4.2.1 AudioChannel XML block

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

A.7.5 /System/Video

URI	/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.			

A.7.5.1 /System/Video/overlayImages

URI	/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 shall set the HTTP `Content-Type:` header field to the correct MIME type for the image.		

A.7.5.1.1 ImageOverlayList XML block

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

A.7.5.2 /System/Video/overlayImages/<ID>

URI	`/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>
DELETE			<ResponseStatus>
Notes	Overlay images can be updated using the PUT command and deleted using the DELETE command.		

A.7.5.3 /System/Video/inputs

URI	`/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.		

A.7.5.3.1 VideoInput XML Block

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

A.7.5.4 /System/Video/inputs/channels

URI	`/System/Video/inputs/channels`	**Type**	Resource
Function			
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.

A.7.5.4.1 VideoInputChannelList XML block

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

A.7.5.5 /System/Video/inputs/channels/<ID>

URI	/System/Video/inputs/channels/*ID*		**Type**	Resource
Function	Set 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.			

A.7.5.5.1 VideoInputChannel XML block

```
<VideoInputChannel version="1.0" xmlns="urn:psialliance-org">
       <id>                                      <!-- req, xs:string -->
              </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>        <!-- opt, 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>
                     <!-- opt, xs:string, "day,night,auto" -->
              </dayNightFilterType>
              <switchScheduleEnabled><!-- opt, xs:boolean -->
       </switchScheduleEnabled>
              <beginTime>                        <!-- dep, xs:time -->
                     </beginTime>
              <endTime>                          <!-- dep, xs:time -->
                     </endTime>
       </DayNightFilter>
<VideoInputChannel>
```

A.7.5.6 /System/Video/inputs/channels/<ID>/focus

URI	/System/Video/inputs/channels/*ID*/lens		Type	Resource
Function				
Methods	Query String(s)	Inbound Data	Return Result	

PUT	focus	<FocusData>	<ResponseStatus>
Notes	<focus>: focus vector data. Negative numbers focus out, positive numbers focus in. Numerical value is a percentage of the maximum focus speed of the lens module.		

A.7.5.6.1 FocusData XML block

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

A.7.5.7 /System/Video/inputs/channels/<ID>/iris

URI	/System/Video/inputs/channels/*ID*/iris	**Type**	Resource
Function			
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.		

A.7.5.7.1 IrisData XML block

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

A.7.5.8 /System/Video/inputs/channels/<ID>/lens

URI	/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 open, 100 is completely open.		

A.7.5.8.1 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>
```

A.7.5.9 /System/Video/inputs/channels/<ID>/overlays

URI	/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>
POST		<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 `/System/Video/overlayImages`.

A.7.5.9.1 VideoOverlay XML block

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

A.7.5.10 /System/Video/inputs/channels/<ID>/overlays/text

URI	`/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.			

A.7.5.10.1 TextOverlayList XML block

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

A.7.5.11 /System/Video/inputs/channels/<ID>/overlays/text/<ID>

URI	`/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.			

A.7.5.11.1 TextOverlay XML block

```
<TextOverlay version="1.0" xmlns="urn:psialliance-org">
       <id>                                <!-- req, xs:string;id -->
       </id>
       <enabled>                           <!-- req, xs:boolean -->
       </enabled>
       <timeStampEnabled>           <!-- req, xs:boolean -->
       </timeStampEnabled>
       <dateTimeFormat>             <!-- req, xs:string -->
       </dateTimeFormat>
       <backgroundColor>            <!-- req, xs:string;color -->
       </backgroundColor>
       <fontColor>                         <!-- req, xs:string;color -->
       </fontColor>
       <fontSize>                          <!-- req, 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>
```

A.7.5.12 /System/Video/inputs/channels/<ID>/overlays/image

URI	/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.			

A.7.5.12.1 ImageOverlayList XML Block

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

A.7.5.13 /System/Video/inputs/channels/<ID>/overlays/image/<ID>

URI	/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 shall have been previously uploaded to the device using the `/System/Video/overlayImages` command.			

A.7.5.13.1 ImageOverlay XML block

```
<ImageOverlay version=“1.0” xmlns=“urn:psialliance-org”>
       <id>                                  <!-- req, xs:string;id -->
              </id>
       <enabled>                             <!-- req, xs:boolean -->
              </enabled>
       <overlayImageID>               <!-- req, xs:string;id -->
       </overlayImageID>
       <positionX>                           <!--   req,   xs:integer;coordinate   -->
       </positionX>
       <positionY>                           <!--   req,   xs:integer;coordinate   -->
       </positionY>
       <transparentColorEnabled>   <!-- req, xs:boolean -->
       </transparentColorEnabled>
       <transparentColor>             <!-- req, xs:string;color -->
       </transparentColor>
</ImageOverlay>
```

A.7.5.14 /System/Video/inputs/channels/<ID>/privacyMask

URI	`/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.			

A.7.5.14.1 PrivacyMask XML block

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

A.7.5.15 /System/Video/inputs/channels/<ID>/privacyMask/regions

URI	`/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.			

A.7.5.15.1 PrivacyMaskRegionList XML block

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

A.7.5.16 /System/Video/inputs/channels/<ID>/privacyMask/regions/<ID>

URI	`/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 shall be provided for a single <PrivacyMaskRegion> block. Ordering of <PrivacyMaskRegion> blocks is insignificant.			

A.7.5.16.1 PrivacyMaskRegion XML block

```
<PrivacyMaskRegion version="1.0" xmlns="urn:psialliance-org">
       <id>                                        <!-- req, xs:string -->
                     </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>
```

A.7.6 /System/Serial

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

A.7.6.1 /System/Serial/ports

URI	/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.			

A.7.6.1.1 SerialPortList XML Block

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

A.7.6.2 /System/Serial/ports/<ID>

URI	/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.			

A.7.6.2.1 SerialPort XML block

```
<SerialPort version="1.0" xmlns="urn:psialliance-org">
       <id>                        <!-- req, xs:string -->
                           </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>
```

A.7.6.3 /System/Serial/ports/<ID>/command

URI	/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.			

A.7.6.3.1 SerialCommand XML block

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

A.7.6.3.2 Example

Send the command using an XML block:

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

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

A.7.7 /Diagnostics

A.7.7.1 /Diagnostics/commands

URI	/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: /Diagnostics/commands/*ID*. <resultType> 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.			

A.7.7.2 /Diagnostics/commands/<ID>

URI	/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.			

A.7.7.3 Diagnostics XML Data

A.7.7.3.1 DiagnosticCommandList XML block

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

A.7.7.3.2 DiagnosticCommand XML block

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

A.7.8 /Security

URI	/Security	**Type**	Service

Methods	**Query String(s)**	**Inbound Data**	**Return Result**
Notes	Security service.		

A.7.8.1 /Security/srtpMasterKey

URI	`/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 RFC 3711 for a description of SRTP.			

A.7.8.2 /Security/deviceCertificate

URI	`/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.			

A.7.9 /Security/AAA

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

A.7.9.1 /Security/AAA/users

URI	`/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.			

A.7.9.1.1 UserList XML block

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

A.7.9.2 /Security/AAA/users/<ID>

URI	`/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, shall match a corresponding <id> tag in `/Security/adminAccesses`. **Note:** <password> is a write-only field.		

A.7.9.2.1 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>
```

A.7.9.3 /Security/AAA/certificate

URI	`/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.		

A.7.9.4 /Security/AAA/adminAccesses

URI	`/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.		

A.7.9.4.1 AdminAccessProtocolList XML Block

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

A.7.9.5 /Security/AAA/adminAccesses/<ID>

URI	`/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.

A.7.9.5.1 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>
```

A.7.10 /Streaming

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

A.7.10.1 /Streaming/status

URI	/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.			

A.7.10.1.1 StreamingStatus XML block

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

A.7.10.2 /Streaming/Channels

URI	/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 `/Streaming/channels/description`.			

A.7.10.2.1 StreamingChannelList XMl Block

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

A.7.10.3 /Streaming/Channels/<ID>

URI	/Streaming/channels/*ID*		Type	Resource
Function				
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 /System/Video/inputs/channel/*ID*. <audioInputChannelID> refers to /System/Audio/channels/*ID*. It shall be configured as an input channel. Use of IPv4 or IPv6 addresses depends on the value of the <ipVersion> field in /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.			

A.7.10.3.1 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>         <!-- opt -->
                            <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>             <!-- opt, xs:string -->
      </destIPAddress>
                    <videoDestPortNo>           <!-- opt, xs:integer -->
      </videoDestPortNo>
                    <audioDestPortNo>           <!-- opt, xs:integer -->
      </audioDestPortNo>
                    <destIPv6Address>           <!-- opt, xs:string -->
      </destIPv6Address>
                    <ttl>                               <!-- opt, xs:integer -->
             </ttl>
             </Multicast>
             <Security>                                 <!-- opt -->
                    <enabled>                           <!-- req, xs:boolean -->
             </enabled>
             </Security>
      </Transport>
      <Video>
             <enabled>                                  <!-- req, xs:boolean -->
             </enabled>
             <videoInputChannelID>              <!-- req, xs:string;id -->
      </videoInputChannelID>
             <videoCodecType>
                    <!-- opt, xs:string, “MPEG4,MJPEG,3GP,H.264,MPNG” -->
             </videoCodecType>
             <videoScanType>
                    <!-- opt, xs:string, “progressive,interlaced” -->
             </videoScanType>
             <videoResolutionWidth>             <!-- opt, xs:integer -->
      </videoResolutionWidth>
             <videoResolutionHeight>     <!-- opt, xs:integer -->
      </videoResolutionHeight>
             <videoPositionX>                   <!-- opt, xs:integer -->
      </videoPositionX>
             <videoPositionY>                   <!-- opt, xs:integer -->
      </videoPositionY>
             <videoQualityControlType>
                    <!-- req, xs:string, “CBR,VBR” -->
             </videoQualityControlType>
             <constantBitRate>    <!-- opt, xs:integer, in kbps -->
      </constantBitRate>
             <fixedQuality>              <!-- opt, xs:integer, percentage, 0..100 -->
             </fixedQuality>
             <vbrUpperCap>        <!-- opt, xs:integer, in kbps -->
      </vbrUpperCap>
             <vbrLowerCap>        <!-- opt, 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>
```

```
              <enabled>                                 <!-- req, xs:boolean -->
              </enabled>
              <audioInputChannelID>             <!-- req, xs:string;id -->
       </audioInputChannelID>
              <audioCompressionType>
                     <!-- opt, 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>
```

A.7.10.3.2 Example: getting streaming channel properties

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

```
GET /Streaming/channels/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>
                            <streamingTransport>HTTP</streamingTransport>
                     </ControlProtocol>
              </ControlProtocolList>
       </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>
```

A.7.10.3.3 Example: getting streaming capabilities

```
GET /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>
                            <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>
              <audioInboundCompressionType/>     <!-- not used by this implementation -
->
              <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>
```

A.7.10.3.4 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 /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>
                            <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>
```

A.7.10.4 /Streaming/Channels/<ID>/status

URI	`/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 `/System/Network/interfaces/ID/ipAddress`.			

A.7.10.4.1 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>
```

A.7.10.5 /Streaming/Channels/<ID>/http

URI	/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 shall 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 `/Streaming/channels/ID/http/capabilities`.			

A.7.10.5.1 Example

```
GET /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>
…
```

A.7.10.6 /Streaming/Channels/<ID>/picture

URI	`/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 shall 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 /Streaming/channels/*ID*/picture/capabilities. Examples: GET /Streaming/channels/123456/picture?snapShotImageType=JPEG POST /Streaming/channels/123456/picture … <?xml version="1.0" encoding="UTF-8"?> <Video>…</Video> GET /Streaming/channels/123456/picture Accept: image/jpeg			

A.7.10.7 /Streaming/channels/<ID>/requestKeyFrame

URI	`/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.			

A.7.11 /PTZ

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

A.7.11.1 /PTZ/channels

URI	/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 /PTZ/channels/description.			

A.7.11.1.1 PTZChannelList XML Block

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

A.7.11.2 /PTZ/channels/<ID>

URI	/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.			

A.7.11.2.1 PTZChannel XML Block

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

A.7.11.3 /PTZ/channels/<ID>/homePosition

URI	/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	<Absolute>: <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. <Momentary>: <duration> indicates the duration of the pan/tilt/zoom action for momentary PTZ. <Relative>: <positionX>: the horizontal coordinate of the current video image for which the screen will be centered on. <positionY>: the vertical coordinate of the current video image for which the screen will be centered on. <relativeZoom> Indicates the relative percentage to zoom the lens module: 0 is the current zoom position. Negative numbers zoom out, positive numbers zoom in. The numerical value indicates roughly what percentage to zoom in respect to the current image. <Absolute>: <elevation> indicates the elevation (degrees) in respect to the absolute home position for which to tilt the device Negative numbers are down, positive numbers are up. <azimuth> indicates the angle (degrees) in respect to the absolute home position for which to pan the device Degrees move clockwise. e.g. If 0 is due North, 90 is East, 180 is South, 270 is West. <absoluteZoom> indicates the absolute percentage to zoom the lens module. <Digital>: <positionX>: The horizontal coordinate of the video image for which the screen will be centered on. This value is based on the "un-zoomed", full resolution of the image. <positionY>: The vertical coordinate of the video image for which the screen will be centered on. This value is based on the "un-zoomed", full resolution of the image. <digitalZoomLevel>: The digital zoom level for digital PTZ 0 is no zoom, 100 is maximum zoom.			

A.7.11.3.1 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>            <!-- opt, xs:integer, milliseconds -->    </duration>
       </Momentary>
       <Relative>
              <positionX>           <!-- opt, xs:integer -->
       </positionX>
              <positionY>           <!-- opt, xs:integer -->
       </positionY>
              <relativeZoom>        <!-- opt, xs:integer, -100..100 -->
       </relativeZoom>
       </Relative>
       <Absolute>
              <elevation>          <!-- opt, xs:integer, -90..90 -->
       </elevation>
              <azimuth>            <!-- opt, xs:integer, 0..360 -->
       </azimuth>
              <absoluteZoom>       <!-- opt, xs:integer, 0..100 -->
       </absoluteZoom>
       </Absolute>
       <Digital>
              <positionX>                         <!-- opt, xs:integer -->
       </positionX>
              <positionY>                         <!-- opt, xs:integer -->
       </positionY>
              <digitalZoomLevel>          <!--     opt,     xs:integer,     0..100     -->
       </digitalZoomLevel>
       </Digital>
</PTZData>
```

A.7.11.4 /PTZ/channels/<ID>/continuous

URI	`/PTZ/channels/`*`ID`*`/continuous`		**Type**	Resource
Function	Set the home position of the PTZ camera to the current position			
Methods	**Query String(s)**	**Inbound Data**	**Return Result**	
PUT	pan tilt zoom	<PTZData>	<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.			

A.7.11.5 /PTZ/channels/<ID>/momentary

URI	`/PTZ/channels/`*`ID`*`/momentary`		**Type**	Resource
Function	Set the home position of the PTZ camera to the current position			
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. 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.			

A.7.11.6 /PTZ/channels/<ID>/relative

URI	`/PTZ/channels/`*`ID`*`/relative`		**Type**	Resource
Function	Set the home position of the PTZ camera 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. The <positionX> and <positionY> tags shall 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.

A.7.11.7 /PTZ/channels/<ID>/absolute

URI	`/PTZ/channels/`*`ID`*`/absolute`		**Type**	Resource
Function	Set the home position of the PTZ camera to the current 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. All parameters in the <Absolute> block shall 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.			

A.7.11.8 /PTZ/channels/<ID>/digital

URI	`/PTZ/channels/`*`ID`*`/digital`		**Type**	Resource
Function	Set the home position of the PTZ camera to the current position			
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. 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 shall be provided with a value greater than 0 (meaning the image shall be zoomed). The <positionX> and <positionY> tags, if provided, shall 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, shall 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.			

A.7.11.9 /PTZ/channels/<ID>/status

URI	`/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.

A.7.11.9.1 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>
```

A.7.11.10 /PTZ/channels/<ID>/presets

URI	`/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	Any currently running/paused patrols shall 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>
```

A.7.11.11 /PTZ/channels/<ID>/presets/<ID>

URI	`/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	A single preset corresponding to the current position is added to the repository. The <presetName> tag shall be unique across the entire device. Any currently running/paused patrols with modified presets shall 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".			

A.7.11.11.1 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:string;color -->
              </backgroundColor>
                     <fontColor>                       <!--  opt,  xs:string;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>
```

A.7.11.12 /PTZ/channels/<ID>/presets/<ID>/goto

URI	`/PTZ/channels/ID/presets/ID/goto`		**Type**	Resource
Function	Goto 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.			

A.7.11.13 /PTZ/channels/<ID>/patrols

URI	`/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.			

A.7.11.13.1 PTZPatrolList XML block

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

A.7.11.14 /PTZ/channels/<ID>/patrols/status

URI	/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 /PTZ/channels/*ID*/patrols/*ID*.			

A.7.11.14.1 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>
```

A.7.11.15 /PTZ/channels/<ID>/patrols/<ID>

URI	/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> shall correspond to a valid ID in /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.			

A.7.11.15.1 PTZPatrol XML block

```
<PTZPatrol version="1.0" xmlns="urn:psialliance-org">
       <id>                               <!-- req, xs:string;id -->
                     </id>
       <patrolName>                <!-- req, xs:string -->
              </patrolName>
       <resumeType>                <!-- req, 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>
```

A.7.11.16 /PTZ/channels/<ID>/patrols/<ID>/start

URI	`/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.			

A.7.11.17 /PTZ/channels/<ID>/patrols/<ID>/stop

URI	`/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(s) is stopped if it is currently running or paused.			

A.7.11.18 /PTZ/channels/<ID>/patrols/<ID>/pause

URI	`/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 `/PTZ/channels/`*`ID`*`/patrols/`*`ID`*`/start`.			

A.7.11.19 /PTZ/channels/<ID>/patrols/<ID>/status

URI	`/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.			

A.7.11.20 /PTZ/channels/<ID>/patrols/<ID>/schedule

URI	`/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.			

A.7.11.21 Patrol Examples

A.7.11.21.1 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 /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 s, “Gate” @ 10 s, “Alley” @ 3 s, and repeat:

```
POST /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 s, “East entrance” @ 7 s, “Alley” @ 7 s, and repeat.

```
POST /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>
```

A.7.11.21.2 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 /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 /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>
```

A.7.12 /Custom/MotionDetection

URI	/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 shall correspond to the video input channel ID.			

A.7.12.1 MotionDetectionList XML Block

```
<MotionDetectionList version=“1.0” xmlns=“urn:psialliance-org”>
        <MotionDetection/>          <!-- opt -->
</MotionDetectionList>
```

A.7.12.2 /Custom /motionDetection/<ID>

URI	/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 SHALL 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 `/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.

A.7.12.2.1 MotionDetection XML block

```
<MotionDetection version="1.0" xmlns="urn:psialliance-org">
       <id>                                <!-- req, xs:string -->
                     </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>
```

A.7.12.3 /Custom/MotionDetection/<ID>/regions

URI	`/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.			

A.7.12.3.1 MotionDetectionRegionList XML block

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

A.7.12.4 /Custom/MotionDetection/<ID>regions/<ID>

URI	/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>.			

A.7.12.4.1 MotionDetectionRegion XML block

```
<MotionDetectionRegion version="1.0" xmlns="urn:psialliance-org">
       <id>                            <!-- req, xs:string -->     </id>
       <enabled>                       <!-- req, xs:boolean -->    </enabled>
       <maskEnabled>            <!-- req, 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>
```

A.7.12.5 Motion detection example

A.7.12.5.1 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 does not 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 shall be detected for at least one second before triggering an event notification (motion shall be stopped for at least one second to stop the triggering).
- As shown in Figure A.1, 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.

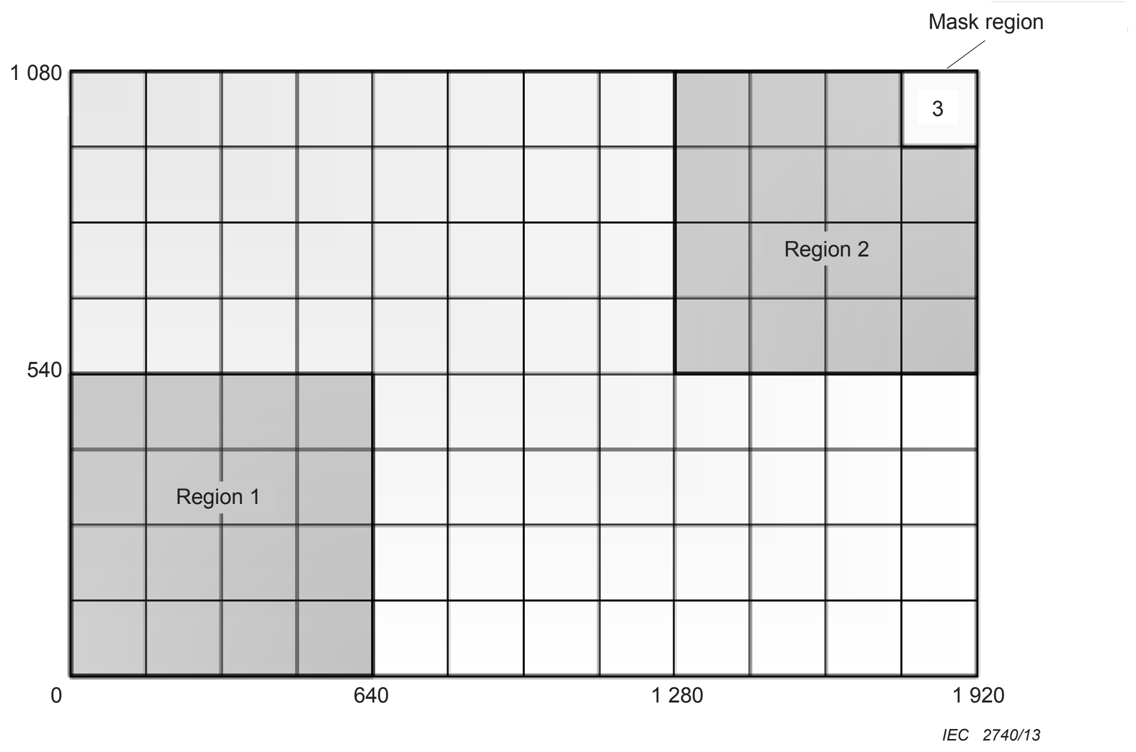

Figure A.1 – Motion detection grid with two detection regions

```
PUT /Custom/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>
```

A.7.13 /Custom/Event

URI	/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.			

A.7.13.1 EventNotification XML block

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

```
       <EventTriggerList/>                <!-- opt -->
       <EventSchedule/>                          <!-- opt -->
       <EventNotificationMethods/> <!-- opt -->
</EventNotification>
```

A.7.13.2 /Custom/Event/triggers

URI	`/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.			

A.7.13.2.1 EventTriggerList XML block

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

A.7.13.3 /Custom/Event/triggers/<ID>

URI	`/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 `/Custom/Event/triggers`. <inputIOPortID> is only required if <eventType> is "IO".			

A.7.13.3.1 EventTrigger XML block

```
<EventTrigger version="1.0" xmlns="urn:psialliance-org">
       <id>                                      <!-- req, xs:string -->
              </id>
       <eventType>                               <!-- req -->
```

```
              <!-- req, xs:string,
                     "IO,VMD,videoloss,diskfailure,recordingfailure,
                      badvideo,POS,analytics,fanfailure,overheat"
              -->
       </eventType>
       <eventDescription>                 <!-- req, xs:string -->
       </eventDescription>
       <inputIOPortID>                    <!-- req, xs:string -->
       </inputIOPortID>
       <intervalBetweenEvents>     <!-- req, xs:integer, seconds -->
       </intervalBetweenEvents>
<EventTriggerNotificationList/>    <!-- opt -->
</EventTrigger>
```

A.7.13.4 /Custom/Event/triggers/<ID>/notifications

URI	/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 subclause determines the kinds of notifications that are supported for a particular event trigger and their recurrences and behaviors.			

A.7.13.4.1 EventTriggerNotificationList XML block

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

A.7.13.5 /Custom/Event/triggers/<ID>/notifications/<ID>

URI	/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".			

A.7.13.5.1 EventTriggerNotification XML block

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

A.7.13.6 /Custom/Event/schedule

URI	/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.			

A.7.13.6.1 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>
```

A.7.13.7 /Custom/Event/notification

URI	/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.			

A.7.13.7.1 EventNotificationMethods XML block

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

```
                <attachedSnapShotEnabled>   <!--      req,      xs:boolean      -->
       </attachedSnapShotEnabled>
                <attachedVideoClipEnabled>  <!--      req,      xs:boolean      -->
       </attachedVideoClipEnabled>
              </BodySetting>
       </EmailFormat>
       <MediaFormat>                        <!-- opt -->
              <snapShotImageType>  <!-- req, xs:string -->     </snapShotImageType>
              <videoClipFormatType>       <!--         req,         xs:string         -->
       </videoClipFormatType>
              <preCaptureLength>          <!--   req,   xs:integer,   milliseconds   -->
       </preCaptureLength>
              <postCaptureLength>  <!--     req,     xs:integer,     milliseconds     -->
       </postCaptureLength>
       </MediaFormat>
</EvenNotificationMethods>
```

A.7.13.8 /Custom/Event/notification/mailing

URI	/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.			

A.7.13.8.1 MailingNotificationList XML block

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

A.7.13.9 /Custom/Event/notification/mailing/<ID>

URI	/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			

A.7.13.9.1 MailingNotification XML block

```
<MailingNotification version="1.0" xmlns="urn:psialliance-org">
       <id>                               <!-- req, xs:string -->            </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>
              <!-- opt, xs:string, "ipaddress,hostname" -->
       </popAddressingFormatType>
       <popServerHostName>  <!-- opt, xs:string -->     </popServerHostName>
       <popServerIPAddress> <!-- opt, xs:string -->     </popServerIPAddress>
       <popServerIPv6Address>      <!-- opt, xs:string -->     </popServerIPv6Address>
       <accountName>              <!-- req, xs:string -->     </accountName>
       <password>                        <!-- req, xs:string -->     </password>
</MailingNotification>
```

A.7.13.10 /Custom/Event/notification/ftp

URI	/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.			

A.7.13.10.1 FTPNotificationList XML block

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

A.7.13.11 /Custom/Event/notification/ftp/<ID>

URI	/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.			

A.7.13.11.1 FTPNotification XML block

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

A.7.13.12 /Custom/Event/notification/httpHost

URI	/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.			

A.7.13.12.1 HttpHostNotificationList XML block

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

A.7.13.13 /Custom/Event/notification/httpHost/<ID>

URI	/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.			

A.7.13.13.1 HttpHostNotification XML block

```
<HttpHostNotification version="1.0" xmlns="urn:psialliance-org">
       <id>                        <!-- req, xs:string -->
       </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>                  <!-- req, xs:string -->
        </hostName>
        <ipAddress>                 <!-- dep, xs:string -->
        </ipAddress>
        <ipv6Address>        <!-- dep, xs:string -->
        </ipv6Address>
        <portNo>                    <!-- req, xs:integer -->
        </portNo>
        <userName>                  <!-- req, xs:string -->
        </userName>
        <password>                  <!-- req, xs:string -->
        </password>
        <httpAuthenticationMethod>
              <!-- req, xs:string, "MD5digest,none" -->
        </httpAuthenticationMethod>
</HttpHostNotification>
```

A.7.13.14 /Custom/Event/notification/alertStream

URI	/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.			

A.7.13.14.1 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 -->           </protocol>
        <macAddress>         <!-- opt, xs:string;MAC --> </macAddress>
        <channelID>                 <!-- dep, xs:string -->           </channelID>
        <dateTime>                  <!-- req, xs:datetime -->         </dateTime>
        <activePostCount>    <!-- req, xs:integer -->          </activePostCount>
        <eventType>
              <!-- req, xs:string,
                    "IO,VMD,videoloss,raidfailure,recordingfailure,
                     badvideo,POS,analytics,fanfailure,overheat"
              -->
        </eventType>
```

```
       <eventState>         <!--      req,      xs:string,      “active,inactive”      -->
       </eventState>
       <eventDescription>   <!-- req, xs:string -->
       </eventDescription>
       <inputIOPortID>      <!-- dep, xs:string, if <eventType> is “IO” -->
       </inputIOPortID>
       <DetectionRegionList>                    <!-- dep, if <eventType> is “VMD” -->
              <DetectionRegionEntry>            <!-- req -->
                     <regionID>                        <!-- req, xs:string -->
              </regionID>
                     <sensitivityLevel>         <!--   req,   xs:integer,   0..100   -->
       </sensitivityLevel>
                     <detectionThreshold> <!--     req,     xs:integer,     0..100     -->
       </detectionThreshold>
                     <detectionLevel>           <!--   req,   xs:integer,   0..100   -->
       </detectionLevel>
              </DetectionRegionEntry>
       </DetectionRegionList>
</EventNotificationAlert>
```

A.7.13.14.2 Example

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

```
GET /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>
…
```

A.7.13.15 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 shall be provided. If <eventType>/EventType refers to a motion-related event, the <DetectionRegionList> block or RegionIndexX parameter(s) shall be provided if detection regions have been defined. If the motion event is for a full-screen configuration, these region indeces 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 indeces 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 shall 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.		

A.7.13.15.1 Notification Alert

```
version=1.0
transactionID=<transaction ID>
action=update
DeviceID=
DeviceName=
IPAddress=
IPv6Address=
PortNo=
Protocol=
MacAddress=
ChannelID=
DateTime=
ActivePostCount=
EventType=
EventState=
EventDescription=
InputIOPortID=
RegionIndex1=
SensitivityLevel1=
DetectionThreshold1=
RegionIndex2=
SensitivityLevel2=
DetectionThreshold2=
…
```

A.7.13.16 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.

A.7.13.16.1 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)
```

A.7.13.17 Event triggering examples

A.7.13.17.1 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 five 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, five seconds from the last notification) with an <eventState> of "active".
- After the signal detection stops for input port 1, the device will wait one second before starting to detect the signal again for this port (indicated by <intervalBetweenEvents>).

```
POST /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>
```

A.7.13.17.2 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 /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>
```

A.7.13.17.3 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 /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
```

Bibliography

ISO/IEC 9945-1:2003, *Information technology – Portable Operating System Interface (POSIX®) –Part 1: Base definitions*

ISO 8601*, Data elements and interchange formats – Information interchange – Representation of dates and times*

IETF RFC 2068, *Hypertext Transfer Protocol HTTP/1.1*, January 1997, Section 19.7.1 Compatibility with HTTP/1.0 Persistent Connections

IETF RFC 2069, *An Extension to HTTP: Digest Access Authentication*

IETF RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

IETF RFC 2571, *An Architecture for Describing SNMP Management Frameworks*

IETF RFC 2782, *A DNS RR for specifying the location of services (DNS SRV)*

IETF RFC 3164, *The BSD syslog Protocol*

IETF RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management*

IETF RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

IETF RFC 4122, A *Universally Unique IDentifier (UUID) URN Namespace*

ITU-T Recommendation H.323, *Packet-based multimedia communications systems*

UIT-T Recommendation I.362.2*, AAL type 2 service specific convergence sublayer for narrow-band services*

____

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™