

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Video surveillance systems for use in security applications –
Part 2-2: Video transmission protocols – IP interoperability implementation
based on HTTP and REST services**

**Systèmes de vidéosurveillance destinés à être utilisés dans les applications
de sécurité –
Partie 2-2: Protocoles de transmission vidéo – Mise en œuvre de
l'interopérabilité IP en fonction des services HTTP et REST**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2013 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you to find IEC publications by a variety of criteria (reference number, text, technical committee,...).

It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,...).

Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**Video surveillance systems for use in security applications –
Part 2-2: Video transmission protocols – IP interoperability implementation
based on HTTP and REST services**

**Systèmes de vidéosurveillance destinés à être utilisés dans les applications
de sécurité –
Partie 2-2: Protocoles de transmission vidéo – Mise en œuvre de
l'interopérabilité IP en fonction des services HTTP et REST**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XF

ICS 13.320

ISBN 978-2-8322-1188-5

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	4
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

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**VIDEO SURVEILLANCE SYSTEMS FOR USE
IN SECURITY APPLICATIONS –****Part 2-2: Video transmission protocols –
IP interoperability implementation based
on HTTP and REST services**

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62676-2-2 has been prepared by IEC technical committee 79: Alarm and electronic security systems.

The text of this standard is based on the following documents:

FDIS	Report on voting
79/436/FDIS	79/449/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62676 series, published under the general title *Video surveillance systems for use in security applications*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

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 la 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 <i>Location</i> 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 <i>Location</i> 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 <i>Location</i> 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 <i>WWW-Authenticate</i> 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 <i>Allow</i> 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 <i>Retry-After</i> 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.

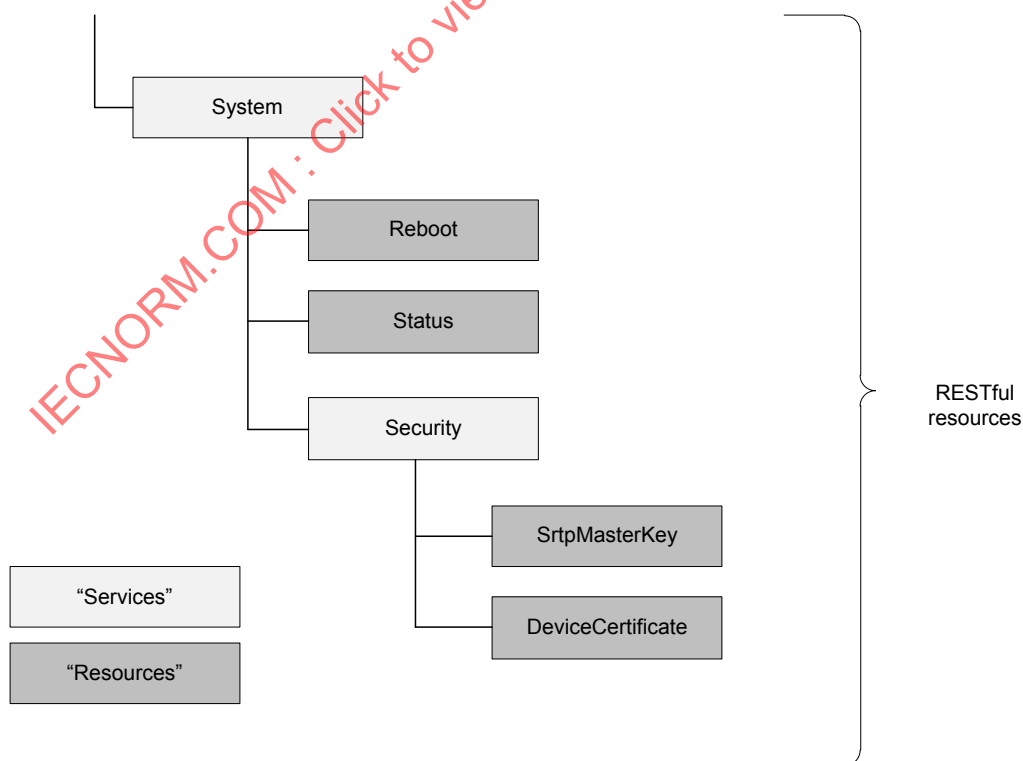


Figure 1 – PSIA service architecture example

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 (Multicast 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 Transformation Format) encoding according to RFC 3629. A BOM (byte-order 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. <i>Example:</i> 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 <i>not</i> 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 <i>not</i> 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 <i>not</i> 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)	<i>Example:</i> If a device supports 5 users the example would be <pre><UserSetting> <UserList size="5"> ...</pre>	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>
  <!-- 0=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>
    <!-- 0=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>
```

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

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	Service URL	Notes
✓	/	
✓	/System	
	/System/Storage	Not all IP media devices support storage.
✓	/System/Network	
	/System/IO	
	/System/Audio	
	/System/Video	
	/System/Serial	
	/Diagnostics	
✓	/Security	
	/Security/AAA	
	/Streaming	
	/PTZ	
	/Custom/MotionDetection	
	/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/mailling	✓	✓	✓	✓
	notification/mailling/<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&rotationDegree=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-nhr	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>
        <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 <code>Accept</code> : 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 <code>Accept:</code> 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 <i>added</i> 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/ <i>ID</i>		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/ <i>ID</i> /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/ <i>ID</i> /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/ <i>ID</i> /files/ <i>ID</i>			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/ <i>ID</i> /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/ <i>ID</i>		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 <authenticationProtocolType> 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 IPFilterAddress 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/>
  <SNMPNotificationFilterList/>
  <notificationEnabled>
    <!-- opt, xs:boolean -->
  </notificationEnabled>
  <SNMPNotificationReceiverList/>
</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>
  </SNMPUser>
</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

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

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

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

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:psi-alliance-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/ <i>ID</i> /qos/cos/ <i>ID</i>			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>
    <enabled>
      <!-- req, xs:boolean -->
    </enabled>
  </UPnP>
  <Zeroconf>
    <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>
```

```
</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:psalliance-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:psalliance-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:psalliance-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:integer, -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/ <i>ID</i> /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/ <i>ID</i> /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/ <i>ID</i> /privacyMask/regions/ <i>ID</i>		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/ <i>ID</i>			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,bidirectional"
--> </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 it is 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/ <i>ID</i>		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/AAA/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 XMI Block

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

A.7.10.3 /Streaming/Channels/<ID>

URI	/Streaming/channels/ <i>ID</i>		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/<i>ID</i>. <audioInputChannelID> refers to /System/Audio/channels/<i>ID</i>. 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/<i>ID</i>/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>
        <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>
        <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>
  </Video>
</StreamingChannel>

```

```

        <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"/>
            <rtspTransportType 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"/>
            <tTtl 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>
    </Video>
</StreamingChannel>

```

```

        <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">32</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/ <i>ID</i> /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		Stream over HTTP	
POST	fixedQuality vbrUpperCap vbrLowerCap maxFrameRate keyFrameInterval rotationDegree mirrorEnabled snapshotImageType	<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 <code>/Streaming/channels/<i>ID</i>/http/capabilities</code>.			

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/ <i>ID</i> /picture			Type	Resource
Function	Get a snapshot of the current image.				
Methods	Query String(s)	Inbound Data	Return Result		
GET	videoResolutionWidth videoResolutionHeight videoPositionX videoPositionY		Picture over HTTP		
POST	rotationDegree mirrorEnabled snapShotImageType	<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 <code>Accept:</code> 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/<i>ID</i>/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/ <i>ID</i> /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:psalliance-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/ <i>ID</i> /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/ <i>ID</i> /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/ <i>ID</i> /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/ <i>ID</i> /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/ <i>ID</i> /presets/ <i>ID</i>		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 <code>strptime()</code> 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/ <i>ID</i> /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>
    <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>

```

</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>

```

```

        </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/>
</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 Setup 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.

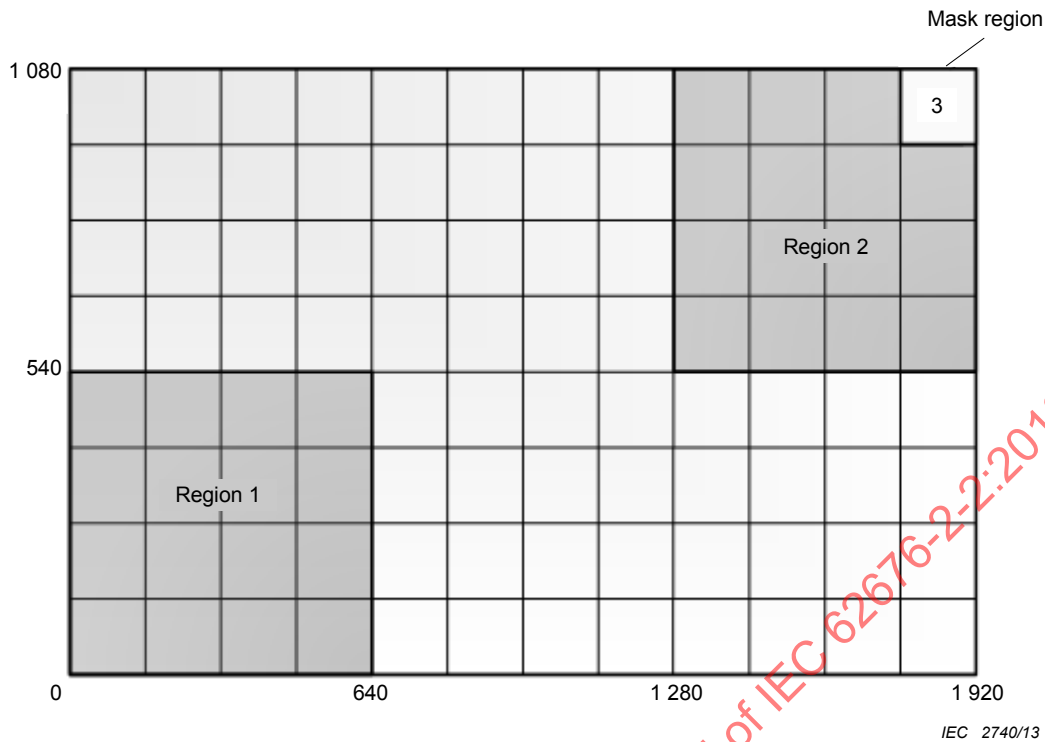


Figure A.1 – Motion detection grid with two detection regions

```
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>
  </MotionDetectionRegionList>
</MotionDetection>
```

```

        </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 <notificationMethod> 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>
    <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>
    <senderEmailAddress> <!-- req, xs:string -->
  </senderEmailAddress>
    <receiverEmailAddress> <!-- req, xs:string -->
  </receiverEmailAddress>
    <subject></subject>
    <BodySetting>
      <attachedVideoURLEnabled> <!-- req, xs:boolean -->
    </attachedVideoURLEnabled>
  </EmailFormat>
</EventNotificationMethods>
```

```

        <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/mailling			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/ <i>ID</i>		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/ <i>ID</i>			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 "quervstring". 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:psalliance-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 indices 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 indices 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:psalliance-org">
  <TimeBlockList>
    <TimeBlock>
      <dayOfWeek>1</dayOfWeek>
      <bitString>00000000111111111000010</bitString>
    </TimeBlock>
    <TimeBlock>
      <dayOfWeek>2</dayOfWeek>
      <TimeRange>
        <beginTime>07:00:00</beginTime>
        <endTime>17:00:00</endTime>
      </TimeRange>
    </TimeBlock>
    <TimeBlock>
      <dayOfWeek>3</dayOfWeek>
      <bitString>00000000111111111000010</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*

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

SOMMAIRE

AVANT-PROPOS	126
INTRODUCTION.....	128
1 Domaine d'application	129
2 Références normatives	129
3 Abréviations	130
4 Présentation	132
5 Considérations sur la conception	133
5.1 Généralités.....	133
5.2 Présentation de REST	133
5.3 Conformité	134
5.3.1 Généralités.....	134
5.3.2 Jeu minimum d'API	134
5.3.3 Exigences XML.....	134
5.3.4 Exigences de protocole	134
5.4 Méthodes HTTP et REST	134
5.5 Codes de statut HTTP et REST	135
5.6 Identifiants uniques	137
5.7 Codage des ID	137
6 Architecture et espace de nom	137
7 Flux système	140
7.1 Généralités.....	140
7.2 Découverte de service	141
7.3 Connexions persistantes	141
7.4 Authentification	142
7.5 Restrictions d'accès	143
7.6 Réglage des configurations	143
7.7 Obtention des configurations	144
7.8 Obtention des fonctionnalités (capabilities)	144
7.9 Chargement de données	145
7.10 Réception de données.....	145
7.11 Opérations	146
7.12 Diagnostics	146
7.13 Statut de réponse.....	146
7.13.1 Généralités.....	146
7.13.2 Code de statut.....	146
7.13.3 Chaîne de statuts	147
7.13.4 ID	147
7.14 Règles de traitement	147
8 Modélisation XML	147
8.1 Format de fichier	147
8.2 Structures de données	147
8.3 Listes	148
8.4 Fonctionnalités	148
9 Services et ressources personnalisés	150
10 Conception de l'interface	150
10.1 Généralités.....	150

10.2	Protocole.....	150
10.3	Nom d'hôte.....	150
10.4	Port.....	150
10.5	URI	150
10.6	Chaîne de requête.....	151
10.7	Description de ressource.....	151
11	Descriptions de ressources normalisées.....	152
11.1	Généralités.....	152
11.2	index	152
11.3	indexr.....	152
11.4	description	152
11.5	capabilities (fonctionnalités)	152
11.6	Schémas	153
11.6.1	Généralités.....	153
11.6.2	ResourceDescription	154
11.6.3	ResourceList	154
11.6.4	QueryStringParameterList	154
11.6.5	responseStatus	154
11.6.6	service.xsd	155
	Annexe A (normative) Spécification API de dispositif média IP Version 1.0.....	158
	Bibliographie.....	247
	Figure 1 – Exemple d'architecture de service PSIA	138
	Figure A.1 – ID de détection de mouvement avec deux régions de détection	232
	Tableau 1 – Méthodes HTTP	135
	Tableau 2 – Codes de statut HTTP et REST	135
	Tableau 3 – Noms de ressources.....	139
	Tableau 4 – URL de services.....	139
	Tableau 5 – Requêtes HTTP	146
	Tableau 6 – Attributs de fonctionnalité	148

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

SYSTÈMES DE VIDÉOSURVEILLANCE DESTINÉS À ÊTRE UTILISÉS DANS LES APPLICATIONS DE SÉCURITÉ –

Partie 2-2: Protocoles de transmission vidéo – Mise en œuvre de l'interopérabilité IP en fonction des services HTTP et REST

AVANT-PROPOS

- 1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
- 2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
- 3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
- 4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
- 5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
- 6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
- 7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
- 8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
- 9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale CEI 62676-2-2 a été établie par le comité d'études 79 de la CEI: Systèmes d'alarme et de sécurité électroniques.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
79/436/FDIS	79/449/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 62676, publiées sous le titre général *Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité*, peut être consultée sur le site web de la CEI.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous "<http://webstore.iec.ch>" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

INTRODUCTION

Le comité d'études 79 de la CEI en charge des systèmes d'alarme et de sécurité électroniques ainsi que de nombreuses organisations gouvernementales, de laboratoires d'essai et de fabricants de matériel ont défini un cadre commun pour la transmission vidéosurveillance afin de permettre l'interopérabilité entre les produits.

La série de normes CEI 62676 dédiées aux systèmes de vidéosurveillance est divisée en 4 parties indépendantes:

- Partie 1: Exigences systèmes
- Partie 2: Protocoles de transmission vidéo
- Partie 3: Interfaces vidéo analogiques et numériques
- Partie 4: Directives d'application (à publier)

Chaque partie propose ses propres articles relatifs au domaine d'application, ainsi qu'aux références, définitions et exigences.

La série CEI 62676-2 comprend 3 sous-parties, respectivement numérotées 2-1, 2-2 et 2-3:

CEI 62676-2-1, *Protocoles de transmission vidéo – Exigences générales*

CEI 62676-2-2, *Protocoles de transmission vidéo – Mise en œuvre de l'interopérabilité IP en fonction des services HTTP et REST*

CEI 62676-2-3, *Protocoles de transmission vidéo – Mise en œuvre de l'interopérabilité IP en fonction des services Web*

Cette deuxième sous-partie de la série CEI 62676-2 traite de la mise en œuvre de l'interopérabilité IP en fonction des services HTTP et REST. Elle est basée sur les exigences des protocoles de transmission vidéo IP traités par la CEI 62676-2-1 qui définit les exigences de protocole à satisfaire par une interface de dispositif vidéo IP de haut niveau.

SYSTÈMES DE VIDÉOSURVEILLANCE DESTINÉS À ÊTRE UTILISÉS DANS LES APPLICATIONS DE SÉCURITÉ –

Partie 2-2: Protocoles de transmission vidéo – Mise en œuvre de l'interopérabilité IP en fonction des services HTTP et REST

1 Domaine d'application

La présente Partie de la CEI 62676 spécifie un protocole vidéo IP reposant sur les services HTTP et REST.

Les dispositifs de vidéotransmission sont souvent équipés de serveurs web qui répondent aux requêtes HTTP. La réponse HTTP peut posséder un contenu XML (pour les actions GET), des informations de réponse XML (pour les actions SET) ou divers contenus texte/binaire (pour l'extraction de données de configuration, etc.). REST est une approche de création de services qui expose uniformément toutes les informations sous forme de ressources. La simplicité d'utilisation de REST réside dans son interface d'opérations uniforme. Tout étant représenté sous forme de ressource, les opérations de création, de récupération, de mise à jour et de suppression (CRUD) utilisent le même URI. Cette spécification influence les caractéristiques HTTP et REST pour la transmission vidéo IP.

Un dispositif de vidéotransmission satisfaisant aux exigences de la présente Norme sur la base des services HTTP et REST tel que décrit dans le présent document est déclaré comme étant compatible avec "l'interopérabilité HTTP et REST de la CEI 62676-2".

2 Références normatives

Les documents ci-après, dans leur intégralité ou non, sont des références normatives indispensables à l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

ISO/CEI 10918-1, *Technologies de l'information – Compression numérique et codage des images fixes de nature photographique: Prescriptions et lignes directrices*

ISO/CEI 11172-3:1993, *Technologies de l'information – Codage de l'image animée et du son associé pour les supports de stockage numérique jusqu'à environ 1,5 Mbit/s – Partie 3: Audio*

ISO/CEI 13818-2, *Technologies de l'information – Codage générique des images animées et du son associé: Données vidéo*

ISO/CEI 14496-2:2004, *Technologies de l'information – Codage des objets audiovisuels – Partie 2: Codage visuel*

ISO/CEI 14496-3, *Technologies de l'information – Codage des objets audiovisuels – Partie 3: Codage audio*

ISO/CEI 14496-10:2012, *Technologies de l'information – Codage des objets audiovisuels – Partie 10: Codage visuel avancé*

IETF RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* (disponible en anglais seulement)

IETF RFC 1945, *Hypertext Transfer Protocol – HTTP/1.0* (disponible en anglais seulement)

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

IETF RFC 2250, *Format de charge utile RTP pour la video MPEG1/MPEG2* (disponible en anglais seulement)

IETF RFC 2326, *Real Time Streaming Protocol (RTSP)* (disponible en anglais seulement)

IETF RFC 2435, *Format de charge utile RTP pour la video JPEG* (disponible en anglais seulement)

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1* (disponible en anglais seulement)

IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication* (disponible en anglais seulement)

IETF RFC 2818, *HTTP Over TLS* (disponible en anglais seulement)

IETF RFC 3016, *Format de charge utile RTP pour flux audio/video MPEG-4* (disponible en anglais seulement)

Protocol (SNMPv3) (disponible en anglais seulement)

IETF RFC 3550, *RTP: A Transport Protocol for Real-Time Applications* (disponible en anglais seulement)

IETF RFC 3551, *RTP Profile for Audio and Video Conferences with Minimal Control* (disponible en anglais seulement)

IETF RFC 3629, *UTF-8 un format de transformation de l'ISO 10646* (disponible en anglais seulement)

IETF RFC 3640, *Format de charge utile RTP pour le transport de flux élémentaires MPEG-4* (disponible en anglais seulement)

IETF RFC 3984, *Format de charge utile RTP pour video H.264* (disponible en anglais seulement)

IETF RFC 4566, *SDP: Session Description Protocol* (disponible en anglais seulement)

UIT-T Recommandation G.726, *Modulation par impulsions et codage différentiel adaptatif (MICDA) à 40, 32, 24, 16 kbit/s*

UIT-T Recommandation H.264, *Codage vidéo évolué pour les services audiovisuels génériques*

UIT-T Recommandation T.81, *Technologies de l'information – Compression numérique et codage des images fixes de nature photographique – Prescriptions et lignes directrices*

3 Abréviations

Pour les besoins du présent document, les abréviations suivantes s'appliquent.

AAC	Codage audio avancé (Advanced Audio Coding)
API	Interface de programmation d'application (Application Program Interface)
AVP	Profil audio/vidéo (Audio/Video Profile)
DHCP	Protocole DHCP (Dynamic Host Configuration Protocol)
DNS	Système de noms de domaine (Domain Name System)
HTTP	Protocole HTTP (Hypertext Transfer Protocol)
HTTPS	Protocole HTTPS (HyperText Transfer Protocol over Secure Socket Layer)
IETF	Groupe IETF (Internet Engineering Task Force)
E/S	Entrée/Sortie
IP	Protocole Internet (Internet Protocol)
IPv4	Protocole Internet Version 4
IPv6	Protocole Internet Version 6
ISO	Organisation internationale de normalisation (International Standards Organization)
UIT	Union Internationale des Télécommunications
JFIF	Format d'échange de fichiers JPEG (File Interchange Format)
JPEG	Norme de compression pour image fixe (Joint Photographic Expert Group)
MPEG	Norme de compression pour audio/video (Moving Pictures Experts Group)
NTP	Protocole NTP (Network Time Protocol)
NVS	Dispositif de stockage vidéo réseau (Network Video Storage Device)
POSIX	Interface de système d'exploitation portable (Portable Operating System Interface)
PTZ	Panoramique/horizontal-vertical/zoom (Pan/Tilt/Zoom)
QoS	Qualité de service (Quality of Service)
REST	Architecture pour systèmes hypermédia distribués (REpresentational State Transfer)
RFC	Demande de commentaire (Request for comment) Normes IETF
RTCP	Protocole de contrôle en temps réel (Real Time Control Protocol)
RTP	Protocole de transport sécurisé en temps réel (Secure Real-time Transport Protocol)
RTSP	Protocole de flux en temps réel (Real Time Streaming Protocol)
SDP	Protocole de description de session (Session Description Protocol)
SHA	Algorithme de hachage sécurisé (Secure Hash Algorithm)
SOAP	Simple Object Access Protocol
SRTP	Protocole de transport sécurisé en temps réel (Secure Real-time Transport Protocol)
SSID	Nom de réseau sans fil (Service Set ID)
SSL	Protocole de sécurisation des échanges sur Internet (Secure Sockets Layer SAML Security Assertion Markup Language)
TCP	Protocole de contrôle de transmission (Transmission Control Protocol)
TCP/IP	Protocole de contrôle de transmission/protocole Internet (Transmission Control Protocol / Internet Protocol)
TKIP	Temporal Key Integrity Protocol
TLS	Sécurité de couche transport (Transport Layer Security)
TTL	Durée de vie (Time-To-Live)

UDP	Protocole de datagramme utilisateur (User Datagram Protocol)
UPnP	Protocole réseau Universal Plug and Play
URI	Identificateur uniforme de ressource (Uniform Resource Identifier)
URL	Localisateur uniforme de ressource (Uniform Resource Locator)
UTC	Temps universel coordonné (Universal Time Coordinated)
UTF	Format de transformation Unicode (Unicode Transformation Format)
UTF-8	Codage de caractères informatique unicode (8-bit Unicode Transformation Format URN Uniform Resource Name)
UUID	Identifiant unique universel (Universally Unique Identifier)
VMS	Système de gestion vidéo (Video management system)
VT	Transmission vidéo (Video Transmission)
VTD	Dispositif de vidéo transmission (Video Transmission device)
W3C	Consortium World Wide Web (World Wide Web Consortium)
WPA	Accès protégé Wi-Fi (Wi-Fi Protected Access)
XML	Langage XML (eXtensible Markup Language)
Zeroconf	Partage de ressources de manière transparente sur un réseau local (Zero Configuration Networking)

4 Présentation

Les applications de sécurité et/ou de gestion de réseau nécessitent la capacité de modifier les configurations et de contrôler les comportements des dispositifs vidéo IP: caméras, codeurs, décodeurs, enregistreurs, etc. Cette fonctionnalité peut être obtenue en envoyant une requête HTTP(S) normalisée à l'unité. Le principe de base de cette interopérabilité IP consiste à spécifier et définir des interfaces de programmation d'application (API) HTTP(S) pour les dispositifs VT et leurs fonctionnalités; c'est-à-dire pour régler/récupérer diverses configurations et contrôler le comportement des dispositifs.

Le modèle de service REST Version 1.1 est destiné à aider les groupes de travail de la CEI à créer des protocoles ou à convertir les protocoles existants en un modèle de service normalisé commun à toutes les spécifications approuvées. L'adhésion à ce modèle de service assure l'interopérabilité entre les protocoles conformes.

Par nature, ce modèle est similaire aux services Web, mais il est orienté vers des exigences de calcul léger pour les dispositifs. Ainsi, ces protocoles n'utilisent pas le protocole SOAP comme défini par les services Web selon le W3C, mais utilisent à la place un schéma XML simplifié et/ou des documents de schéma xml (.xsd).

Sauf mention contraire, il convient que toutes les spécifications du présent article traitent tous les aspects de configuration et de gestion comme des ressources utilisant l'architecture REST (REpresentational State Transfer).

Le modèle de service repose sur une architecture REST. Tandis que REST spécifie que toutes les interfaces sont définies comme des ressources, dans le modèle de la présente Norme, ces ressources sont regroupées par service. Cette architecture offre un moyen commode de regrouper les ressources associées dans un espace de nom hiérarchique, et se prête à la découverte de services et à une extension future.

Tout le monde peut ajouter des services à tout moment, dès lors que ces services adhèrent au modèle de service défini. Il convient de faire tous les efforts nécessaires pour entretenir une compatibilité complète en amont lors de l'ajout de nouveaux services. Le modèle de service est conçu pour prendre en charge les extensions avec compatibilité en amont.

5 Considérations sur la conception

5.1 Généralités

Les dispositifs reliés à un réseau sont souvent équipés d'un serveur Web qui entretient diverses pages Web. Ces pages permettent de configurer les dispositifs par l'intermédiaire d'un navigateur. Il est naturel de réutiliser ce serveur Web et le protocole HTTP afin que des applications externes configurent et contrôlent le dispositif. Ainsi, toutes les ressources utilisent une requête HTTP normalisée traitée par le serveur Web du dispositif.

Dans la mesure du possible, il convient que les dispositifs IP mettent en œuvre HTTPS pour prendre en charge la confidentialité des données. On suppose que l'infrastructure du réseau est correctement configurée, avec pare-feu, 802.1x, etc., et d'autres caractéristiques, et qu'elle offre une sécurité de réseau de base. De plus, les dispositifs IP étant généralement des dispositifs de points terminaux, HTTPS est supposé assurer une protection suffisante, en combinaison avec les autres caractéristiques mentionnées ci-dessus.

Certains dispositifs ne sont pas capables de mettre en œuvre HTTPS, et dans certains déploiements, ils peuvent s'avérer inutiles (par exemple, les réseaux fermés). De plus, SSL/TLS implique la gestion de certificat sur un point terminal, ce qui peut poser d'autres problèmes. Les dispositifs intégrés peuvent ne pas avoir de certificat "de confiance" si un client ne sécurise pas le certificat ou ne charge pas un certificat sécurisé. En outre, les certificats peuvent nécessiter d'être régénérés suite à des changements de configuration (adresse IP, etc.).

Ainsi, les protocoles utilisent les méthodes HTTP Get et Post comme décrit dans "Hypertext Transfer Protocol -- HTTP/1.0" (RFC 1945) et "Hypertext Transfer Protocol -- HTTP/1.1" (RFC 2616).

5.2 Présentation de REST

REST est une approche de création de services qui expose uniformément toutes les informations sous forme de ressources. Cette approche est assez différente du mécanisme classique d'appel de procédure à distance (RPC) qui identifie les fonctions qu'une application peut appeler. Dit simplement, une application web REST est basée sur les noms tandis qu'une application web RPC est basée sur les verbes. Par exemple, s'il faut qu'une application Web définisse une API RPC pour la gestion de l'utilisateur, celle-ci peut être écrite comme suit:

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

D'autre part, une API REST pour les mêmes opérations apparaîtrait comme suit:

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

La simplicité de REST réside en partie dans son interface uniforme pour les opérations. Tout étant représenté sous forme de ressource, les opérations de création, de récupération, de mise à jour et de suppression (CRUD) utilisent le même URI.

5.3 Conformité

5.3.1 Généralités

Chaque protocole définit un ou plusieurs services REST conformes. Pour assurer l'interopérabilité, les exigences de conformité suivantes existent également dans chaque protocole.

5.3.2 Jeu minimum d'API

En plus des exigences obligatoires spécifiques au service, un système/dispositif doit prendre en charge tous les services obligatoires.

Chaque spécification peut définir un ou plusieurs services conformes. Un domaine d'application (obligatoire ou facultatif) doit être attribué à chaque service et chaque ressource contenue. Toutes les ressources obligatoires doivent être mises en œuvre dans chaque type de service mis en œuvre.

5.3.3 Exigences XML

Un système/dispositif doit prendre en charge la syntaxe définie par la spécification W3C XML 1.0.

Un système/dispositif doit prendre en charge le jeu de caractères UTF-8 tel que décrit par <http://www.w3.org/International/O-charset>

De plus, le contenu XML doit correspondre aux schémas suivants définis à l'Annexe A:

- "ResourceList XML Schema"
- "ResourceDescription XML Schema"
- "QueryStringParameterList XML Schema"
- "ResponseStatus XML Schema"

Les fournisseurs peuvent éventuellement étendre la présente norme pour y inclure un contenu XML propriétaire, dès lors qu'il n'entre pas en conflit avec le jeu minimum d'API. Dans ce cas, il est recommandé d'utiliser un espace de nom XML spécifique au fournisseur afin d'éviter les conflits de noms susceptibles de se produire avec les révisions ultérieures.

Par exemple, si le fournisseur XYZ123 Inc. prévoit d'étendre la norme XML pour y inclure un paramètre <configOption>, il est recommandé d'utiliser <configOption xmlns="urn:XYZ123-com:configuration:options"> pour éviter de futurs conflits d'espace de nom.

5.3.4 Exigences de protocole

Un système/dispositif doit prendre en charge le transport de XML via le protocole HTTP/1.0 ou HTTP/1.1, comme spécifié respectivement dans les normes RFC 1945 et RFC 2616. Il est particulièrement recommandé d'utiliser HTTP/1.1 afin de pouvoir prendre en charge les caractéristiques clés (connexions persistantes, HTTPS, etc.). Lorsque HTTP 1.0 est mis en œuvre, les applications client ne doivent pas adresser plusieurs messages aux systèmes/dispositifs cibles.

5.4 Méthodes HTTP et REST

Les opérations CRUD sont définies par la méthode HTTP comme indiqué dans le Tableau 1 ci-dessous.

Tableau 1 – Méthodes HTTP

Méthode HTTP	Opération
POST	Créer la ressource
GET	Récupérer la ressource
PUT	Mettre à jour la ressource
DELETE	Supprimer la ressource

Règles empiriques.

Il convient que les appels GET ne modifient jamais l'état du système. Ils sont uniquement censés retourner les données au demandeur et non avoir un quelconque effet secondaire.

Il convient que les appels POST ne soient utilisés que pour AJOUTER un élément qui n'existait pas.

Les appels PUT sont prévus pour mettre à jour une ressource existante, mais si la ressource spécifiée n'existe pas déjà, elle peut également être créée. Il s'agit du comportement par défaut supposé des appels PUT. Si une ressource souhaite dévier de ce comportement, il convient de considérer cela comme une exception et de l'indiquer dans les notes de mise en œuvre de la ressource.

5.5 Codes de statut HTTP et REST

Le Tableau 2 ci-dessous indique comment les codes de statut HTTP sont mappés aux opérations REST, et spécifie le cas général d'utilisation des en-têtes et des corps de réponse. Pour plus d'informations, voir le tableau sous chaque API REST.

Tableau 2 – Codes de statut HTTP et REST

Codes de statut HTTP	Signification REST	POST	GET	PUT	DEL
200	"OK" - La requête a réussi. Notes d'en-tête: Néant Notes de corps: La ressource demandée est retournée dans le corps.		X	X	
201	"Créée" - La requête a créé une nouvelle ressource. Notes d'en-tête: L'en-tête <i>Emplacement</i> contient l'URI de la nouvelle ressource. Notes de corps: La réponse retourne une entité décrivant la nouvelle ressource.	X			
204	"Pas de contenu" - La requête a réussi, mais il n'y a pas de données à retourner. Notes d'en-tête: Néant Notes de corps: Aucun corps n'est autorisé.			X	X
301	"Définitivement déplacée" - La ressource demandée a été déplacée définitivement. Notes d'en-tête: L'en-tête <i>Emplacement</i> contient l'URI du nouvel emplacement. Notes de corps: Le corps peut contenir le nouvel emplacement de la ressource.		X		
302	"Trouvée" - Il convient que la ressource demandée soit accessible par cet emplacement, mais la ressource se trouve en fait à un autre emplacement. Ceci sert typiquement à établir un		X		

Codes de statut HTTP	Signification REST	POST	GET	PUT	DEL
	alias. Notes d'en-tête: L'en-tête <i>Emplacement</i> contient l'URI de la ressource. Notes de corps: Le corps peut contenir le nouvel emplacement de la ressource.				
400	"Mauvaise requête" - La requête est mal formulée. Ceci sert communément à créer ou mettre à jour une ressource mais les données sont incomplètes ou incorrectes. Notes d'en-tête: La phrase-raison envoyée avec l'en-tête de statut HTTP peut contenir des informations sur l'erreur. Notes de corps: La réponse peut contenir plus d'informations sur l'erreur sous-jacente qui s'est produite en plus de la phrase-raison.	X		X	
401	"Non autorisé" - La requête nécessite l'authentification de l'utilisateur pour accéder à cette ressource. Si la requête contient des données d'authentification non valides, ce code est envoyé. Notes d'en-tête: Au moins un mécanisme d'authentification doit être spécifié dans l'en-tête <i>Authentication WWW</i> . La phrase-raison envoyée avec l'en-tête de statut HTTP peut contenir des informations sur l'erreur. Notes de corps: La réponse peut contenir plus d'informations sur l'erreur sous-jacente qui s'est produite en plus de la phrase-raison.	X	X	X	X
403	"Interdit" - La requête n'est pas autorisée car le serveur refuse de remplir la requête. Une raison courante pour cela est que le dispositif ne prend pas en charge la fonctionnalité demandée. Notes d'en-tête: La phrase-raison envoyée avec l'en-tête de statut HTTP peut contenir des informations sur l'erreur. Notes de corps: La réponse peut contenir plus d'informations sur l'erreur sous-jacente qui s'est produite en plus de la phrase-raison.	X	X	X	X
404	"Non trouvée" - La ressource demandée n'existe pas. Notes d'en-tête: Néant Notes de corps: Néant	X	X	X	X
405	"Méthode non autorisée" - La requête a utilisé une méthode HTTP qui n'est pas prise en charge pour la ressource car la spécification {Protocole API} n'autorise pas cette méthode. Si le dispositif ne prend pas en charge la fonctionnalité, mais qu'il s'agit d'une opération {Protocole API} valide, un code 403 est retourné. Notes d'en-tête: L'en-tête <i>Autoriser</i> énumère les méthodes HTTP prises en charge pour cette ressource. Notes de corps: Néant	X	X	X	X
500	"Erreur serveur interne" - Une erreur de serveur interne s'est produite. Notes d'en-tête: Néant Notes de corps: Néant	X	X	X	X
503	"Service indisponible" - Le serveur HTTP fonctionne, mais le service REST n'est pas	X	X	X	X

Codes de statut HTTP	Signification REST	POST	GET	PUT	DEL
	disponible. Cela s'explique en général par de trop nombreuses demandes client. Notes d'en-tête: L'en-tête <i>Réessayer-après</i> suggère au client le moment où il peut tenter de soumettre à nouveau la requête. Notes de corps: Néant				

5.6 Identifiants uniques

Les ID sont définis sous forme de chaînes URL valides, comme requis par REST. Le dispositif crée un ID pour toutes les ressources qui ajoutent une ressource.

Il convient que les ID de chaque type soient uniques au moins au niveau de la voie, mais il n'existe pas d'exigence d'identifiant unique sur les dispositifs. Si l'on souhaite des ID globalement uniques, il convient de déduire un ID unique au moyen de la méthode décrite dans la norme RFC 4122.

5.7 Codage des ID

Étant donné que les ID apparaissent dans le cadre d'un URI, il existe deux moyens de les coder: en se conformant au RFC 3986 ou, pour les ID purement binaires, sous la forme d'une chaîne hexadécimale.

La norme RFC 3986 convertit d'abord l'URI en UTF, puis imprime les caractères non réservés suivants dans l'URI sans aucun codage:

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

Tous les caractères non imprimables ou réservés sont codés sous la forme d'une valeur hexadécimale à deux chiffres précédée du préfixe %. Par exemple, un espace (valeur ASCII: 32) est codé ainsi: %20.

Étant donné qu'un ID purement binaire peut contenir des valeurs susceptibles de gêner le fonctionnement des navigateurs et des serveurs Web, les protocoles prennent en charge le codage hexadécimal de l'ID. L'ID doit commencer par 0x (0X est acceptable également) suivi de paires de valeurs hexadécimales. Chaque paire hexadécimale représente un seul octet dans l'ID. Par exemple: 0x3F431245DE67FAC46F9D034CA23AEFD4. Les caractères hexadécimaux A-F peuvent également être représentés par a-f. Ainsi, 0x3f431245de67fac46f9d034ca23aefd4 est équivalent à l'ID précédent.

Pour que les ID soient lisibles, il est recommandé de les créer à l'aide de caractères ASCII non réservés et imprimables.

6 Architecture et espace de nom

Dans un espace de nom REST classique, chaque nœud ou objet d'une arborescence est considéré comme une ressource.

Le modèle de service ajoute une sous-classe de ressource appelée "Service". Les services sont simplement des nœuds qui peuvent contenir d'autres nœuds. Les nœuds qui ne contiennent pas d'autres nœuds (autres que les ressources de nœud normalisées du modèle) sont toujours appelés des ressources, alors que le terme nœud est utilisé pour se référer à la fois aux services et aux ressources.

Sous la forme d'un arbre, les services sont analogues à des branches et les ressources sont analogues à des feuilles.

La Figure 1 ci-après donne un exemple d'architecture de service PSIA.

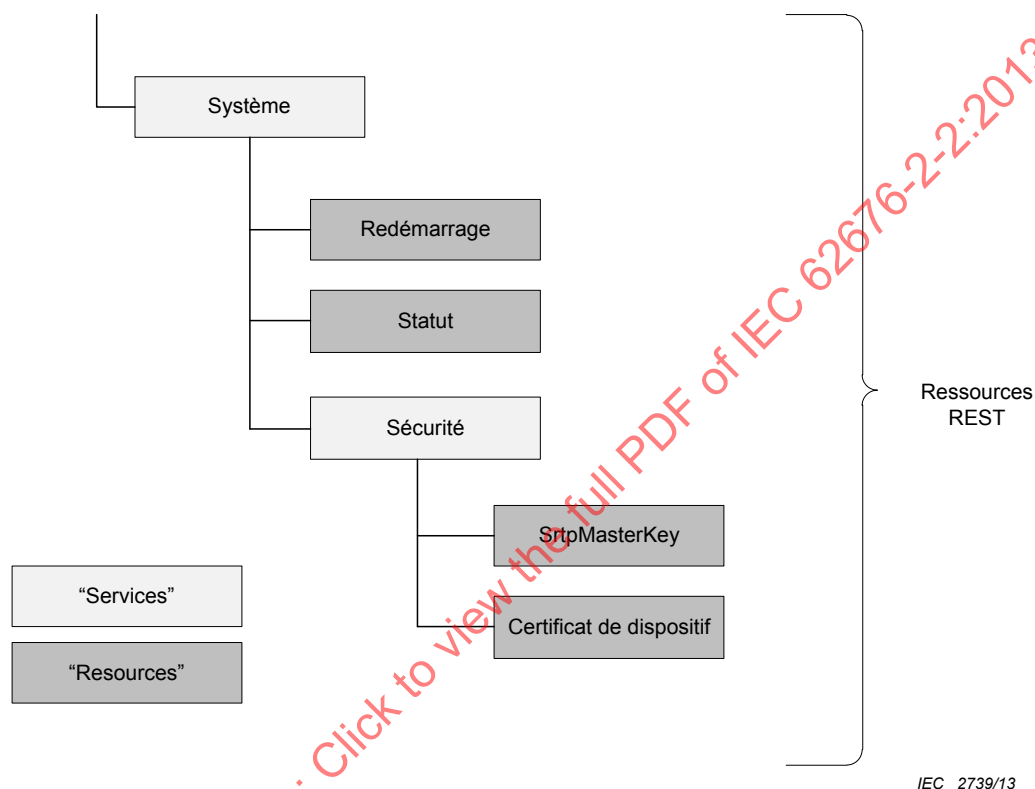


Figure 1 – Exemple d'architecture de service PSIA

Chaque nœud doit contenir les ressources normalisées suivantes (voir Tableau 3):

description qui répond à une requête GET HTTP avec un bloc de données ResourceDescription

indice qui répond à une requête GET HTTP avec un bloc de données ResourceList

Chaque nœud peut contenir les ressources normalisées suivantes:

indexr qui répond à une requête GET HTTP avec un bloc de données ResourceList

capabilities qui répond à une requête GET HTTP avec un document XML spécifique à la ressource

La ressource index renvoie une liste de tous les "enfants" immédiats d'un nœud. Pour les services, cette liste peut contenir d'autres services, ainsi que des ressources (voir Tableau 4). Pour les ressources, il convient que cette liste indique uniquement les ressources normalisées contenues (description IE, index et, éventuellement, indexr et capabilities). La ressource facultative indexr renvoie une liste récursive qui descend dans la hiérarchie d'espace de nom.

Tableau 3 – Noms de ressources

Nom de la ressource	Description	Obligatoire/Facultatif
description	répond à une requête GET HTTP avec un bloc de données <ResourceDescription>	Obligatoire
capabilities	répond à une requête GET HTTP avec un document XML spécifique à la ressource	Facultatif
index	répond à une requête GET HTTP avec un bloc de données <ResourceList>	Obligatoire
indexr	répond à une requête GET HTTP avec un bloc de données <ResourceList>	Facultatif

Pour tous les protocoles de cet article, l'espace de nom racine de "PSIA" est demandé, ce qui signifie qu'il faut l'inclure dans l'URL. Par conséquent, la racine de l'espace de nom d'un service est "PSIA". Chaque service est obligatoire ou facultatif, ce qui indique aux personnes chargées de la mise en œuvre quels services elles doivent mettre en œuvre au minimum. Dans chaque service, les ressources sont également obligatoires ou facultatives. Ce domaine d'application est hiérarchique, de sorte que toutes les ressources d'un service facultatif sont, par définition, facultatives, mais que si un type de service facultatif est déployé, alors chaque ressource obligatoire dans ce service devient obligatoire. Le Tableau 4 liste les URL de services possibles.

Tableau 4 – URL de services

URL de service	Description	Obligatoire/Facultatif
/System	Ressources associées à la configuration et au fonctionnement général du système	Obligatoire
/System/Storage	Ressources relatives au stockage local	Facultatif
/System/Network	Ressources relatives aux paramètres du réseau	Obligatoire
/Security	Ressources relatives à la sécurité du dispositif	Obligatoire
/Security/AAA	Ressources relatives aux fonctions AAA	Obligatoire
/Streaming	Ressources relatives au contenu de transmission en continu	Facultatif
/PTZ	Ressources relatives au panoramique / horizontal / vertical / zoom	Facultatif
/Archive	Ressources relatives au stockage du contenu	Facultatif
/Diagnostics	Ressources relatives aux diagnostics	Facultatif
/Personnalisé	Ressources spécifiques à un protocole ou un fournisseur	Facultatif

Voies et versions multiples

Pour permettre la prise en charge de voies multiples, un service doit insérer le service "Voies" concerné sous la forme d'un nœud enfant. Il convient que celui-là contienne alors une ressource d'ID pour chaque voie. Chaque ressource d'ID répond alors à chacune des ressources applicables au service.

Pour les dispositifs à voie unique, le service Voies doit également être inclus pour maintenir la cohérence entre les dispositifs à voies unique et multiples, et pour prévoir le cas où un dispositif à voies multiples n'a créé qu'une seule voie.

Noter que les ID de voies sont attribués de manière arbitraire par le dispositif.

(par exemple, pour un dispositif à voie unique:

/Streaming/Channels/0/keyFrame

pour un dispositif à voies multiples

/Streaming/Channels/0/keyFrame

/Streaming/Channels/1/keyFrame)

Les dispositifs peuvent prédéfinir cette structure à voies multiples ou prendre en charge les ajouts et suppressions dynamiques de voies (au moyen de POST et DELETE HTTP), selon le cas.

Afin de différencier les services qui sont essentiellement prévus pour plusieurs instances d'un élément de la hiérarchie, il est recommandé d'appeler les services à la racine "services racine", le terme service étant toujours utilisé pour décrire un nœud contenant d'autres nœuds (la transmission en continu est un service racine, à l'inverse des voies, par exemple).

Chaque nœud, qu'il s'agisse d'une ressource ou d'un service, est capable de renvoyer une description de lui-même dans le modèle de service. Cette description inclut un attribut de version afin de prendre en charge les différentes versions dans le modèle de service. Même si cette pratique permet à des ressources de différentes versions d'exister au sein des mêmes services, il est obligatoire que toutes les ressources d'un conteneur de services soient totalement compatibles en amont.

Si une nouvelle version d'un service est introduite, mais qu'elle n'est pas compatible en amont avec les versions précédentes, un nouveau service doit être créé pour la nouvelle version incompatible (/Streaming et /StreamingV2, par exemple). Il est acceptable d'ajouter des ressources à un service, mais pas de les remplacer par de nouvelles versions qui ne sont pas compatibles en amont. Si de nouvelles versions de ressource doivent être ajoutées, il convient de modifier le nom du service racine afin d'indiquer une nouvelle version de service.

7 Flux système

7.1 Généralités

Pour pouvoir utiliser un protocole de gestion d'un dispositif, il doit en premier lieu être découvert. Il est nécessaire que la technologie ZeroConf (Zero Configuration Networking) soit prise en charge pour découvrir/localiser le dispositif. Une fois cette étape franchie, les transactions peuvent commencer.

Les dispositifs doivent prendre en charge ZeroConf, mais cela ne les empêche pas d'utiliser DHCP ou l'adressage IP manuel. Il convient que les dispositifs contrôlent les adresses IP attribuées manuellement et les adresses IP DHCP avant de tenter d'attribuer une adresse à l'aide de l'adressage IPv4 Link-Local (qui est la méthode de découverte des adresses IP pour ZeroConf).

ZeroConf est normalement prévu pour fonctionner dans un réseau local. Si la découverte doit être prise en charge dans un réseau d'acheminement ou étendu, la partie 2 de ZeroConf

(DNS de multidiffusion) devient superflue et la partie 3 (DNS-SD) doit être prise en charge par la configuration des serveurs DNS.

Les requêtes HTTP sont effectuées par l'intermédiaire du serveur Web du dispositif. La réponse HTTP peut posséder un contenu XML (pour les actions GET), des informations de réponse XML (pour les actions PUT ou POST) ou divers contenus texte/binaire (pour l'extraction de données de configuration, etc.). Il convient que les dispositifs périphériques soient capables de traiter des requêtes HTTP en chevauchement/simultanées, ainsi que des connexions persistantes, afin de traiter plusieurs transactions HTTP.

Il convient que le contenu XML soit décrit par des documents .xsd. Les structures de données XML pertinentes doivent être documentées dans une annexe de chaque spécification.

7.2 Découverte de service

La technologie Zeroconf (Zero Configuration Networking) spécifie la DNS-SD (Découverte de service DNS comme décrit à <http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt>) pour découvrir/localiser un dispositif.

Tous les protocoles de ce paragraphe nécessitent DNS-SD pour la découverte de dispositif. Pour prendre en charge ce modèle de découverte, la PSIA enregistre un type de service DNS SRV (RFC 2782) à utiliser pour découvrir tous les protocoles vidéo IP via DNS-SD (DNS Service Discovery).

Il convient d'utiliser les découvertes DNS-SD pour le type de service DNS public de la PSIA pour découvrir le dispositif selon la découverte de service DNS (<http://www.dns-sd.org/ServiceTypes.html>). Une fois qu'un dispositif est établi comme étant conforme, ses services et ressources peuvent être découverts à l'aide des requêtes normalisées GET HTTP au moyen des ressources obligatoires normalisées.

Il convient d'indiquer les informations suivantes:

Un chemin de "/index/" - peut être obtenu à partir de la touche "chemin" dans l'enregistrement TXT

L'{hôte} - peut être obtenu à partir de l'enregistrement SRV du service

Le {port} - peut être obtenu à partir de l'enregistrement SRV du service

La version de l'enregistrement SVR DNS dans "txtvers"

La version du protocole vidéo IP dans "protovers"

Si un dispositif conforme a été découvert, une requête GET HTTP de sa ressource d'index obligatoire renvoie une liste des services qu'il prend en charge. À ce stade, les méthodes normalisées peuvent être utilisées pour explorer l'arbre d'espace de nom et découvrir les services et les ressources pris en charge.

Il convient de noter que la ressource d'index renvoie uniquement les ressources de premier niveau d'un nœud, mais que la ressource indexr renvoie une liste à structure arborescente récursive dont la ressource courante est la racine.

7.3 Connexions persistantes

Il convient que les dispositifs qui mettent en œuvre HTTP/1.1 prennent en charge les connexions persistantes afin de gérer les systèmes de gestion vidéo ou les applications client qui émettent plusieurs transactions HTTP(S). La présente Norme suppose que HTTP/1.1 est mis en œuvre et utilisé conformément à la norme RFC 2616. Pour les connexions

persistantes avec les dispositifs qui prennent en charge HTTP/1.0, il convient de consulter la norme RFC 2068, section 19.7.1.

Lorsqu'un système de gestion vidéo (ou une application client) utilise une connexion persistante pour des transactions multiples, il convient qu'il mette en œuvre l'en-tête HTTP "Connexion: Keep-Alive". Il convient également que le système de gestion utilise le champ d'en-tête HTTP "Connexion:Fermer" pour la dernière transaction effectuée sur cette connexion persistante. Ce processus suppose que l'application connaît la dernière requête dans une séquence de plusieurs requêtes.

7.4 Authentification

Lorsqu'une application envoie une requête au dispositif, elle doit être authentifiée au moyen d'une authentification Basic Access ou Digest conformément à la norme RFC 2617. Cela signifie que les justificatifs d'accès de l'utilisateur sont envoyés avec chaque requête. Lorsqu'un utilisateur est authentifié, la requête suit le flux d'exécution normal. L'authentification Basic Access et/ou Digest est obligatoire pour toutes les mises en œuvre de dispositif. C'est au client de déterminer quelle méthode utiliser pour les différents scénarios de déploiement.

Le dispositif IP doit fournir un compte d'utilisateur par défaut "admin". Il convient que ce compte ait un niveau de privilège par défaut d'"administrateur" et il ne doit pas être supprimé. Il convient que le mot de passe par défaut du compte "admin" soit nul dans la configuration d'usine par défaut.

Exemple d'en-tête et de corps de requête HTTP client sans justificatif d'authentification:

```
GET /index
...
```

Exemple d'en-tête et de corps de réponse HTTP non autorisée:

```
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>
```

Exemple d'en-tête et de corps de requête HTTP client avec justificatif d'authentification (nom d'utilisateur "Mufasa" et mot de passe "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"
```

Exemple d'en-tête et de corps de réponse HTTP autorisée:

```
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 Restrictions d'accès

Toutes les ressources prises en charge sur un dispositif doivent être totalement accessibles aux utilisateurs dotés du niveau de privilège "Administrateur". En d'autres termes, pour pouvoir utiliser la suite complète de ressources offertes par un dispositif, l'authentification doit être accordée avec un compte d'utilisateur dont le niveau de privilège correspond à "Administrateur".

Il est nécessaire qu'au moins un compte doté des privilèges d'administrateur soit actif à tout moment.

Il n'existe aucune restriction quant aux ressources accessibles aux utilisateurs dotées d'autres niveaux de privilège. Par exemple, un fournisseur peut choisir de limiter les ressources autorisées pour les comptes d'utilisateur dotés de privilèges plus faibles. Cependant, l'autorisation spécifique à l'utilisateur n'étant pas une fonction du protocole, on suppose souvent que les pleins droits d'administration sont disponibles via le protocole. Les fonctions d'autorisation spécifique à l'utilisateur sont censées être traitées par l'application appelante.

Bien que le niveau de privilège "Administrateur" doive être prévu, il n'existe aucune exigence d'attribution à des utilisateurs spécifiques d'un niveau de privilège administratif. Dans les cas où des exigences externes empêchent un utilisateur unique de jouir de tous les privilèges, une autorisation plus granulaire peut être obtenue en utilisant des attributions d'utilisateur et de rôle qui n'ont pas les privilèges administratifs complets.

7.6 Réglage des configurations

Les ressources de réglage des configurations des dispositifs utilisent la méthode PUT HTTP s'il existe un paramètre de bloc XML pour la requête, et la méthode GET HTTP en l'absence de paramètre de bloc XML. Le format XML entrant est défini en fonction d'un schéma XML spécifique à la ressource. Pour les opérations PUT, le statut de la requête est indiqué par les informations de réponse XML renvoyées depuis le dispositif, et il peut être utilisé pour indiquer le statut de l'opération de réglage. Ce format XML est défini conformément au "Schéma de réponse XML (XML Response Schema)" (voir les détails en 7.13). Après avoir correctement mis à jour le référentiel, le dispositif renvoie une réponse XML avec le code de statut "OK". Un code de statut séparé est utilisé pour les opérations non réussies. Dans les deux cas, le dispositif ne renvoie pas de réponse tant qu'il n'est pas prêt à continuer en fonctionnement normal. Cela inclut d'accepter des requêtes de transmission en continu, de recevoir des commandes de comportement, etc.

Exemple d'en-tête et de corps de requête HTTP:

```

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>

```

Exemple d'en-tête et de corps de réponse HTTP:

```
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 Obtention des configurations

Les ressources permettant d'obtenir les configurations des dispositifs ou les informations de statut utilisent la méthode GET HTTP. En cas de réussite, le résultat est renvoyé au format XML selon la description de la ressource. Si pour une quelconque raison la requête échoue (par exemple, pas d'authentification), le résultat est renvoyé au format XML selon le "ResponseStatus XML Schema". Le Content-Type et le Content-Length sont établis dans les en-têtes de la réponse HTTP contenant les données XML. Le Content-Type est: application/xml; charset="UTF-8"

Exemple d'en-tête et de corps de requête HTTP:

```
GET /System/deviceInfo HTTP/1.1
...
```

Exemple d'en-tête et de corps de réponse HTTP:

```
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 Obtention des fonctionnalités (capabilities)

Les fonctionnalités peuvent également être récupérées par un nœud de ressource qui spécifie une charge utile XML pour les données entrantes avec une requête GET HTTP de sa ressource "capabilities" (fonctionnalités). En d'autres termes, une application client peut demander à un dispositif ses fonctionnalités afin de comprendre quelles balises XML sont prises en charge, les plages de données acceptables, etc. Voir 5.5 pour plus de détails sur les fonctionnalités retournées.

Exemple d'en-tête et de corps de requête HTTP:

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

Exemple d'en-tête et de corps de réponse HTTP:

```
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 Chargement de données

Les ressources de chargement de données (c'est-à-dire micrologiciel, fichier de configuration, etc.) sur le dispositif utilisent la méthode PUT HTTP. Le contenu des données est stocké dans le corps de la requête HTTP. Le Content-Type et le Content-Length sont établis dans les en-têtes de la requête HTTP. Le Content-Type est "application/octet-stream". De plus, chaque ressource peut éventuellement spécifier une structure XML d'entrée spécifique.

Exemple d'en-tête et de corps de requête de chargement HTTP:

```

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

[proprietary configuration data content]

```

Exemple d'en-tête et de corps de réponse de chargement HTTP:

```

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 Réception de données

Les ressources permettant d'extraire des données (c'est-à-dire fichier de configuration, etc.) depuis le dispositif utilisent la méthode GET HTTP. Le contenu des données est stocké dans le corps de la réponse HTTP. Le Content-Type et le Content-Length sont établis dans les en-têtes de la réponse HTTP, en fonction du type de données renvoyées.

Le client peut utiliser la chaîne d'en-têtes Accept: pour indiquer au serveur les formats qu'il accepte. En fonction de ce que le client accepte, le serveur peut transcoder, transformer ou même compresser les données pour satisfaire aux attentes du client.

Exemple d'en-tête et de corps de requête de téléchargement HTTP:

```

GET /System/configurationData HTTP/1.1
...

```

Exemple d'en-tête et de corps de réponse de téléchargement HTTP:

```
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 Opérations

Pour les opérations sans état (c'est-à-dire les appels de fonction), la formule est la suivante:

PUT /Service/<Operation>

Les ressources doivent indiquer dans leurs descriptions quelle charge utile XML est requise ou les paramètres de la chaîne de requête à utiliser dans l'opération.

7.12 Diagnostics

Les diagnostics (et autres opérations à état) fonctionnent en arrière-plan du dispositif, on doit donc pouvoir les créer de manière asynchrone et être en mesure de demander leur statut. Le Tableau 5 liste plusieurs requêtes HTTP disponibles pour gérer les opérations de diagnostic.

Le modèle REST fonctionne bien ici:

Tableau 5 – Requêtes HTTP

Requête	Résultat
POST /Diagnostics/<command>	Retourne l'ID de diagnostic
GET /Diagnostics/<command>/<ID>	Obtient l'information sur cet ID
DELETE /Diagnostics/<command>/<ID>	Supprime la commande en cours
GET /Diagnostics/commands	Obtient l'information sur toutes les commandes en cours

7.13 Statut de réponse

7.13.1 Généralités

Les réponses à la plupart des appels de ressources contiennent des données sous la forme du document XML ResponseStatus.

Dans chaque spécification, des services et ressources séparés peuvent avoir chacun leur propre structure de données. La seule disposition du modèle est que chaque ResourceDescription doit indiquer les structures utilisées, et chaque structure doit être définie dans un document de schéma XML dans le document de spécification. Si les ressources ne définissent pas leurs propres structures de réponse, elles peuvent utiliser la structure ResponseStatus de la présente Norme telle que définie à l'Article 10.

7.13.2 Code de statut

Un ResponseStatus avec statusCode=OK est envoyé à l'issue du traitement de la commande sur le dispositif. Même si la requête contient certains paramètres qui ne sont pas pris en charge, le dispositif ignore ces paramètres et renvoie statusCode=OK.

Un dispositif envoie une réponse de dispositif occupé à une commande qui ne peut pas être traitée à cet instant (s'il reçoit une commande de redémarrage alors que la mémoire flash est mise à jour, par exemple).

Si le dispositif échoue à réaliser la requête (suite à une erreur matérielle, par exemple), il renvoie un statusCode Device Error et un message de défaillance dans la statusString.

Un statusCode d'opération non valide est renvoyé en réponse à une commande qui n'a pas été mise en œuvre. Une opération non valide est également renvoyée si une tentative d'authentification échoue ou si le niveau de privilège de l'utilisateur connecté est insuffisant pour exécuter la commande.

Un format XML non valide (Invalid XML Format) est renvoyé si le format XML est incorrect et entraîne la défaillance de l'analyseur. Il convient que la statusString indique la défaillance.

Un message incomplet ou un message contenant un paramètre hors plage renvoie un statusCode de contenu XML non valide et une statusString associée.

Pour les réglages dont la prise d'effet nécessite un redémarrage (le changement de l'adresse du réseau ou une mise à jour de micrologiciel, par exemple), le statusCode Reboot Required (redémarrage requis) est retourné.

7.13.3 Chaîne de statuts

Pour toutes les réponses où le statusCode renvoyé n'est pas OK, il est recommandé qu'une statusString descriptive soit renvoyée, indiquant la raison pour laquelle la commande n'est pas traitée.

7.13.4 ID

Dans les opérations POST où le dispositif retourne un ID de la ressource créée, cet attribut est utilisé pour passer derrière l'ID créé.

7.14 Règles de traitement

Il convient que tous les champs (en particulier dans les paramètres XML entrants) qui ne sont pas pris en charge par le dispositif soient ignorés. Pour une ressource donnée, il peut exister certaines règles de traitement spéciales. Ces règles sont documentées dans la colonne associée au titre "Note de mise en œuvre".

8 Modélisation XML

8.1 Format de fichier

Tous les fichiers XML doivent utiliser le codage UTF-8 (format de transformation Unicode/8 bits UCS) conforme à la norme RFC 3629. En option, un BOM (byte-order mark - marque d'ordre d'octets) peut être utilisé. Ainsi, il convient qu'un dispositif média prenne en charge le codage UTF-8 avec ou sans BOM.

8.2 Structures de données

Une ressource peut spécifier des documents XML d'entrée et de sortie séparés. Si une structure de données spécifique est définie, elle doit être spécifiée sous la forme de documents de schéma XML (xsd) dans la spécification. Les xsd créés pour les spécifications basées sur la présente Norme de services HTTP et REST sont à inclure dans la section Annexes de la spécification concernée. De plus, l'Alliance PSIA va poster les documents xsd des schémas associés sur <http://www.psialliance.org> comme référence en ligne des schémas. Cependant, il n'y a aucune garantie que les schémas soient postés au moment de la

publication des documents. C'est la raison pour laquelle les définitions de schémas dans les documents de spécification eux-mêmes constituent l'exigence minimale.

8.3 Listes

La plupart des blocs XML contiennent des listes. La syntaxe de ces listes est <XXXList>, où XXX est un nom se référant au paramètre XML. À l'intérieur de la balise <XXXList> se trouve(nt) un ou plusieurs nœuds <XXX>. Par exemple, le bloc <ChannelList> peut contenir le contenu suivant:

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

8.4 Fonctionnalités

Les fonctionnalités d'une ressource qui définit un bloc XML d'entrée sont renvoyées sous la forme d'un document XML, qui est essentiellement une instance XML du bloc XML d'entrée spécifique à la ressource. Ce document XML doit contenir les valeurs acceptables pour chaque attribut (voir Tableau 6).

Bien que les documents de schéma XML soient également requis pour toutes les données XML définies par une spécification basée sur la présente norme de services HTTP et REST et que les documents xsd soient capables de définir la plage de valeurs acceptable pour tous les attributs, l'utilisation d'un xsd global pour définir les fonctionnalités implique que tous les dispositifs prennent en charge les mêmes options pour tous les paramètres. En permettant aux dispositifs de répondre à la requête de fonctionnalités, chaque dispositif peut prendre en charge différentes valeurs pour un attribut, dans les contraintes du schéma.

Tableau 6 – Attributs de fonctionnalité

Attribut de fonctionnalité	Description	Syntaxe	Types de données XML applicables
Min	La longueur minimale des caractères pour une chaîne ou la valeur numérique minimale d'un nombre	Exemples: min="0" min="64" min="-100" (numérique uniquement) min="1,2"	Tous les types de données sauf les fixes ^a
max	La longueur maximale des caractères pour une chaîne ou la valeur numérique maximale d'un nombre	Exemples: max="5" max="64" max="4 096" max="10,50"	Tous les types de données sauf les fixes ^a
range	Indique la plage possible de valeurs numériques dans les limites des attributs "min" et "max" d'un élément. Il convient d'utiliser cet attribut uniquement si les	Les plages sont énumérées dans l'ordre alphabétique, séparées par un caractère ",". Une plage se présente sous la forme "x~y", où x est le	Tous les types de données numériques

Attribut de fonctionnalité	Description	Syntaxe	Types de données XML applicables
	valeurs possibles pour un élément XML n'incluent pas la plage numérique entière comprise entre les attributs "min" et "max"	plancher et y est le plafond. Des numéros uniques peuvent également être utilisés. <i>Exemple:</i> si un élément XML prend en charge les valeurs 0, 123, 1024 à 2000, et 2003, la syntaxe est la suivante: range="0,123,1024~2000,2003"	
opt	Énumère les options prises en charge pour un type de données CodeID. Requis pour les éléments XML avec un type de données CodeID. Il convient de ne <i>pas</i> utiliser cet attribut pour un autre type de données	Si toutes les options sont prises en charge, la syntaxe est "all". Sinon, les options prises en charge sont séparées par un caractère ",". Exemples: opt="all" opt="1,2,3" opt="1,2,5,8,9,10,11"	CodeID
Def	Indique la valeur par défaut de l'élément XML. Si l'élément n'a pas de valeur par défaut, il convient de ne <i>pas</i> utiliser cet attribut.	Exemples: def="1234" def="Device ABC" def="3"	Tous les types de données
reqReboot	Indique si la configuration de cet élément XML nécessite un redémarrage du dispositif avant de prendre effet. Si un élément n'a pas besoin d'un redémarrage, il convient de ne <i>pas</i> utiliser cet attribut.	reqReboot="true"	Tous les types de données
dynamic	Indique si des fonctionnalités dynamiques d'un élément XML dépendent d'autres configurations XML. Par exemple, si la plage de données d'un élément change en fonction de la valeur configurée d'un autre élément, cet attribut doit être utilisé. Dans ce cas, les attributs de fonctionnalité de l'élément doivent toujours refléter la configuration courante du dispositif	dynamic="true"	Tous les types de données
Size	Indique le nombre maximal d'entrées dans une liste XML. Cet attribut est applicable uniquement aux éléments de liste XML. Il convient de ne pas utiliser cet attribut pour un autre type d'élément (voir 8.3 pour plus de détails)	<i>Exemple:</i> Si un dispositif prend en charge 5 utilisateurs, l'exemple est le suivant: <UserSetting> <UserList size="5"> ... </UserList> </UserSetting>	Uniquement pour les éléments de liste (voir 8.3)
a Les types de données fixes prédéfinis n'ont pas besoin de certains attributs de fonctionnalité, car leurs formats/plages de données sont déjà définis. Lorsque des types de données prédéfinis sont utilisés, chaque document de protocole doit intégrer une énumération de ces formats dans une annexe.			

9 Services et ressources personnalisés

Afin de prendre en charge les ressources spécifiques à un système/dispositif qui ne sont pas communes à toutes les définitions de service public, le type de service CUSTOM est prévu. Une requête GET HTTP de la ressource d'index du service CUSTOM renvoie une liste des services et ressources personnalisés pris en charge par le système/dispositif.

Pour chaque ressource personnalisée, une ressource implicite obligatoire nommée "Description" doit être prise en charge. Une requête GET HTTP d'une ressource Description pour une ressource personnalisée doit renvoyer un ResourceDescriptionBlock similaire à l'information de description de ressource décrite en 7.7.

Les services et ressources personnalisés peuvent servir à prendre en charge des ressources spécifiques à un protocole supposé être provisoire (c'est-à-dire qu'un protocole à venir va très probablement rendre ces ressources obsolètes) ou des ressources propriétaires spécifiques à un fournisseur. Dès lors que tous les services et ressources personnalisés sont mis en œuvre conformément au présent modèle de service, ils peuvent être découverts et appelés par les clients et les applications satisfaisant au présent article.

10 Conception de l'interface

10.1 Généralités

Le format d'URL HTTP présente la forme générale suivante:

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

Toutes les requêtes suivent ce format. Suit une brève description de ces composants:

10.2 Protocole

Le champ de protocole se réfère au schéma URL utilisé pour la requête particulière. À noter que la spécification actuelle autorise les schémas suivants:

- http
- https

10.3 Nom d'hôte

Le champ de nom d'hôte se réfère au nom d'hôte, à l'adresse IP ou au FQDN (fully qualified domain name - nom de domaine complet) d'un dispositif IP.

10.4 Port

Le champ de port indique le numéro de port à utiliser pour la requête HTTP. Le numéro de port par défaut pour HTTP est 80. Pour HTTPS, le port par défaut est 443. Pour RTSP, le port par défaut est 554. S'ils ne sont pas précisés dans l'URL, ces numéros de port par défaut sont utilisés pour la requête (voir RFC 2616, RFC 2818 et RFC 2326 respectivement).

Les numéros de port HTTP et HTTPS sont configurables pour les dispositifs IP. Les ports HTTP et HTTPS normalisés (80 et 443) sont supposés, sauf spécification contraire.

10.5 URI

Le chemin absolu URI se présente le plus souvent sous la forme "<SERVICE>/<resource>", où <resource> correspond à l'une des ressources définies dans la spécification. Par exemple, <SERVICE> peut se référer à "Système" ou "Sécurité". Cela est vrai pour les ressources qui mettent à jour ou qui récupèrent des configurations de dispositif.

10.6 Chaîne de requête

Les ressources spécifient les paramètres requis et facultatifs de la chaîne de requête. Dans les deux cas, ces paramètres de chaîne de requête doivent apparaître dans une chaîne de paires nom-valeur (p1=v1&p2=v2...&pn=vn) à la suite de l'URI.

Exemple de requête GET HTTP avec paramètres de chaîne de requête:

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

Exemple de requête POST HTTP avec paramètres de chaîne de requête:

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

Chaque ressource peut définir un ensemble de paramètres, sous la forme de paires nom-valeur, qui existent dans la chaîne de requête. Si les ressources définissent des données d'entrée spécifiques à la ressource, cela l'emporte sur l'utilisation des paramètres de chaîne de requête pour l'entrée.

10.7 Description de ressource

Pour chaque ressource de ce document, les composants suivants sont définis:

Format – indique le format URL de la requête HTTP

Type – indique s'il s'agit d'un service ou d'une ressource

Méthode spécifique (GET, PUT, POST, DELETE)

Paramètres de chaîne de requête - indiquent les paires nom-valeur (P1, P2, P3, ..., Pn) pour la ressource.

Données entrantes – indique les données d'entrée pour la ressource, comme suit:

- **NÉANT** – indique qu'il n'y a aucune donnée d'entrée
- **Bloc de données** – nom d'un bloc de données défini dans la spécification. Les blocs de données utilisés ici doivent être définis dans le document de spécification. De plus, il est fortement recommandé que les documents de schéma .xml soient créés pour chaque bloc de données référencé.
- **Type MIME** – indique que les données d'entrée sont dans la charge utile HTTP selon le type MIME indiqué. NOTE un type d'application/xml n'est pas considéré comme valide.

Si un dispositif ne prend pas en charge des balises ou des blocs XML particuliers, ceux-ci n'ont pas besoin d'être utilisés dans les opérations sur les ressources.

Généralement, si des champs ne sont pas prévus dans le XML entrant, il convient que les valeurs courantes pour ces champs restent inchangées dans le référentiel du dispositif.

Si des champs requis n'existent pas déjà dans le référentiel du dispositif, ils doivent être prévus dans les opérations applicables sur les ressources.

Fonction – décrit le comportement général de la fonction

Résultat de retour – décrit la réponse à la requête HTTP

Note de mise en œuvre – décrit le comportement de mise en œuvre et toutes les règles de traitement particulières de la ressource.

Par exemple,

URI	index	Version	1,0	Type	Ressource
Fonction	Énumérer les nœuds enfants				
Méthodes	Chaîne(s) de requête	Données entrantes		Résultat de retour	
GET	Néant	Néant		<ResourceList>	
Notes	Retourne une liste plate (non récursive) de tous les nœuds enfants				

Afin de prendre en charge la découverte de ressources du service CUSTOM, cette structure de données de description de ressource est également capturée sous forme de bloc de données de type ResourceDescription. Chaque fois qu'une requête GET HTTP d'une ressource CUSTOM/Index d'un dispositif est exécutée, une liste des ressources personnalisées du dispositif est retournée. Pour chaque ressource personnalisée, une requête GET HTTP de la ressource obligatoire "Description" renvoie un bloc ResourceDescription indiquant ce que fait la ressource et comment il convient de l'utiliser.

11 Descriptions de ressources normalisées

11.1 Généralités

Le présent article décrit les ressources normalisées.

11.2 index

URI	index	Version	1,0	Type	Ressource
Fonction	Énumérer les nœuds enfants				
Méthodes	Chaîne(s) de requête	Données entrantes		Résultat de retour	
GET	Néant	Néant		<ResourceList>	
Notes	Retourne une liste plate (non récursive) de tous les nœuds enfants				

11.3 indexr

URI	indexr	Version	1,0	Type	Ressource
Fonction	Énumérer les nœuds enfants				
Méthodes	Chaîne(s) de requête	Données entrantes		Résultat de retour	
GET	Néant	Néant		<ResourceList>	
Notes	Retourne une liste récursive de tous les nœuds enfants				

11.4 description

URI	Description	Version	1,0	Type	Ressource
Fonction	Décrire la ressource courante				
Méthodes	Chaîne(s) de requête	Données entrantes		Résultat de retour	
GET	Néant	Néant		<ResourceDescription>	
Notes	Retourne une description de la ressource				

11.5 capacités (fonctionnalités)

URI	capabilities	Version	1,0	Type	Ressource
Fonction	Retourner les fonctionnalités de la ressource courante				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET	Néant	Néant	Spécifique à la ressource		

Notes	Retourne une description des fonctionnalités de la ressource
-------	--

11.6 Schémas

11.6.1 Généralités

Les structures de données qui suivent sont définies pour être utilisées avec le modèle de service du présent paragraphe. Les formats utilisés dans ce paragraphe sont des exemples basiques destinés à représenter rapidement la structure des blocs de données. À noter qu'il faut que les protocoles vidéo IP réels du présent paragraphe incluent leurs structures de données documentées sous forme de fichiers .xsd.

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

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>
  <!-- 0=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

Le document de schéma XML qui suit contient des définitions de schémas XML pour toutes les structures de données du modèle de service. Il faut que toutes les spécifications du présent paragraphe utilisent ce document de schéma pour maintenir la cohérence des structures de données du modèle de service.

Ce document et tous les documents de schéma XML ultérieurs sont postés sur <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>
    <!-- 0=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>
```

IECNORM.COM : Click to view the full PDF of IEC 62676-2-2:2013

Annexe A (normative)

Spécification API de dispositif média IP Version 1.0.

A.1 Présentation

La présente annexe spécifie une interface qui permet aux systèmes de gestion vidéo de communiquer avec divers dispositifs média IP de manière normalisée. Il n'est alors plus utile de personnaliser le pilote du dispositif pour obtenir une interopérabilité entre des produits de différents fabricants. Cette spécification a pour objet d'améliorer l'interopérabilité des produits de surveillance vidéo IP provenant de différents fournisseurs.

A.2 Domaine d'application

En tant que première norme adhérent au modèle de service de la PSIA, le présent document définit les services PSIA obligatoires pour TOUTES les spécifications PSIA (les futures spécifications PSIA référenceront la présente Spécification sur les dispositifs média IP pour ces services obligatoires). De plus, il définit plusieurs services et ressources subordonnées spécifiques aux dispositifs média.

Tous les services et ressources obligatoires des services obligatoires et facultatifs sont complets dans la présente version de la spécification. Au contraire, plusieurs des ressources facultatives peuvent faire l'objet de certaines modifications dans la prochaine version de la spécification, sur la base des enseignements tirés lors de la mise en œuvre de cette première version. Ces ressources facultatives sont considérées comme préliminaires et sont indiquées comme telles dans les notes de définition des ressources.

Il faut que tous les services et ressources du service personnalisé soient considérés comme préliminaires, et la probabilité est forte qu'ils soient transférés à un autre service le moment venu. L'utilisation des ressources faisant réellement partie du service personnalisé n'est pas découragée, car tout est tenté pour offrir une compatibilité en amont à ces ressources existantes dans les spécifications suivantes. Par exemple, les services et ressources Custom/Event/Notification (Personnalisé/Événement/Notification) sont utilisables sous leur forme actuelle et peuvent être conservés tels quels, mais déplacés dans un autre service lors de la publication d'une spécification traitant des événements.

Il convient de soumettre les suggestions de correction de cette version de la spécification et les ajouts au protocole à la CEI ou directement au service Spécification média IP du Forum PSIA. Un fil servant à suivre les modifications pour la prochaine version se trouve à l'adresse suivante:

<http://www.psiaforums.org>

Merci de poster une réponse à ce fil pour soumettre une correction ou un ajout.

A.3 Définition du problème

Les applications de sécurité et/ou de gestion de réseau nécessitent la capacité à modifier les configurations et à contrôler les comportements des dispositifs média IP: caméras, codeurs, décodeurs, enregistreurs, etc. Cette fonctionnalité peut être obtenue en envoyant une requête HTTP(S) normalisée à l'unité. Le domaine d'application de cette spécification consiste à définir toutes les interfaces de programmation d'application HTTP(S) pour les dispositifs média et leurs fonctionnalités, c'est-à-dire pour régler/récupérer diverses configurations et contrôler le comportement des dispositifs.

A.4 Conformité

A.4.1 Généralités

Le présent article est conforme au modèle de service présenté dans le paragraphe précédent, qui décrit les méthodes utilisées pour la découverte de service et l'introspection. Les exigences pour les services et ressources obligatoires définies par ce modèle sont implicites, en plus des éventuelles exigences définies ici.

Les services requis définis ci-dessous sont les services fondamentaux pour toutes les spécifications vidéo IP HTTP et REST et sont destinés à être référencés par d'autres spécifications.

Les services facultatifs définis sont spécifiques aux dispositifs média IP.

A.4.2 Exigences de service

Le tableau ci-dessous décrit les exigences de service du modèle de service.

REQ	URL de service	Notes
✓	/	
✓	/System	
	/System/Storage	Tous les dispositifs média IP ne prennent pas en charge le stockage.
✓	/System/Network	
	/System/IO	
	/System/Audio	
	/System/Video	
	/System/Serial	
	/Diagnostics	
✓	/Security	
	/Security/AAA	
	/Streaming	
	/PTZ	
	/Custom/MotionDetection	
	/Custom/Event	

A.4.3 Exigences de ressource

A.4.3.1 Généralités

Les ressources qui suivent sont requises pour les services mis en œuvre.

A.4.3.2 Service racine

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

A.4.3.3 /System

REQ	Commande	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	Commande	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	Commande	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>	✓	✓		✓

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

A.4.3.3.5 /System/Video

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

A.4.3.4 /Diagnostics

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

A.4.3.5 /Security

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

A.4.3.5.1 /Security/AAA

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

A.4.3.6 /Streaming

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

A.4.3.7 /PTZ

REQ	Commande	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	✓			

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

A.4.3.9 /Custom/Event

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

A.5 Transmission en continu du média

A.5.1 Généralités

Il existe plusieurs méthodes de transmission de vidéo et audio en direct d'un dispositif média IP à un client.

A.5.2 Transmission en continu avec RTP et RTSP

A.5.2.1 Généralités

Un dispositif média IP doit prendre en charge la transmission en continu de contenu vidéo et audio au moyen de RTSP [RFC 2326], SDP [RFC 4566] et RTP [RFC 3550, RFC 3551].

RTP offre un cadre pour le transport en temps réel. RTP est flexible et a été utilisé avec succès pour transmettre les signaux téléphoniques via IP, les données vocales et audio en temps réel de téléconférences sur des liaisons à faible bande passante et la télévision haute définition sur des connexions à large bande. Sa flexibilité et son utilisation répandue ont fait de RTP une norme de facto depuis son origine.

RTP a été adopté comme norme par le groupe de travail IETF (Internet Engineering Task Force). RTP a également été adopté par l'Union internationale des télécommunications (UIT) comme partie de sa série de recommandations H.323.

RTP peut transmettre des données média sur des réseaux à diffusion unique et à diffusion multiple. Tel que défini, RTP se concentre sur la transmission en continu et laisse le contrôle de transmission et l'établissement de sessions à d'autres protocoles normalisés qui sont indiqués ci-dessous. RTP est délibérément incomplet: des profils supplémentaires spécifient des algorithmes et des cadres pour la lecture des médias et la régénération d'horodatage, la synchronisation entre les flux média, la dissimulation d'erreur et le contrôle de correction ou d'encombrement. RTP ne spécifie pas non plus les mappings entre la charge utile et les types de média.

RTP repose sur deux principes importants: l'encadrement au niveau de l'application et le principe de bout en bout. L'encadrement au niveau de l'application, tel qu'il s'applique au transport de médias, implique qu'il convient que le protocole de transmission fasse un minimum de suppositions sur les exigences des données transmises et laisse à l'application (expéditeur et récepteurs) la mise en trame (en paquets) du contenu et la gestion des transmissions non fiables. Le principe de bout en bout implique que l'intelligence se trouve chez l'expéditeur et le récepteur. Le réseau est considéré comme un système de transport de paquets sans état, "stupide". Ces deux principes combinés donnent un cadre unifié au transport audio/vidéo en temps réel, qui satisfait directement la plupart des applications tout en étant malléable pour les applications qui touchent à ses limites.

RTSP est un protocole de contrôle conçu pour les sessions multimédia. RTSP peut agir comme un "contrôle de réseau à distance" pour les serveurs média (dont les équipements de surveillance). La syntaxe et le fonctionnement de RTSP sont similaires à ceux de HTTP.

RTSP définit un moyen de délivrer une transmission RTP en continu à l'aide de TCP. Cela peut être utile pour transmettre des données dans des situations où aucune perte ne peut être tolérée, par exemple pour télécharger un clip vidéo ou transmettre en continu des données importantes.

SDP est un protocole qui décrit une session média. Du fait du format d'une description de session, certaines informations sont toujours nécessaires. SDP est utilisé pour acheminer les adresses de transport sur lesquelles le média circule, le format du média, les formats de charge utile RTP correspondants, le profil et les heures de la session ainsi que la durée du média.

RTSP et SDP se chevauchent: certaines informations disponibles dans la description de session le sont également via les commandes RTSP.

A.5.2.2 Utilisation de RTP et RTCP

Il est particulièrement recommandé que les dispositifs média IP mettent en œuvre l'envoi de rapports d'expéditeur RTCP afin de faciliter la synchronisation temporelle et à des fins de

diagnostic du réseau et de génération de rapport. Le rapport d'expéditeur doit contenir un horodatage NTP complet calé sur le 1er janvier 1900, avec son horodatage RTP correspondant.

Il est particulièrement recommandé que les dispositifs média IP envoient un paquet RTCP BYE lors de leur déconnexion d'une session, même en diffusion unique. Ces informations permettent à un client de faire la distinction entre une déconnexion volontaire et une déconnexion involontaire.

A.5.2.3 Utilisation de RTSP et SDP

A.5.2.3.1 URI de requête média

Un dispositif média IP rend les voies de transmission en continu accessibles en RTSP via un URI RTSP associé. Cet URI se présente de la manière suivante:

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

Un URI RTSP est constitué d'un chemin de base et d'un ensemble de paires nom-valeur de paramètres.

Pour la transmission de contenu en direct, les URI RTSP se présentent sous la forme suivante:

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

où le chemin correspond à la ressource REST pour une voie donnée de transmission en continu, comme défini en 7.11.3. Le jeu de paramètres valides correspond aux chaînes de requête et à leurs types selon 7.11.3. Cette correspondance est spécifiée afin de permettre l'introspection sur le jeu de paramètres de requête pris en charge pour une voie donnée de transmission en continu.

Exemple:

```
rtsp://192.168.99.11:554/Streaming/channels/123456?videoCodecType=mjpeg&rotationDegree=90
```

A.5.2.3.2 Mise en œuvre minimale

Le numéro de port par défaut d'un serveur RTSP IPMD est 554.

Tous les clients et le serveur doivent mettre en œuvre toutes les caractéristiques requises de la mise en œuvre RTSP minimale décrite à l'Annexe D de la norme RFC 2326.

La méthode DESCRIBE doit être prise en charge et doit prendre en charge SDP comme format de description, conformément à l'Annexe C de la norme RFC 2326.

Le profil RTP/AVP défini dans la RFC 3551 doit être pris en charge. Pour les requêtes SETUP, les en-têtes "Transport", "client_port", "server_port", "source", "ssrc" et "RTP-Info" doivent être prises en charge.

Si la diffusion multiple est prise en charge, les en-têtes "destination", "port" et "ttl" doivent être prises en charge.

A.5.2.3.3 Protocoles de transport pris en charge

Il est nécessaire que les serveurs RTSP pour dispositifs média IP prennent en charge UDP comme protocole de transmission.

Il est particulièrement recommandé que les serveurs RTSP pour dispositifs média IP prennent en charge TCP comme protocole de transmission. Si TCP est pris en charge, le serveur RTSP doit prendre en charge l'entrelacement des paquets RTP/RTCP sur la connexion RTSP TCP.

Il est particulièrement recommandé que les serveurs RTSP pour dispositifs média IP prennent en charge le transport à diffusion multiple UDP.

A.5.2.3.4 Vigilance (Keep Alive)

Le serveur RTSP doit indiquer le délai d'expiration de la session: il convient que les flux média d'un client qui n'informe pas le serveur qu'il est toujours vigilant dans cette limite temporelle (et régulièrement ensuite) soient interrompus.

Le serveur RTSP d'un dispositif média IP doit prendre en charge la méthode RTSP OPTIONS et les rapports du récepteur RTCP sur la vigilance. Il convient que le dispositif média traite toutes les commandes RTSP valides provenant du client comme une requête de "vigilance", et maintienne une session active en conséquence. La prise en charge de GET_PARAMETER pour les vigilances est facultative.

A.5.2.3.5 Authentification RTSP

Les serveurs RTSP pour dispositifs média IP doivent prendre en charge l'authentification HTTP de base. Il est particulièrement recommandé que les serveurs RTSP prennent en charge l'authentification de prétraitement HTTP tel que spécifié en RFC 2069.

Le jeu de noms d'utilisateur et de mots de passe valides requis pour accéder à une session RTSP est configuré sous /System/AAA. La granularité de l'autorisation est laissée à la mise en œuvre du dispositif.

A.5.2.4 Règles de mise en paquets RTP pour les codecs

Il est nécessaire que les règles de description et de mise en paquets des codecs permettent l'interopérabilité sur RTSP, SDP et RTP. Elles sont définies dans les documents RFC supplémentaires de l'IETF. Le tableau ci-dessous énumère quelques-uns des codecs communément utilisés pour les applications de vidéosurveillance:

Codec	Référence normative	Type Mime	RFC
Vidéo			
Motion JPEG	ISO/CEI 10918-1 UIT-T Rec. T.81	video/jpeg	RFC 2435
MPEG-2	ISO/CEI 13818-2	video/MPV	RFC 2250
MPEG-4 SP/ASP	ISO/CEI 14496-2:2004	video/MP4V-ES	RFC 3016 RFC 3640
H.264 AVC	ISO/CEI 14496-10:2005 UIT-T Rec. H.264:2005	video/H264	RFC 3984
H.264 SVC	ISO/CEI 14496-10 Amd 3 UIT-T Rec. H.264 Annexe G	video/H264-SVC	Projet IETF
Audio			
G.726 ^a	UIT-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 Couche II & III	ISO/CEI 11172-3:1993	audio/mpeg	RFC 2250
AAC	ISO/CEI 14496-3	audio/aac-lbr audio/aac-hbr	RFC 3640
a L'ordre des mots du code G.726 dans les paquets RTP est actuellement ambigu. Selon la recommandation UIT-T I.366.2, Annexe E, pour le transport ATM AAL2, il convient que les mots de code G.726 soient dans un ordre "gros-boutiste" (ordre d'octets dans le réseau) dans les octets de charge utile d'un paquet RTP. Toutefois, cela va à l'encontre de la dernière révision du projet de spécification de l'IETF sur les profils RTP audio et vidéo, qui spécifie un ordre "petit-boutiste" des paquets et délègue différents types MIME pour les paquets gros-boutistes ("AAL2-G726-32"). Il convient de noter que rien ne permet de détecter directement l'ordre des paquets à partir des données elles-mêmes. Certains fabricants de matériel ayant déjà mis en œuvre le transport RTP selon l'ordre le plus ancien, l'interopérabilité entre les dispositifs est problématique. Il apparaît toutefois que l'ordre "gros-boutiste" des paquets pour G.726 est largement accepté dans l'industrie. Les mises en œuvre doivent interpréter et générer le type MIME approprié à l'ordre des mots du code G.726 conformément à RFC 3551. Afin d'intégrer les équipements qui ne mettent pas correctement en œuvre cette norme, il doit être possible d'identifier la source grâce au champ d'en-tête RTSP "Server" ou au champ SDP "a=tool", afin de pouvoir modifier l'ordre des mots du code pour une transmission ou un décodage ultérieur.			

Pour la mise en paquets RTP, il faut que tous les codecs supplémentaires soient conformes au format de charge utile défini dans une spécification IETF ou un projet de spécification, selon le cas.

A.5.3 Transmission en continu à l'aide du modèle push de serveur HTTP

Il est particulièrement recommandé qu'un dispositif média IP prenne en charge la transmission en continu de contenu vidéo et audio via une connexion push de serveur HTTP. Voir /Streaming/channels/ID/http dans le A.7.10.5 pour une description de la transmission en continu HTTP.

A.6 Types de données communes

A.6.1 Généralités

Les blocs de données XML décrits dans le présent document contiennent des annotations qui décrivent les propriétés du champ. Pour une définition complète, voir les définitions de schéma XML.

Les informations qui suivent sont insérées dans les commentaires pour décrire les données transportées dans le champ:

Annotation	Description
req	Champ requis.
opt	Champ facultatif. Pour les données chargées sur le dispositif, si le champ est

Annotation	Description
	présent, mais que le dispositif ne le prend pas en charge, il convient de l'ignorer.
dep	Ce champ est requis en fonction de la valeur d'un autre champ.
ro	Lecture seule. Pour les données XML qui sont à la fois lues et écrites sur le dispositif, ce champ est présent uniquement dans le XML retourné depuis le dispositif. Si ce champ est présent dans le XML chargé sur le dispositif, il convient de l'ignorer.
wo	Écriture seule. Ce champ est présent uniquement dans le XML qui peut être chargé sur le dispositif. Il convient que ce champ ne soit jamais présent dans les données retournées depuis le dispositif. [Il sert à charger les mots de passe].
xs:<type>	Type défini dans XML Schema Part 2: Datatypes Second Edition, voir http://www.w3.org/TR/xmlschema-2

À noter que les structures XML qui sont facultatives peuvent comporter des champs obligatoires. Cela signifie que le bloc XML entier est facultatif, mais que s'il est présent, les champs requis sont obligatoires.

A.6.2 Types intégrés

Type	Description
Débit en bauds	Valeur numérique positive indiquant le débit de transmission des données en symboles par seconde. Valeur ≥ 0 . Exemple: 9 600
Couleur	Triplet RGB en format hexadécimal (3 octets) sans "0x" précédent. Exemple: "FF00FF"
Coordonnées	Valeur numérique positive en pixels. Une paire de coordonnées de 0,0 (x,y) indique le coin inférieur gauche de l'image vidéo. Valeur ≥ 0 . La valeur maximum dépend de la résolution vidéo.
FPS	Taux de trame multiplié par 100. Exemple: 2 500 [PAL]
ID	ID d'après le modèle de service.
Adresse IPv4	La notation est xxx.xxx.xxx.xxx Exemple: 3.137.217.220
Adresse IPv6	La notation est xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx selon la notation CIDR. Exemple: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
MAC	Adresse MAC La notation est aa:bb:cc:dd:ee:ff avec 6 octets hexadécimaux.
TTL	Valeur numérique positive indiquant le nombre de bonds (routeurs) par lesquels le trafic est autorisé à passer avant d'expirer. Valeur ≥ 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

- En fonction de la valeur de <addressingFormatType>, le champ <hostName> ou l'adresse IP est utilisé pour localiser le serveur NTP.
- L'utilisation des adresses IPv4 ou IPv6 dépend de la valeur du champ <ipVersion> dans /System/Network/interfaces/ID/ipAddress.

A.6.4 TimeBlockList

<TimeBlockList> contient un jeu XML de <TimeBlock> qui définissent un ensemble de plages temporelles.

```
<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

- Si <dayOfWeek> n'est pas présent, le bloc temporel est valide tous les jours. Deux <TimeBlock> dans la même liste ne donnent pas le même <dayOfWeek>.
- Si la balise <bitString> existe, <TimeRange> n'est pas inclus, et inversement.
- Le champ <bitString> peut être utilisé pour réduire la quantité de XML transférables requis. Le champ est une chaîne de 24 bits, dont chacun spécifie une heure de la journée. Le bit le plus à gauche est l'heure 0, et le bit le plus à droite l'heure 24. "1" indique que l'heure spécifiée est activée pour la détection et le déclenchement d'événement, "0" indiquant qu'elle ne l'est pas. Ainsi, il est nécessaire que tous les champs <bitString> contiennent 24 bits.

A.7 Détails des commandes de service**A.7.1 /System**

URI	/System		Type	Service
Fonction	Services système.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
Notes				

A.7.1.1 /System/reboot

URI	/System/reboot		Type	Ressource
Fonction	Redémarrer le dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
PUT			<ResponseStatus>	
Notes	Les données XML <ResponseStatus> sont retournées avant que le dispositif procède au redémarrage.			

A.7.1.2 /System/updateFirmware

URI	/System/updateFirmware		Type	Ressource
-----	------------------------	--	------	-----------

Fonction	Mettre à jour le microgiciel du dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
PUT			<ResponseStatus>
Notes	À l'issue de l'API, les données XML <ResponseStatus> sont retournées et le dispositif procède au redémarrage.		

A.7.1.3 /System/configurationData

URI	/System/configurationData	Type	Ressource
Fonction	La fonction permet d'obtenir ou d'établir les données de configuration pour le dispositif. Il s'agit de données opaques qui peuvent servir à enregistrer et restaurer la configuration du dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			Données opaques
PUT		Données opaques	<ResponseStatus>
Notes	Les données de configuration dépendent du dispositif, elles peuvent être en format binaire ou autre. Le client peut utiliser le champ d'en-tête HTTP <code>Accept</code> pour informer le serveur des formats acceptés par le client. Peut redémarrer le dispositif après l'application des données de configuration.		

A.7.1.4 /System/factoryReset

URI	/System/factoryReset	Type	Ressource
Fonction	Cette fonction sert à réinitialiser la configuration par défaut du dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
PUT	mode		<ResponseStatus>
Notes	Deux modes de réinitialisation sont pris en charge: "complet" réinitialise tous les paramètres et réglages du dispositif aux valeurs d'usine par défaut. "basique" réinitialise tous les paramètres et réglages du dispositif, sauf les valeurs de /System/Network et /System/Security. Le mode par défaut est "complet". Le dispositif peut être redémarré après sa réinitialisation.		

A.7.1.5 /System/deviceInfo

URI	/System/deviceInfo	Type	Ressource
Fonction	Cette fonction sert à obtenir ou établir des informations sur le dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<DeviceInfo>
PUT		<DeviceInfo>	<ResponseStatus>
Notes	Certains champs sont en lecture seule et sont susceptibles de ne pas être configurés. Si ces champs sont présents dans le bloc XML entrant, ils sont ignorés. Pour le <DeviceInfo> chargé sur le dispositif durant une opération PUT, tous les champs sont considérés comme étant facultatifs, et un champ qui n'est pas présent dans le XML entrant n'est pas modifié sur le dispositif. Cela permet de configurer les champs individuellement sans avoir à charger le bloc XML entier sur le dispositif. <deviceDescription> est une description du dispositif tel que défini dans la norme RFC 1213. <deviceLocation> est l'emplacement du dispositif tel que défini dans la norme RFC 1213. <systemContact> est l'information de contact pour le dispositif telle que définie		

dans la norme RFC 1213.
 <systemObjectID> est l'identifiant d'objet système défini dans la norme RFC 1213.

A.7.1.5.1 Bloc XML DeviceInfo

```
<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	Ressource
Fonction	Cette fonction permet d'obtenir une archive compressée d'informations de support pour le dispositif. Il faut que l'archive contienne au moins la configuration courante du dispositif et les fichiers journaux. D'autres éléments peuvent inclure des informations sur syslog et le système d'exploitation, des statistiques, etc.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			Données de support	
Notes	Le format de l'archive dépend du dispositif (tar, zip, etc.) Utiliser le champ d'en-tête http <code>Accept:</code> pour informer le serveur des formats acceptés par le client.			

A.7.1.7 /System/status

URI	/System/status			Type	Ressource
Fonction	Cette fonction sert à obtenir le statut du dispositif.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<DeviceStatus>		
Notes	Tous les champs de <DeviceStatus> peuvent être présents.				

A.7.1.7.1 Bloc XML DeviceStatus

```
<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	Ressource
Fonction	Accéder à l'information temporelle du dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour

GET			<Time>
PUT	timeMode localTime timeZone	<Time>	<ResponseStatus>
Notes	Si la chaîne de requête "localTime" est spécifiée avec une valeur, le bloc XML <Time> n'est pas requis en données entrantes. Si <timeMode> est défini sur "local", les champs <localTime> et <timeZone> sont requis. Le bloc <LocalTime> établit l'heure du dispositif. Si <timeMode> est défini sur "NTP", seul le champ <timeZone> est requis. L'heure du dispositif est réglée par synchronisation avec NTP.		

A.7.1.8.1 Bloc XML Time

```
<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	Ressource
Fonction	Accéder à l'information temporelle locale du dispositif.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			ISO 8601 Chaîne Date-Heure		
PUT		ISO 8601 Chaîne Date-Heure	<ResponseStatus>		
Notes	Une chaîne Date/Heure ISO 8601 est acceptée et retournée. Si la valeur date/heure est une zone temporelle, l'heure est convertie dans la zone temporelle locale du dispositif. Si l'heure du dispositif est sur "ntp", cette valeur n'a aucun effet.				

A.7.1.10 /System/time/timeZone

URI	/System/time/timeZone			Type	Ressource
Fonction	Accéder à la zone temporelle du dispositif.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			Chaîne zone temporelle		
PUT		Chaîne zone temporelle	<ResponseStatus>		
Notes	Les zones temporelles sont définies par les notations de zone temporelle de l'Article 8 de l'ISO/CEI 9945-1:2003. À noter que la valeur qui suit le +/- est la quantité de temps qu'il faut <i>ajouter</i> à l'heure locale pour obtenir un résultat UTC.				
	Exemple:				
	EST+5EDT01:00:00,M3.2.0/02:00:00,M11.1.0/02:00:00				
	Définit l'heure standard de l'Est comme "EST" avec un décalage GMT-5. L'heure d'été est appelée "EDT", est une heure plus tardive, commence le deuxième dimanche de mars à 2 h et se termine le premier dimanche de novembre à 2 h.				
	CET-1CEST01:00:00,M3.5.0/02:00:00,M10.5.0/03:00:00				
	Définit l'heure d'Europe centrale à GMT+1, avec une heure d'été de une heure plus tardive ("CEST") qui commence le dernier dimanche de mars à 2 h et se termine le dernier dimanche d'octobre à 3 h.				

A.7.1.11 /System/time/ntpServers

URI	/System/time/ntpServers		Type	Ressource
Fonction	Accéder aux serveurs NTP configurés pour le dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			< NtpServerList>	
PUT		<NtpServerList>	<ResponseStatus>	
POST		<NtpServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Si le <timeMode> est défini sur "NTP", les serveurs de cette liste sont utilisés pour synchroniser l'heure système du dispositif.			

A.7.1.11.1 Bloc XML NtpServerList

```
<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	Ressource
Fonction	Accéder à un serveur NTP configuré pour le dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<NtpServer>	
PUT		<NtpServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	En fonction de la valeur de <addressingFormatType>, le champ <hostName> ou l'adresse IP est utilisé pour localiser le serveur NTP. L'utilisation des adresses IPv4 ou IPv6 dépend de la valeur du champ <ipVersion> dans /System/Network/interfaces/ID/ipAddress.			

A.7.1.12.1 Bloc XML NtpServer

```
<NTPServer version="1.0" xmlns="urn:psialliance-org">
  <id <!-- reg, 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	Ressource
Fonction	Cette fonction sert à accéder aux paramètres de journalisation.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<Logging>	
PUT		<Logging>	<ResponseStatus>	
Notes	Le dispositif conserve un journal des <maxEntries> qui peuvent être configurées			

et demandées.

A.7.1.14.1 Bloc XML Logging

```
<Logging version="1.0" xmlns="urn:psialliance-org">
  <LogTrigger
    <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	Ressource
Fonction	Cette fonction sert à accéder au journal de messages.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<LogMessageList>	
Notes	Les dispositifs peuvent définir des champs de journalisation supplémentaires pour plus d'informations. Le journal de messages est en lecture seule.			

A.7.1.15.1 Bloc XML LogMessageList

```
<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
Fonction	Cette fonction sert à accéder aux paramètres de stockage.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
Notes	Service de stockage.			

A.7.2.1 /System/Storage/volumes

URI	/System/Storage/volumes		Type	Service
Fonction	Cette fonction sert à accéder aux volumes et fichiers de stockage sur un dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<StorageVolumeList>	
Notes	Le stockage est organisé en volumes. Chaque volume est un espace de stockage individuel. La création et la configuration des volumes ne font pas partie du domaine d'application de la présente interface, les informations sont donc			

	disponibles uniquement en lecture seule.
--	--

A.7.2.1.1 Bloc XML StorageVolumeList

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

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

URI	/System/Storage/volumes/ <i>ID</i>		Type	Ressource
Fonction	Cette fonction sert à accéder à un volume de stockage particulier au moyen de son ID.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<StorageVolume>	
Notes	Les informations sur le volume sont en lecture seule sur cette interface.			

A.7.2.2.1 Bloc XML StorageVolume

```
<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/ <i>ID</i> /status		Type	Ressource
Fonction	Cette fonction sert à demander le statut d'un stockage particulier.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<StorageVolumeStatus>	
Notes	Demande le statut du volume. Actuellement, seule la quantité d'espace libre est retournée. Les dispositifs peuvent étendre le bloc XML pour permettre de demander des informations supplémentaires.			

A.7.2.3.1 Bloc XML StorageVolumeStatus

```
<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/ <i>ID</i> /format	Type	Ressource
Fonction	Formater un volume de stockage.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
PUT			<ResponseStatus>
Notes	Le formatage peut prendre du temps.		

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

URI	/System/Storage/volumes/ <i>ID</i> /files	Type	Ressource
Fonction	Obtenir la liste de fichiers stockés dans un stockage particulier.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<StorageFileList>
DELETE			<ResponseStatus>
Notes	Les fichiers de stockage sont en lecture seule, sauf pour la possibilité d'effacement. DELETE supprime tous les fichiers du volume de stockage.		

A.7.2.5.1 Bloc XML StorageFileList

```
<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/ <i>ID</i> /files/ <i>ID</i>	Type	Ressource
Fonction	Accéder à un fichier et le manipuler.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<StorageFile>
DELETE			<ResponseStatus>
Notes	DELETE supprime un fichier particulier du volume de stockage.		

A.7.2.6.1 Bloc XML StorageFile

```
<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/ <i>ID</i> /data	Type	Ressource
Fonction	Cette fonction sert à accéder aux données d'un fichier particulier.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			Fichier de données brutes
Notes	Les données audio/vidéo peuvent être chiffrées en fonction des spécifications de chaque dispositif. Le format vidéo/audio dépend des fonctionnalités et des configurations du dispositif. Le client peut utiliser la chaîne d'en-têtes http Accept: pour négocier le format de données.		

A.7.3 /System/Network

URI	/System/Network		Type	Service
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
Notes	Configuration du réseau.			

A.7.3.1 /System/Network/interfaces

URI	/System/Network/interfaces		Type	Ressource
Fonction	Accéder aux interfaces de réseau du dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<NetworkInterfaceList>	
Notes	En tant que ressources systèmes câblées, les interfaces de réseau ne peuvent être ni créées, ni détruites.			

A.7.3.1.1 Bloc XML NetworkInterfaceList

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<NetworkInterface>	
PUT		<NetworkInterface>	<ResponseStatus>	
Notes	Accéder à une interface de réseau particulière.			

A.7.3.2.1 Bloc XML NetworkInterface

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IPAddress>	
PUT		<IPAddress>	<ResponseStatus>	
Notes	Si <addressingType> est dynamique, il n'est pas nécessaire d'inclure les champs en dessous. Si <addressingType> est dynamique, un client DHCP est utilisé pour le dispositif. Si <addressingType> est statique, l'adresse IP du dispositif est configurée			

	manuellement, et les champs de passerelle et DNS sont facultatifs. Si <addressingType> se réfère à APIPA, l'adresse IP du dispositif est automatiquement configurée sans DHCP. Dans ce cas, les champs de passerelle et DNS sont facultatifs. L'utilisation de <ipAddress> ou <ipv6Address> dans les champs est dictée par le champ <ipVersion>. Si <ipVersion> est "v4", les champs <ipAddress> sont utilisés, si <ipVersion> est "v6", les champs <ipv6Address> sont utilisés. La notation de <subnetMask> est "xxx.xxx.xxx.xxx". <IPv6Address> est "xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx" avec la notation CIDR.
--	---

A.7.3.3.1 Bloc XML IPAddress

```

<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	Ressource
Fonction	Accéder aux paramètres du réseau sans fil.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<Wireless>	
PUT		<Wireless>	<ResponseStatus>	
Notes	Si le champ <securityMode> est "WEP", il faut que le bloc <WEP> soit inclus. Si le champ <securityMode> est "WPA" ou "WPA2-personal", il faut que le bloc <WPA> soit inclus. Si le mode de sécurité "WPA" ou "WPA2-enterprise" est utilisé, il faut utiliser le bloc <WPA> et régler les paramètres relatifs à 802.1x à l'aide de la ressource /System/Network/interfaces/ID/ieee802.1x. <channel> correspond à un numéro de voie sans fil 802.11g ou à "auto" pour autoconfiguration. <wmmEnabled> active 802.11e, QoS pour les réseaux IEEE 802.11 (Wi-Fi Multimédia) <defaultTransmitKeyIndex> indique quelle clé de chiffrement est utilisée pour la sécurité WEP. <encryptionKey> est la clé de chiffrement WEP au format hexadécimal. <sharedKey> est la clé prépartagée utilisée en WPA.			

A.7.3.4.1 Bloc XML Wireless

```
<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	Ressource
Fonction			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<IEEE802_1x>
PUT		<IEEE802_1x>	<ResponseStatus>
Notes	Si la balise <authenticationProtocolType> correspond à "EAP-TTLS", il faut que la balise <innerTTLSAuthenticationMethod> soit incluse. Si <authenticationProtocolType> correspond à "EAP-PEAP" ou "EAP-FAST", il faut que la balise <innerEAPProtocolType> soit incluse. La balise <anonymousID> est facultative. Si <authenticationProtocolType> correspond à "EAP-FAST", il faut que la balise <autoPACProvisioningEnabled> soit incluse. <anonymousID> est l'ID anonyme facultatif à utiliser à la place de <userName>.		

A.7.3.5.1 Bloc XML IEEE802_1x

```
<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	Ressource
Fonction					
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<IPFilter>		
PUT		<IPFilter>	<ResponseStatus>		
Notes	Le champ <permissionType>, s'il est inclus comme enfant direct de <IPFilter>, fait office de configuration de niveau système et s'applique à toutes les entrées de <IPFilterAddress> en écrasant la valeur fournie dans un bloc <IPFilterAddress> particulier.				

A.7.3.6.1 Bloc XML IPFilter

```

<IPFilter version="1.0" xmlns="urn:psalliance-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/ <i>ID</i> /ipFilter/filterAddresses			Type	Res- source
Fonction					
Méthodes	Chaîne(s) de requête		Données entrantes	Résultat de retour	
GET				<IPFilterAddressList>	
PUT			<IPFilterAddressList>	<ResponseStatus>	
POST			<IPFilterAddress>	<ResponseStatus>	
DELETE				<ResponseStatus>	
Notes	La liste d'adresses de filtrage IP permet d'ajouter et de retirer des adresses de la liste ou de charger la liste entière en une fois.				

A.7.3.7.1 Bloc XML IPFilterAddressList

```

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

```

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

URI	/System/Network/interfaces/ID/ipFilter/filterAddresses/ID		Type	Res-source
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IPFilterAddress>	
PUT		<IPFilterAddress>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Si la balise <permissionType> n'est pas incluse comme enfant direct de <IPFilter>, il faut que la balise <permissionType> soit incluse pour chaque <IPFilterAddress>. Étant donné que l'ordre des filtres peut modifier le comportement, le filtrage est appliqué dans l'ordre en commençant par le premier <IPFilterAddress> de la liste. Le champ <bitMask> est appliqué à l'adresse IP correspondante pour identifier une plage d'adresses. Il indique le nombre de bits '1' utilisés pour masquer l'adresse. Par exemple: '24' correspond à un masque de sous-réseau de 255.255.255.0 et '32' à un masque de sous-réseau de 255.255.255.255 (une adresse IP unique) pour IPv4. Si <addressFilterType> se réfère à "mask", il faut que le bloc <AddressMask> soit inclus à la place du bloc <AddressRange>. S'il se réfère à "range", il faut que le bloc <Range> soit inclus à la place du bloc <AddressMask>. L'utilisation des adresses IPv4 ou IPv6 dépend de la valeur du champ <ipVersion> dans /System/Network/interfaces/ID/ipAddress.			

A.7.3.8.1 Bloc XML IPFilterAddress

<pre> <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> </pre>	
--	--

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

URI	/System/Network/interfaces/ID/snmp	Type	Ressource
Fonction	Paramètres SNMP.		

Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<SNMP>
PUT		<SNMP>	<ResponseStatus>
Notes	Il faut qu'au moins l'un des blocs <SNMPv2c> ou <SNMPAdvanced> soit inclus.		

A.7.3.9.1 Bloc XML SNMP

```
<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	Ressource
Fonction	Paramètres V2C SNMP.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<SNMPv2c>		
PUT		<SNMPv2c>	<ResponseStatus>		
Notes	La configuration SNMP v2c comprend les paramètres de configuration SNMP et un jeu de récepteurs de déROUTement SNMP. SNMP v2c comprend SNMP v2 sans le nouveau modèle de sécurité controversé SNMP v2, utilisant à la place le schéma de sécurité communautaire simple de SNMP v1.				

A.7.3.10.1 Bloc XML SNMPv2c

```
<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	Ressource
Fonction	Liste des récepteurs de déROUTement SNMP.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<SNMPTrapReceiverList>	
PUT		<SNMPTrapReceiverList>	<ResponseStatus>	
POST		<SNMPTrapReceiver>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	On peut effectuer PUT sur la liste entière en même temps.			

A.7.3.11.1 Bloc XML SNMPTrapReceiverList

```
< 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	Res- source
-----	---	------	----------------

Fonction	Information des récepteurs de déROUTement SNMP.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<SNMPTrapReceiver>
PUT		<SNMPTrapReceiver>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Il faut que le format <communityString> soit conforme à la norme SNMPv2c .		

A.7.3.12.1 Bloc XML SNMPTrapReceiver

```
<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	Ressource
Fonction	Paramètres SNMP avancés.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<SNMPAdvanced>
PUT		<SNMPAdvanced>	<ResponseStatus>
Notes	<localEngineID> est une chaîne hexadécimale indiquant l'ID du moteur du dispositif local. <authenticationNotificationEnabled> indique si la notification d'échec de l'authentification SNMP est activée sur le dispositif. <SNMPNotificationFilterList> est une liste de déROUTements de filtrage basée sur OID.		

A.7.3.13.1 Bloc XML SNMPAdvanced

```
< 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	Ressource
Fonction	Utilisateurs SNMP.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<SNMPUserList>
PUT		<SNMPUserList>	<ResponseStatus>
POST		<SNMPUser>	<ResponseStatus>

DELETE			<ResponseStatus>
Notes	Définit l'ensemble d'utilisateurs SNMP et leurs permissions.		

A.7.3.14.1 Bloc XML SNMPUserList

```
< 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	Ressource
Fonction	Paramètres utilisateurs SNMP.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<SNMPUser>	
PUT		<SNMPUser>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<remoteEngineID> indique l'entité SNMP distante à laquelle l'utilisateur est connecté. <snmpAuthenticationMethod> indique la méthode d'authentification utilisée. <snmpAuthenticationKey> définit la clé d'authentification si le chiffrement est utilisé pour <snmpAuthenticationMethod>. <snmpAuthenticationPassword> mot de passe facultatif utilisé pour calculer la valeur <snmpAuthenticationKey> si le chiffrement est utilisé pour <snmpAuthenticationMethod>. <snmpPrivacyMethod> indique si les messages sont protégés de la divulgation et, si c'est le cas, le type de protocole de confidentialité utilisé. <snmpPrivacyKey> définit la clé de confidentialité si le chiffrement est utilisé pour <snmpPrivacyMethod>. <snmpPrivacyPassword> mot de passe facultatif utilisé pour calculer la valeur <snmpPrivacyKey> si des chiffrements sont utilisés pour <snmpPrivacyMethod>.			

A.7.3.15.1 Bloc XML SNMPUser

```
< 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	Ressource
Fonction	Filtres de notification SNMP				

Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<SNMPNotificationFilterList>
PUT		<SNMPNotificationFilterList>	<ResponseStatus>
POST		<SNMPNotificationFilter>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Gère une liste de filtres de notification pour SNMP v2 ou v3.		

A.7.3.16.1 Bloc XML SNMPNotificationFilterList

```
<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	Ressource
Fonction	Paramètres de filtres de notification SNMP.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<SNMPNotificationFilter>	
PUT		<SNMPNotificationFilter>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<oidSubtree> spécifie l'OID pour lequel les notifications sont envoyées ou bloquées. <filterAction> indique si les notifications relatives à l'OID sont envoyées aux destinataires de déROUTement.			

A.7.3.17.1 Bloc XML SNMPNotificationFilter

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

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

URI	/System/Network/interfaces/ID/snmp/advanced/notificationReceivers			Type	Res- source
Fonction	Récepteurs de notification SNMP				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<SNMPNotificationReceiverList>		
PUT		<SNMPNotificationReceiverList>	<ResponseStatus>		
POST		<SNMPNotificationReceiver>	<ResponseStatus>		
DELETE			<ResponseStatus>		

Notes	Gère la liste de récepteurs de notification pour SNMP v2 ou v3.
--------------	---

A.7.3.18.1 Bloc XML SNMPNotificationReceiverList

```
<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	Res-source
Fonction	Paramètres de récepteurs de notification SNMP.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<SNMPNotificationReceiver>		
PUT		<SNMPNotificationReceiver>	<ResponseStatus>		
DELETE			<ResponseStatus>		
Notes	<notificationType> indique si cette entrée de récepteur est pour un détournement ou une information. Il faut que <userID> corresponde à un ID d'utilisateur dans /System/Network/interfaces/ID/snmp/advanced/users/ID. <securityType> définit le niveau de sécurité associé à l'utilisateur. L'option "authentification" authentifie les messages SNMP et assure que l'origine est authentifiée. L'option "confidentialité" authentifie et chiffre les messages SNMP. <filterName> associe un filtre si <filterEnabled> est vrai. <timeout> indique le temps d'attente (en secondes) avant que le dispositif ne renvoie les informations. <retries> indique le nombre de fois où le dispositif renvoie une requête d'information.				

A.7.3.19.1 Bloc XML SNMPNotificationReceiver

```
< 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	Res-source
Fonction	Paramètres SNMP v3.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<SNMPAdvanced>		

PUT		<SNMPAdvanced>	<ResponseStatus>
Notes	Cette ressource est un alias vers /System/Network/interfaces/ID/snmp/advanced. Les balises <snmpAuthenticationPassword> et <snmpPrivacyPassword> sont éventuellement utilisées si la mise en œuvre du dispositif choisit de calculer les clés correspondantes sur la base d'un mot de passe (comme dans la norme RFC 3414). Dans ce cas, <snmpAuthenticationKey> et <snmpPrivacyKey> peuvent ou non être incluses. La balise <localEngineID> est utilisée pour les messages "détournement" et la balise <remoteEngineID> est utilisée pour les messages "information".		

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

URI	/System/Network/interfaces/ID/qos	Type	Ressource
Fonction	Cette fonction sert à établir le paramètre QoS pour le dispositif.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<QoS>
PUT		<QoS>	<ResponseStatus>
Notes	Il faut qu'au moins l'un des blocs <CoSList> ou <DSCPList> soit inclus.		

A.7.3.21.1 Bloc XML QoS

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

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

URI	/System/Network/interfaces/ID/qos/cos	Type	Ressource
Fonction	Paramètres de classe de service (CoS).		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<CoSList>
PUT		<CoSList>	<ResponseStatus>
POST		<CoS>	<ResponseStatus>
DELETE			<ResponseStatus>
Notes	Une liste de classes de blocs de paramètres de service est spécifiée pour chaque type de trafic: gestion de dispositif, commande et contrôle, transmission vidéo et audio en continu. Les dispositifs peuvent étendre le jeu de types de trafic.		

A.7.3.22.1 Bloc XML CoSList

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

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

URI	/System/Network/interfaces/ID/qos/cos/ID	Type	Ressource
Fonction	Paramètres de classe de service.		
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour
GET			<CoS>
PUT		<CoS>	<ResponseStatus>

DELETE		<ResponseStatus>
Notes	<trafficType> détermine à quel type de trafic les paramètres s'appliquent.	

A.7.3.23.1 Bloc XML CoS

```
<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	Ressource
Fonction	Paramètres de services différenciés (DiffServ).			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<DSCPList>	
PUT		<DSCPList>	<ResponseStatus>	
POST		<DSCP>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Une liste de blocs de paramètres DSCP est spécifiée pour chaque type de trafic: gestion de dispositif, commande et contrôle, transmission vidéo et audio en continu. Les dispositifs peuvent étendre le jeu de types de trafic.			

A.7.3.24.1 Bloc XML DSCPList

```
<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	Ressource
Fonction	Paramètres d'entrée DSCP.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<DSCP>	
PUT		<DSCP>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	<trafficType> détermine à quel type de trafic les paramètres s'appliquent.			

A.7.3.25.1 Bloc XML DSCP

```
< 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	Ressource
Fonction	Paramètres de découverte de dispositif.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<Discovery>	
PUT		<Discovery>	<ResponseStatus>	
Notes	L'utilisation des adresses IPv4 ou IPv6 dépend de la valeur du champ <ipVersion> dans /System/Network/interfaces/ID/ipAddress. <portNo> est le numéro de port pour l'adresse de découverte à multidiffusion. <ttl> est la durée de vie des paquets de découverte à multidiffusion.			

A.7.3.26.1 Bloc XML Discovery

```
<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	Ressource
Fonction	Paramètres syslog.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<Syslog>	
PUT		<Syslog>	<ResponseStatus>	
Notes	Configure les paramètres système.			

A.7.3.27.1 Bloc XML Syslog

```
<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	Ressource
Fonction	Liste de serveur syslog.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<SyslogServerList>	
PUT		<SyslogServerList>	<ResponseStatus>	
POST		<SyslogServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	Gère un jeu de serveurs syslog qui reçoivent les notifications de journalisation.			

A.7.3.28.1 Bloc XML SyslogServerList

```
<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	Ressource
Fonction	Paramètres de serveur syslog.			
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<SyslogSever>	
PUT		<SyslogServer>	<ResponseStatus>	
DELETE			<ResponseStatus>	
Notes	En fonction de la valeur de <addressingFormatType>, le champ <hostName> ou l'adresse IP est utilisé pour localiser le serveur NTP. L'utilisation des adresses IPv4 ou IPv6 dépend de la valeur du champ <ipVersion> dans /System/Network/interfaces/ID/ipAddress. <facilityType> indique l'installation de stockage des messages syslog. Voir RFC3164. <severity> indique la sévérité minimale de journal pour laquelle envoyer un message syslog. Voir RFC 3164.			

A.7.3.29.1 Bloc XML SyslogServer

```
< 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 Exemples

A.7.3.30.1 Exemple: Obtention des paramètres réseau

```
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>infrastructure</wirelessNetworkMode>
        <WirelessSecurity>
            <securityMode>WPA-personal</securityMode>
            <WPA>
                <algorithmType>AES</algorithmType>
                <sharedKey>ac34587bc8a8fff7a</sharedKey>
            </WPA>
        </WirelessSecurity>
    </Wireless>
</NetworkInterface>
</NetworkInterfaceList>

```

A.7.3.30.2 Exemple: Réglage de l'adresse IP

```

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
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOPortList>	
Notes	Il faut que l'allocation des ID entre les ports d'entrée et de sortie soit unique.			

A.7.3.31.1 Bloc XML IOPortList

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOPortStatusList>	
Notes	<ioPortID> se réfère à /System/IO/inputs/ID ou /System/IO/outputs/ID. Les ID de port sont garantis uniques sur les ports d'entrée et de sortie. <ioState> indique si le port d'entrée est actif ou inactif. Dans la plupart des applications, un signal haut est considéré comme actif.			

A.7.3.32.1 Bloc XML IOPortStatus

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOInputPortList>	
Notes	Les entrées ES sont câblées, ce qui signifie que les entrées sont attribuées statiquement par le dispositif et ne peuvent être ni créées ni supprimées.			

A.7.3.33.1 Bloc XML IOInputPortList

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOInputPort>	
PUT		<IOInputPort>	<ResponseStatus>	
Notes	<triggeringType> indique les conditions du signal pour déclencher le port d'entrée. Rising/Falling se réfère à un front croissant/décroissant d'un signal. High/Low se déclenche en continu pendant la durée d'un signal d'entrée haut/bas.			

A.7.3.34.1 Bloc XML IOInputPort

```
< 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	Ressource
Fonction					
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<IOPortStatus>		
Notes	Voir /System/IO/status pour une explication des champs.				

A.7.3.36 /System/IO/outputs

URI	/System/IO/outputs		Type	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOOutputPortList>	
Notes	Les sorties ES sont câblées, ce qui signifie que les entrées sont attribuées statiquement par le dispositif et ne peuvent être ni créées ni supprimées.			

A.7.3.36.1 Bloc XML IOOutputPortList

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

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

URI	/System/IO/outputs/ <i>ID</i>		Type	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET			<IOOutputPort>	
PUT		<IOOutputPort>	<ResponseStatus>	
Notes	<PowerOnState> définit la configuration du port de sortie lorsque le dispositif est sous tension. <defaultState> est le signal par défaut du port de sortie lorsqu'il n'est pas déclenché. <outputState> est le signal du port de sortie lorsqu'il est déclenché. L'impulsion entraîne l'envoi d'un signal par le port de sortie (opposé au <defaultState>) pendant une durée spécifiée par la balise <pulseDuration>. <pulseDuration> est la durée d'un signal d'impulsion du port de sortie lorsqu'il est déclenché. Il faut qu'il soit inclus si le <outputState> est "impulsion". <actionMapping> est utilisé dans les interfaces qui permettent la configuration de signaux "On" et "Off" pour "Haut" et "Bas".			

A.7.3.37.1 Bloc XML IOOutputPort

```
<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	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
PUT	outputState pulseDuration	<IOPortData>	<ResponseStatus>	
Notes	Le port de sortie ES est basculé en conséquence en signal haut ou bas. Si le <outputState> se réfère à l'impulsion, alors il faut que la balise <pulseDuration> soit incluse, et le port de sortie est déclenché à l'état spécifié pendant la durée spécifiée par <pulseDuration>.			

A.7.3.38.1 Bloc XML IOPortData

```

< 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	Ressource
Fonction	Demander le statut d'un port de sortie.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<IOPortStatus>		
Notes	Voir /System/IO/status pour une explication des champs.				

A.7.3.40 Exemples de ports ES

A.7.3.40.1 Exemple: Réglage du déclenchement du port ES

NOTE L'exemple qui suit requière que la détection d'événement de port d'entrée et le déclenchement de port de sortie soient activés et planifiés à l'avance à l'aide de /Custom/Event/triggers et /Custom/Event/schedule.

Les commandes qui suivent configurent un port d'entrée de dispositif et deux ports de sortie de dispositif (le nombre de ports ES dépend du dispositif) de la manière suivante:

- Le port d'entrée 111 déclenche un événement en continu (spécifié dans l'exemple de A.7.13.17) lorsque le signal d'entrée est haut. Il convient que le port d'entrée cesse de déclencher cet événement lorsque le signal d'entrée revient à bas.
- Le port de sortie 222 a un signal bas par défaut lorsqu'il n'est pas déclenché. Lorsqu'il est déclenché, il passe à un signal haut. Il convient que le port repasse automatiquement à un signal bas lorsque le déclenchement cesse, mais dans le cas où un dispositif ne peut pas

prendre cette caractéristique en charge, le port peut être réinitialisé manuellement (voir A.7.13.17).

- Le port de sortie 333 a un signal bas par défaut lorsqu'il n'est pas déclenché. Lorsqu'il est déclenché, il envoie une "impulsion" du signal opposé - haut dans ce cas - pendant trois secondes, puis repasse à un signal bas.

```
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 Exemple: Déclencher et réinitialiser manuellement un port de sortie

Utiliser les commandes suivantes pour établir manuellement un signal bas. À noter que cette caractéristique n'a aucun effet sur la détection et le déclenchement futurs, par exemple, si le port de sortie 1 est automatiquement déclenché plus tard, cela remplacera le comportement établi ici.

```
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>
```

ou, même chose sans charge utile XML:

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

A.7.4 /System/Audio

URI	/System/Audio			Type	Service
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
Notes	Service audio.				

A.7.4.1 /System/Audio/channels

URI	/System/Audio/channels		Type	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET		Néant	<AudioChannelList>	
Notes	Les entrées étant des ressources définies par la configuration matérielle du dispositif, les voies audio ne peuvent être ni créées ni supprimées. Il convient de considérer la numérotation ou les valeurs d'ID de manière arbitraire et en fonction du dispositif.			

A.7.4.1.1 Bloc XML AudioChannelList

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

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

URI	/System/Audio/channels/ <i>ID</i>		Type	Ressource
Fonction				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour	
GET		Néant	<AudioChannel>	
PUT		<AudioChannel>	<ResponseStatus>	
Notes	<audioMode> est le mode bidirectionnel simultané pour la transmission audio entre le client et le dispositif média. <microphoneSource> indique si le microphone du dispositif est interne ou externe. <microphoneVolume> contrôle du volume en pourcentage pour le microphone du dispositif. 0 est muet. <speakerVolume> contrôle du volume en pourcentage pour le haut-parleur du dispositif. 0 est muet.			

A.7.4.2.1 Bloc XML AudioChannel

```
<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
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
Notes	Service vidéo.				
	Les sorties vidéo (c'est-à-dire le décodage) seront couvertes dans une spécification IPMD future.				

A.7.5.1 /System/Video/overlayImages

URI	/System/Video/overlayImages			Type	Ressource
Fonction	Gérer les images en surimpression.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<OverlayImageList>		
POST		Données d'image brutes	<ResponseStatus>		
DELETE			<ResponseStatus>		
Notes	Il s'agit de topogrammes binaires utilisés par les images en surimpression pour les voies vidéo. Le référentiel d'images est centralisé, de manière à pouvoir utiliser la même image pour plusieurs voies. <imageType> est le type MIME de l'image, c'est-à-dire "image/jpeg". <imageWidth> et <ImageHeight> sont la largeur et la hauteur de l'image. En cas d'émission d'un POST des données d'image, il faut que le client attribue le type MIME correct de l'image au champ d'en-tête HTTP Content-Type:..				

A.7.5.1.1 Bloc XML ImageOverlayList

<pre> <OverlayImageList version="1.0" xmlns="urn:psalliance-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> </pre>					
--	--	--	--	--	--

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

URI	/System/Video/overlayImages/ID			Type	Ressource
Fonction	Accéder à l'image en surimpression pour une voie particulière.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			Données d'image brutes		
PUT		Données d'image brutes	<ResponseStatus>		
DELETE			<ResponseStatus>		
Notes	Les images en surimpression peuvent être mises à jour en utilisant la commande PUT et supprimées en utilisant la commande DELETE.				

A.7.5.3 /System/Video/inputs

URI	/System/Video/inputs			Type	Ressource
Fonction	Accéder aux entrées vidéo sur un dispositif média IP.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<VideoInput>		

Notes	Un dispositif média IP peut contenir un ensemble d'entrées vidéo. Ces entrées sont câblées par le dispositif, ce qui signifie que les ID peuvent être découverts mais ni créés ni supprimés. Il convient de considérer la numérotation ou les valeurs d'ID de manière arbitraire et en fonction du dispositif.
--------------	--

A.7.5.3.1 Bloc XML VideoInput

```
< 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	Ressource
Fonction					
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET		Néant	<VideoInputChannelList>		
Notes	Les voies d'entrée vidéo étant des ressources définies par la configuration matérielle du dispositif, elles ne peuvent être ni créées ni supprimées.				

A.7.5.4.1 Bloc XML VideoInputChannelList

```
< 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	Ressource
Fonction	Régler les propriétés des voies d'entrée.				
Méthodes	Chaîne(s) de requête	Données entrantes	Résultat de retour		
GET			<VideoInputChannel>		
PUT		<VideoInputChannel>	<ResponseStatus>		