AS/NZS IEC 62676.2.2:2020
IEC 62676-2-2:2013

AS/NZS IEC 62676.2.2:2020

Australian/New Zealand Standard™

Video surveillance systems for use in security applications

Part 2.2: Video transmission protocols — IP interoperability implementation based on HTTP and REST services

Documents
Document Sharing Hub

Full document details: AS NZS IEC 62676.2.2-2020

AS/NZS IEC 62676.2.2:2020

This Joint Australian/New Zealand Standard™ was prepared by Joint Technical Committee EL-031, Intruder Alarm Equipment and Installations. It was approved on behalf of the Council of Standards Australia on 18 March 2020 and by the New Zealand Standards Approval Board on 5 February 2020.

This Standard was published on 24 April 2020.

The following are represented on Committee EL-031:

Australian Digital & Telecommunications Industry Association
Australian Federal Police
Australian Industry Group
Australian Security Industry Association
Engineers Australia
Fire Protection Association Australia
Insurance Council of Australia
New Zealand Security Association
NSW Police Force
Queensland Police Service
Security Providers Association of Australia
South Australia Police
Victoria Police
Victorian Security Institute

This Standard was issued in draft form for comment as DR AS/NZS IEC 62676.2.2:2019.

Keeping Standards up-to-date

Ensure you have the latest versions of our publications and keep up-to-date about Amendments, Rulings, Withdrawals, and new projects by visiting:

www.standards.org.au

www.standards.govt.nz

ISBN 978 1 76072 825 0

Documents
Document Sharing Hub

Full document details: AS NZS IEC 62676.2.2-2020

AS/NZS IEC 62676.2.2:2020
IEC 62676-2-2:2013

Australian/New Zealand Standard™

Video surveillance systems for use in security applications

Part 2.2: Video transmission protocols — IP interoperability implementation based on HTTP and REST services

First published as AS/NZS IEC 62676.2.2:2020.

COPYRIGHT

© IEC 2020 — All rights reserved
© Standards Australia Limited/the Crown in right of New Zealand, administered by the New Zealand Standards Executive 2020

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968 (Cth) or the Copyright Act 1994 (New Zealand).

Documents
Document Sharing Hub

Full document details: AS NZS IEC 62676.2.2-2020

Preface

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee EL-031, Intruder Alarm Equipment and Installations.

The objective of this Standard is to specify 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 conformance to the requirements of this standard based on HTTP and REST Services as described in this document is declared as compatible to ´AS/NZS IEC 62676-2 HTTP and REST interoperability.'

This Standard is identical with, and has been reproduced from, IEC 62676-2-2:2013, *Video surveillance systems for use in security applications — Part 2-2: Video transmission protocols — IP interoperability implementation based on HTTP and REST services*.

As this document has been reproduced from an International Standard, the following applies:

(a) In the source text "this part of IEC 62676" should read "this Australian/New Zealand Standard".

(b) A full point substitutes for a comma when referring to a decimal marker.

Australian or Australian/New Zealand Standards that are identical adoptions of international normative references may be used interchangeably. Refer to the online catalogue for information on specific Standards.

The terms "normative" and "informative" are used in Standards to define the application of the appendices or annexes to which they apply. A "normative" appendix or annex is an integral part of a Standard, whereas an "informative" appendix or annex is only for information and guidance.

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

NOTES

Documents
Document Sharing Hub

Full document details: AS NZS IEC 62676.2.2-2020

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

AVA Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

AVA Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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*

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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

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

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

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

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

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

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

IETF RFC 2818, *HTTP Over TLS*

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

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

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

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

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

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

IETF RFC 4566, *SDP: Session Description Protocol*

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

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

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

3 Abbreviations

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

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

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.).

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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”

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

Table 2 – HTTP status codes and REST

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

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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

5.6 Unique identifiers

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

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

5.7 ID encoding

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

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

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

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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.

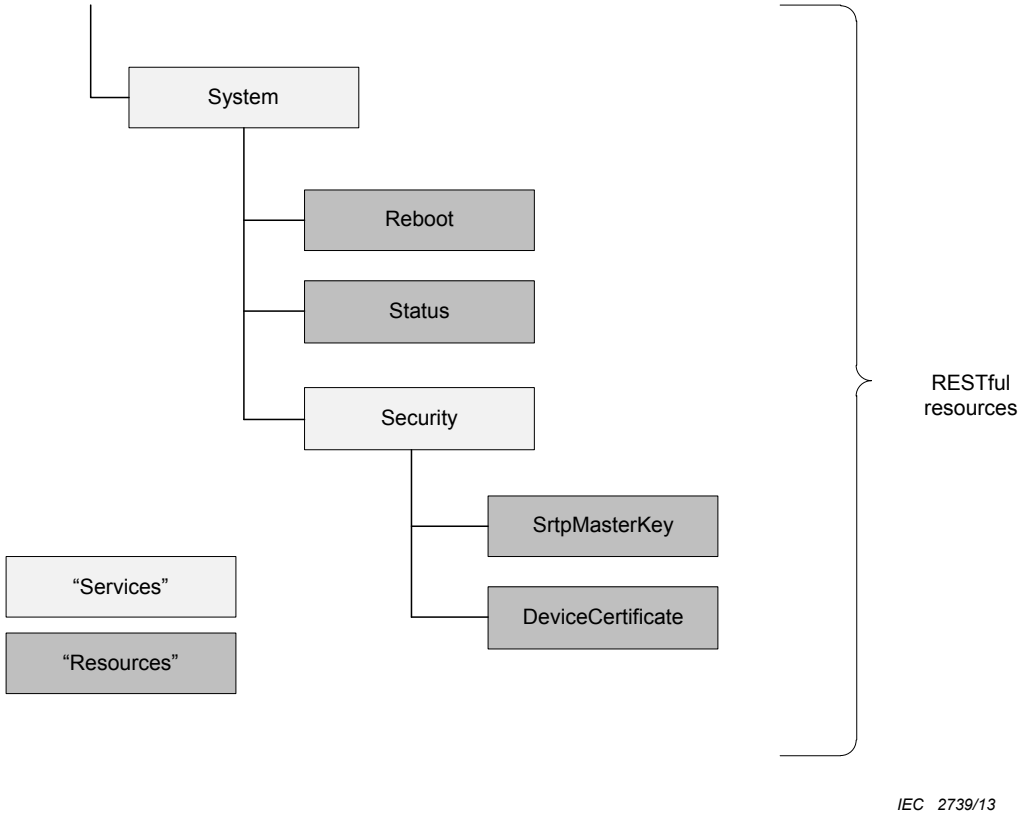

Figure 1 – PSIA service architecture example

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

Multiple channels and versions

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

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

Note that Channel IDs are arbitrarily assigned by the device.

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

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

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

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

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

7 System flow

7.1 General

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

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

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

HTTP requests are made through the device’s web server. The HTTP response may contain XML content (for GET actions), XML response information (for PUT or POST actions), or various text/binary content (for retrieval of configuration data, etc.). Edge devices should be

AVA Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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,

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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”.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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>
```

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

7.13.3 Status string

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

7.13.4 ID

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

7.14 Processing rules

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

8 XML modeling

8.1 File format

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

8.2 Data structures

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

8.3 Lists

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

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

8.4 Capabilities

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

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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

Table 6 – Capability attributes

Capability attribute	Description	Syntax	Applicable XML data types
Min	The minimum character length for a string, or the minimum numerical value of a number	Examples: min="0" min="64" min="-100" (numerical only) min="1,2"	All except fixed data types [a]
max	The maximum character length for a string, or the maximum numerical value of a number	Examples: max="5" max="64" max="4 096" max="10,50"	All except fixed data types [a]
range	Indicates the possible range of numerical values within the "min" and "max" attributes of an element. This attribute should only be used if the possible values for an XML element does not include the entire numerical range between "min" and "max" attributes	Ranges are listed in numerical order separated by a "," character. A range has the form "x~y" where x is the range floor and y is the range ceiling. Single numbers may also be used. *Example*: if an XML element supports values 0, 123, 1024 to 2000, and 2003, the syntax would be: range="0,123,1024~2000,2003"	All numerical data types
opt	Lists the supported options for a CodeID data type. Required for XML elements with a CodeID data type. This attribute should *not* be used for any other data type	If all options are supported, the syntax is "all". Otherwise, supported options are listed separated by a "," character. Examples: opt="all" opt="1,2,3" opt="1,2,5,8,9,10,11"	CodeID
Def	Indicates the default value of the XML element. If the element has no default value, this attribute should *not* be used	Examples: def="1234" def="Device ABC" def="3"	All data types
reqReboot	Indicates if configuration of this XML element requires a device reboot before taking effect. If an element doesn't require a reboot, this attribute should *not* be used	reqReboot="true"	All data types
dynamic	Indicates if an XML element has dynamic	dynamic="true"	All data types

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

Capability attribute	Description	Syntax	Applicable XML data types
	capabilities dependent on other XML configurations. For example, if an element's data range changes based on another element's configured value, this attribute shall be used. In this case, the element's capability attributes shall always reflect the current device configuration		
Size	Indicates the maximum number of entries in an XML list. This attribute is only applicable to XML list elements. This attribute should not be used for any other type of element (see 8.3 for details)	*Example*: If a device supports 5 users the example would be <UserSetting> <UserList size="5"> …	Only supported for list elements (see section 8.3)
a Fixed, pre-defined data types do not need certain capability attributes because their formats/data ranges are already defined. Where pre-defined data types are used, each protocol document shall include an enumeration of these formats in an appendix.			

9 Custom services and resources

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

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

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

10 Interface design

10.1 General

The HTTP URL format is of the general form

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

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

10.2 Protocol

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

- http
- https

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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:

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

- **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				

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

11.6.2 ResourceDescription

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

11.6.3 ResourceList

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

11.6.4 QueryStringParameterList

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

11.6.5 responseStatus

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

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

```
  </ResponseStatus>
```

11.6.6 service.xsd

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

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

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

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

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

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

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

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

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

```
  <!-- 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>
```

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

```
  <!-- 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>
```

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020

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.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2020

Full document details: AS NZS IEC 62676.2.2-2020

A.4 Conformance

A.4.1 General

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

The required services defined below are the fundamental services for all IP video HTTP and REST specifications and are intended to be referenced by other specifications.

The optional services defined are specific to IP media devices.

A.4.2 Service requirements

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

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

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2020
Full document details: AS NZS IEC 62676.2.2-2020