IEC 60839-11-31

Edition 1.0 2016-11

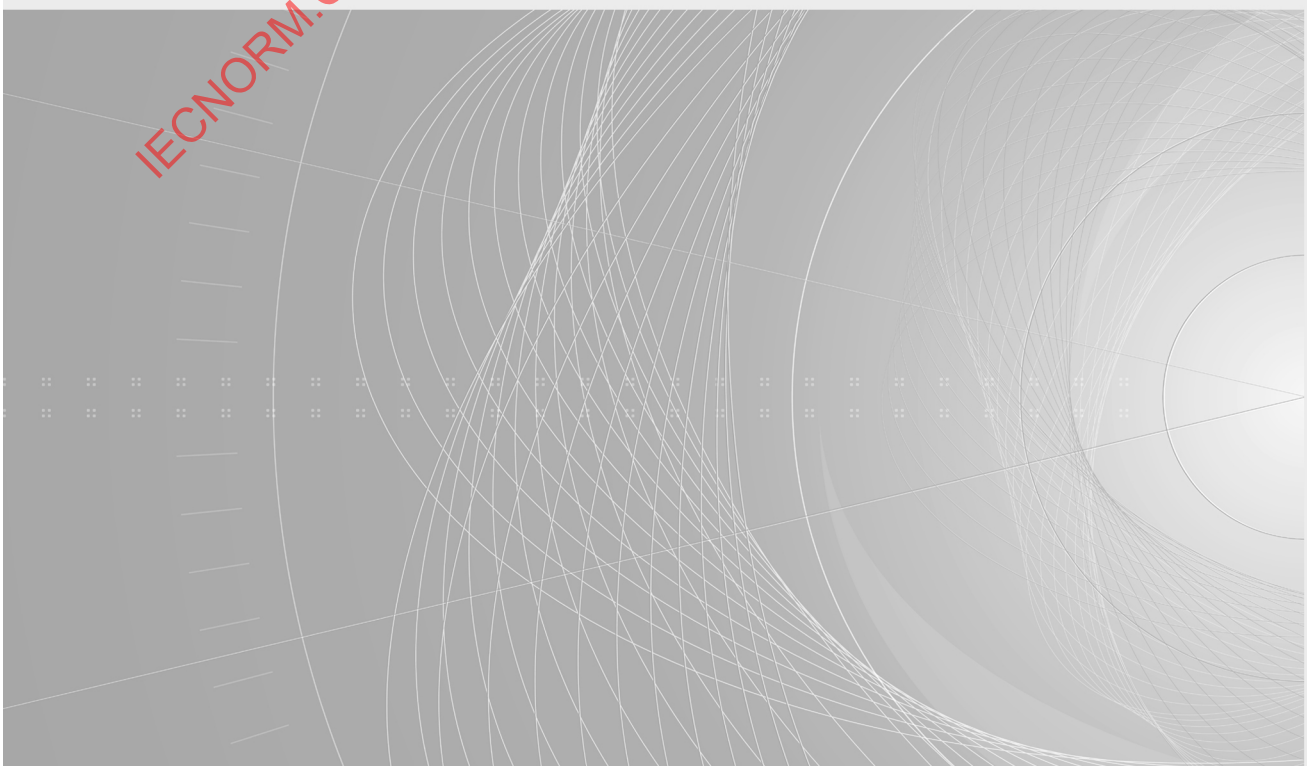
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Alarm and electronic security systems – Part 11-31: Electronic access control systems – Core interoperability protocol based on Web services

Systèmes d'alarme et de sécurité électroniques – Partie 11-31: Systèmes de contrôle d'accès électronique – Protocole de base d'interopérabilité en fonction des services Web



IEC 60839-11-31:2016-11(en-fr)

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

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

IEC 60839-11-31

Edition 1.0 2016-11

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Alarm and electronic security systems – Part 11-31: Electronic access control systems – Core interoperability protocol based on Web services

Systèmes d'alarme et de sécurité électroniques – Partie 11-31: Systèmes de contrôle d'accès électronique – Protocole de base d'interopérabilité en fonction des services Web

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 13.320

ISBN 978-2-8322-3778-6

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

ALARM AND ELECTRONIC SECURITY SYSTEMS –

Part 11-31: Electronic access control systems – Core interoperability protocol based on Web 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 60839-11-31 has been prepared by IEC technical committee 79: Alarm and electronic security systems.

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

CDV	Report on voting
79/522/CDV	79/546/RVC

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 60839 series, published under the general title *Alarm and electronic security systems*, 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 website 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.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The object of this document is to provide the common base for a fully interoperable network implementation comprised of products from different network vendors. This document describes the network model, interfaces, data types and data exchange patterns. This document reuses existing relevant standards where available, and introduces new specifications only where necessary.

This document is based upon work done by the ONVIF open industry forum. The ONVIF Core specification is compatible with this document.

This document is accompanied by a set of computer readable interface definitions:

- Device Service WSDL, see Clause B.1;
- Device IO Service WSDL, see Clause B.2;
- Event Service WSDL, see Clause B.3;
- Common schema, see Clause B.4.

This document is divided into the following clauses:

Document overview: Gives an overview of the different standard parts and how they are related to each other.

Web services frame work: Offers a brief introduction to Web services and the Web services basis for this document.

IP configuration: Defines the network IP configuration requirements.

Device discovery: Describes how devices are discovered in local and remote networks.

Device management: Defines the configuration of basics like network and security related settings.

Device IO: Defines the handling of input and output ports on a device.

Event handling: Defines how to subscribe to and receive notifications (events) from a device.

Security: Defines the transport and message level security requirements.

ALARM AND ELECTRONIC SECURITY SYSTEMS –

Part 11-31: Electronic access control systems – Core interoperability protocol based on Web services

1 Scope

This part of IEC 60839 defines procedures for communication between network clients and devices. This series of interoperability standards makes it possible to build an alarm and electronic security system with clients and devices from different manufacturers using common and well defined interfaces. The functions defined in this document covers discovery, device management and event framework. Supplementary dedicated services are defined in separate documents.

The management and control interfaces defined in this document are described as Web services. This document also contains full XML schema and Web Service Description Language (WSDL) definitions.

In order to offer full plug-and-play interoperability, this document defines procedures for device discovery. The device discovery mechanisms in this document are based on the WS-Discovery specification with extensions.

This document does not in any way limit a manufacturer to add other protocol or extend the protocol defined here and rules on how to accomplish this are also provided in this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEEE 1003.1, *The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition <http://pubs.opengroup.org/onlinepubs/009695399/>*

IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*
<http://standards.ieee.org/getieee802/download/802.11-2007.pdf>

IEEE 802.1X, *Port-Based Network Access Control*
<http://standards.ieee.org/getieee802/download/802.1X-2004.pdf>

IETF RFC 952, Internet Host Table Specification
<https://tools.ietf.org/html/rfc952

IETF RFC 1123:1989, Requirements for Internet Hosts – Application and Support
<https://tools.ietf.org/html/rfc1123>

IETF RFC 2131, *Dynamic Host Configuration Protocol*
<http://www.ietf.org/rfc/rfc2131.txt>

IETF RFC 2136, *Dynamic Updates in the Domain Name System (DNS UPDATE)*
<http://www.ietf.org/rfc/rfc2136.txt>

IETF RFC 2246, *The TLS Protocol Version 1.0*
<http://www.ietf.org/rfc/rfc2246.txt>

IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*
<http://www.ietf.org/rfc/rfc2617.txt>

IETF RFC 2986, PKCS #10, *Certification Request Syntax Specification Version 1.7*
<http://www.ietf.org/rfc/rfc2986.txt>

IETF RFC 3268, *Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS)*
<http://www.ietf.org/rfc/rfc3268.txt>

IETF RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
<http://www.ietf.org/rfc/rfc3315.txt>

IETF RFC 3927, *Dynamic Configuration of IPv4 Link-Local Addresses*
<http://www.ietf.org/rfc/rfc3927.txt>

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*
<http://www.ietf.org/rfc/rfc4122.txt>

IETF RFC 4514, *Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names*
<http://www.ietf.org/rfc/rfc4514.txt>

IETF RFC 4702, *The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option*
<http://www.ietf.org/rfc/rfc4702.txt>

IETF RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*
<http://www.ietf.org/rfc/rfc4861.txt>

IETF RFC 4862, *IPv6 Stateless Address Auto configuration*
<http://www.ietf.org/rfc/rfc4862.txt>

ISO/IEC 8824-2, *Information Technology – Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-3, *Information Technology – Abstract Syntax Notation One (ASN.1): Constraint specification*

ISO/IEC 8824-4, *Information Technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

ISO/IEC 8825-1, *Information Technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

OASIS WS-BaseNotification, *Web Services Base Notification 1.3 (WS-BaseNotification)*
<http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf>

OASIS WS-Topics, *Web Services Topics 1.3 (WS-Topics)*
<http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf>

W3C SOAP-MTOM, *SOAP Message Transmission Optimization Mechanism*
<http://www.w3.org/TR/soap12-mtom/>

W3C SOAP12-PART1, *SOAP 1.2 Part 1, Messaging Framework*
<http://www.w3.org/TR/soap12-part1/>

W3C WS-Addressing, *Web Services Addressing 1.0 – Core*
<http://www.w3.org/TR/ws-addr-core/>

WS-I BP 2.0, *Basic Profile Version 2.0*
<http://www.ws-i.org/Profiles/BasicProfile-2.0-2010-11-09.html>

XMLSOAP WS-Discovery, *Web Services Dynamic Discovery (WS-Discovery)", J. Beatty et al., April 2005*
<http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf>

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
ad-hoc network
independent basic service set

Note 1 to entry: "Ad-hoc network" is a vernacular term.

3.1.2
basic service set
set of IEEE 802.11 stations that have successfully joined in a common network

3.1.3
capability
command which allows a client to ask for the services provided by a device

3.1.4
public key cryptography standards
PKCS
group of standards devised and published by RSA Security

Note 1 to entry: This note only applies to the French language.

3.1.5
pre shared key
static key that is distributed to the device

3.1.6
pull point
resource for pulling messages

Note 1 to entry: By pulling messages, notifications are not blocked by firewalls.

3.1.7
service set ID
identity of an IEEE 802.11 wireless network

[SOURCE: IEEE 802.11-2007]

3.1.8
Wi-Fi protected access
WPA
certification program created by the Wi-Fi Alliance to indicate compliance with the security protocol covered by the program

Note 1 to entry: This note only applies to the French language.

3.2 Abbreviated terms

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN	Abstract Syntax Notation
BSSID	Basic Service Set Identification
CA	Certificate Authority
CBC	Cipher-Block Chaining
CCMP	Counter mode with Cipher-block chaining Message authentication code Protocol
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DM	Device Management
DNS	Domain Name Server
DP	Discovery Proxy
EAP	Extensible Authentication Protocol
GW	Gateway
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IANA	Internet Assigned Numbers Authority
IO, I/O	Input/Output
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
LAN	Local Area Network
MTOM	Message Transmission Optimization Mechanism
NAT	Network Address Translation
NFC	Near Field Communication
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
POSIX	Portable Operating System Interface
PKCS	Public Key Cryptography Standards
PSK	Pre Shared Key
REL	Rights Expression Language

RSA	Rivest, Sharmir and Adleman
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SSID	Service Set ID
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TKIP	Temporal Key Integrity Protocol
TTL	Time To Live
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URN	Uniform Resource Name
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
WDR	Wide Dynamic Range
WS	Web services
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
XML	eXtensible Markup Language
XPath	XML Path Language

4 Overview

4.1 General

This document defines a core set of interface functions for configuration and operation of network devices by defining their server side interfaces. This document covers device discovery, device configuration as well as an event framework.

All services share a common XML schema, see Clause B.4. The different services are defined in the respective sections and service WSDL documents.

4.2 Web services

The term Web services is the name of a standardized method of integrating applications using open, platform independent Web services standards such as XML, SOAP 1.2 [W3C SOAP12-PART1] and WSDL1.1 [W3C WSDL11] over an IP network. XML is used as the data description syntax, SOAP is used for message transfer and WSDL is used for describing the services.

This framework is built upon Web services standards. All configuration services defined in this document are expressed as Web services operations and defined in WSDL with HTTP [RFC 2616] as the underlying transport mechanism.

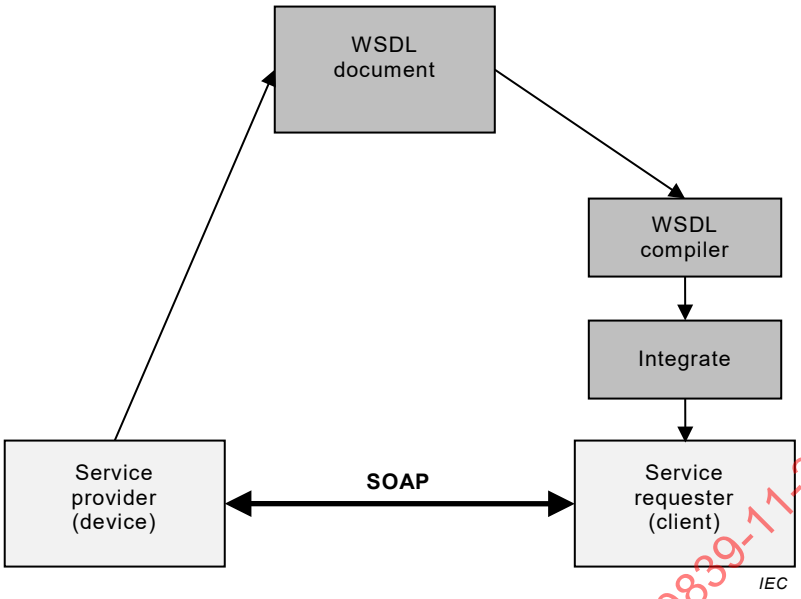


Figure 1 – Web services based development principles

Figure 1 gives an overview of the basic principles for development based on Web services. The service provider (device) implements the service or services. The service is described using the XML-based WSDL. Then, the WSDL is used as the basis for the service requester (client) implementation/integration. Client-side integration is simplified through the use of WSDL compiler tools that generate a platform specific code that can be used by the client side developer to integrate the Web Service into an application.

The Web services provider and requester communicate using the SOAP message exchange protocol. SOAP is a lightweight, XML-based messaging protocol used to encode the information in a Web Service request and in a response message before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols. This document defines conformant transport protocols for the SOAP messages for the described Web services.

The Web services subclause introduces the general service structure, the command definition syntax used in this document, error handling principles and the adopted Web Service security mechanisms.

To ensure interoperability, all services follow the Web services Interoperability Organization (WS-I) basic profile 2.0 recommendations and use the document/literal wrapped pattern.

4.3 IP configuration

The IP configuration subclause defines the IP configuration compliance requirements and recommendations. IP configuration includes:

- IP network communication capability,
- static IP configuration,
- dynamic IP configuration.

4.4 Device discovery

The configuration interfaces defined in this document are Web services interfaces that are based on the WS-Discovery standard. This use of this document makes it possible to reuse a suitable existing Web Service discovery framework, instead of requiring a completely new service or service addressing definition.

This document introduces a specific discovery behaviour suitable for alarm and electronic security systems. For example, a fully interoperable discovery requires a well defined service definition and a service searching criteria. This document covers device type and scopes definitions in order to achieve this.

A successful discovery provides the device service address. Once a client has the device service address it can receive detailed device information through the device service, see 4.5 below.

4.5 Device management

4.5.1 General

Device management functions are handled through the device service. The device service is the entry point to all other services provided by a device. WSDL for the device service is provided in the device management WSDL file. The device management interfaces consist of these subcategories:

- capabilities,
- network,
- system,
- security.

4.5.2 Capabilities

The capability commands allow a client to ask for the services provided by a device and to determine which general and vendor specific services are offered by the device. The capabilities are structured per service. This document defines the capability exchange for the device, device IO and the event service. For the other services refer to the respective service specification:

- device,
  - network,
  - system,
  - security,
- device IO,
- event.

The capabilities for the different categories indicate those commands and parameter settings that are available for the particular service or service subcategory.

4.5.3 Network

The following set of network commands allows standardized management of functions:

- Get and set hostname.
- Get and set DNS configurations.
- Get and set NTP configurations.
- Get and set dynamic DNS.
- Get and set network interface configurations.
- Enable/disable and list network protocols.
- Get and set default gateway.
- Get and set zero configuration.
- Get, set, add and delete IP address filter.

- Wireless network interface configuration.

4.5.4 System

The system commands are used to manage the following device system settings:

- Get device information.
- Make system backups.
- Get and set system date and time.
- Factory default reset.
- Upgrade firmware.
- Get system log.
- Get device diagnostics data (support information).
- Reboot.
- Get and set device discovery parameters.

4.5.5 Retrieval of system information

System information, such as system logs, vendor-specific support information and configuration backup images, may be retrieved using either MTOM or HTTP.

The MTOM method is supported by the GetSystemLog, GetSystemSupportInformation and GetSystemBackup commands. The HTTP method is supported by the GetSystemUris command; this retrieves URIs from which the files may be downloaded using an HTTP GET operation.

4.5.6 Firmware upgrade

Two mechanisms are provided for upgrading the firmware on a device. The first uses the UpgradeSystemFirmware command to send the new firmware image using MTOM.

The second is a two stage process; first the client sends the StartFirmwareUpgrade command to instruct the device to prepare for upgrade, then it sends the firmware image using HTTP POST.

The HTTP method is designed for resource-limited devices that may not be capable of receiving a new firmware image in its normal operating state.

4.5.7 SystemRestore

The SystemRestore capability allows a device's configuration to be restored from a backup image. Again two mechanisms are provided. The first uses the RestoreSystem command to send the backup image using MTOM. The second uses the StartSystemRestore command followed by an HTTP POST operation to send the backup image.

4.5.8 Security

The following security operations are used to manage the device security configurations:

- Get and set access security policy.
- Handle user credentials and settings.
- Handle HTTPS server certificates.
- Enable/disable HTTPS client authentication.
- Key generation and certificate download functions.

- Handle IEEE 802.1X supplicant certificate.
- Handle IEEE 802.1X CA certificate.
- IEEE 802.1X configuration.

4.6 DeviceIO

The DeviceIO service offers commands to retrieve and configure the settings of physical inputs and outputs of a device.

The DeviceIO service supports the configuration of the following device interfaces:

- RelayOutputs,
- DigitalInputs,
- Send and/or receive serial data communication.

4.7 Event handling

Event handling is based on the OASIS WS-BaseNotification and WS-Topics specifications. These specifications allow the reuse of a rich notification framework without the need to redefine event handling principles, basic formats and communication patterns.

Firewall traversal, according to WS-BaseNotification, is handled through a PullPoint notification pattern. This pattern, however, does not allow real-time notification. Hence, this standard defines an alternative PullPoint communication pattern and service interface. The PullPoint pattern allows a client residing behind a firewall to receive real-time notifications while utilizing the WS-BaseNotification framework.

A fully standardized event handling requires standardized notifications. However, the notification topics will, to a large extent, depend on the application needs. This document defines a set of basic notification topics.

WSDL for the event service including extensions is provided in the Event WSDL file.

4.8 Security

Subclause 4.8 describes network security requirements. This document defines security mechanism on two different communication levels:

- transport-level security,
- message-level security.

This document also defines port-based network security as follows.

- IEEE 802.1X

The general security requirements, definitions and transport security requirements are specified in 11.3. Message level security requirements are specified in 5.13. IEEE 802.1X requirements are specified in 8.4.8. Security management is handled through the device management service as listed above in 4.5.8.

5 Web services framework

5.1 General

All management and configuration commands are based on Web services.

For the purposes of this document:

- the device is a service provider;
- the client is a service requester.

A typical network system does have multiple clients that handle device configuration and device management operations for numerous devices. Additionally a device providing services may also act as a client.

Web services also require a common way to discover service providers. This discovery is achieved using the Universal Discovery, Description and Integration Registry (UDDI) specifications [UDDI API ver2], [UDDI Data Structure ver2]. The UDDI specifications utilize service brokers for service discovery. This document targets devices while the UDDI model is not device oriented. Consequently, UDDI and service brokers are outside the scope of this document.

According to this document, devices (service providers) are discovered using WS-Discovery [XMLSOAP WS-Discovery] based techniques. The service discovery principles are described in Clause 7.

Web services allow developers the freedom to define services and message exchanges, which may cause interoperability problems. The Web services interoperability organization (WS-I) develops standard profiles and guidelines to create interoperable Web services. The devices and the clients shall follow the guidelines in the WS-I Basic Profile 2.0 [WS-I BP 2.0]. The service descriptions in this document follow the WS-I Basic Profile 2.0 recommendations.

5.2 Services overview

5.2.1 General

A device shall support a number of Web services which are defined in this document and related specifications.

The device management service is the entry point for all other services of the device and therefore also the target service for the WS-Discovery behavior defined by this document, see Clause 7.

The entry point for the device management service is fixed to:

```
http://onvif_host/onvif/device_service
```

5.2.2 Services requirements

A device shall provide the device management and event service.

If a device supports a certain service, the device shall respond to all commands defined in the corresponding service WSDL. If the specific command is not required for that service and the device does not support the command, the device should respond to a request with the error codes:

env:Receiver,

ter:ActionNotSupported,

See 5.12.3 for the definitions of the error codes.

5.3 WSDL overview

“WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined

into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate" [SOURCE: W3C WSDL11].

This document follows the WSDL 1.1 specification, see [W3C WSDL11], and uses the document/literal wrapped pattern.

A WSDL document consists of the following sections:

- types – Definition of data types using XML schema definitions.
- message – Definition of the content of input and output messages.
- operation – Definition of how input and output messages are associated with a logical operation.
- portType – Groups a set of operations together.
- binding – Specification of which protocols that are used for message exchange for a particular portType.
- port – Specifies an address for a binding.
- service – Used to group a set of related ports.

5.4 Namespaces

Note that the Namespace URI, listed in Tables 1, 2 and 3, is primarily used to create a unique idenfier. It does not have to be useful for retrieving a corresponding document, for example an XML schema or WSDL file.

Prefix and namespaces used in this document are listed in Table 1. These prefixes are not part of this document and an implementation can use any prefix.

Table 1 – Defined namespaces in this document

Prefix	Namespace URI	Description
tt	http://www.onvif.org/ver10/schema	XML schema descriptions in this document.
tds	http://www.onvif.org/ver10/device/wsdl	The namespace for the WSDL device service.
tmd	http://www.onvif.org/ver10/deviceIO/wsdl	The namespace for the WSDL device IO service.
trt	http://www.onvif.org/ver10/media/wsdl	The namespace for the WSDL media service.
tev	http://www.onvif.org/ver10/events/wsdl	The namespace for the WSDL event service.
ter	http://www.onvif.org/ver10/error	The namespace used for faults defined by this document.
tns1	http://www.onvif.org/ver10/topics	The namespace used for topics defined by this document.

The namespaces listed in Table 2 are referenced by this document.

Table 2 – Referenced namespaces (with prefix)

Prefix	Namespace URI	Description
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace for WSDL framework.
wsoap12	http://schemas.xmlsoap.org/wsdl/soap12/	WSDL namespace for WSDL SOAP 1.2 binding.
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace for WSDL HTTP GET & POST binding.
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [W3C SOAP12-PART11]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [W3C XML-Schema-1] and [W3C XML-Schema- 2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML schema instance namespace.
d	http://schemas.xmlsoap.org/ws/2005/04/discovery	Device discovery namespace as defined by [XMLSOAP WS-Discovery].
wsadis	http://schemas.xmlsoap.org/ws/2004/08/addressing	Device addressing namespace referred in WS-Discovery [XMLSOAP WS-Discovery].
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [W3C WS-Addressing].
wstop	http://docs.oasis-open.org/wsn/t-1	Schema namespace of the [OASIS WS-Topics] specification.
wsnt	http://docs.oasis-open.org/wsn/b-2	Schema namespace of the [OASIS WS-BaseNotification] specification.
xop	http://www.w3.org/2004/08/xop/include	XML-binary Optimized Packaging namespace as defined by [W3C XOP]

In addition, this document refers without prefix to the namespaces listed in Table 3.

Table 3 – Referenced namespaces (without prefix)

Namespace URI	Description
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete	Topic expression dialect defined for topic expressions.
http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet	Topic expressions dialect defined by this document.
http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter	Filter dialect, used for message content filtering, defined by this document.

5.5 Types

Data types are defined using XML schema descriptions Part 1 and Part 2. All data types defined in this document are included in the common schema, see Clause B.4.

5.6 Messages

According to WSDL 1.1 operations are described using input and output messages in XML. The message section contains the message content.

A message in this document contains two main elements:

- message name,
- message parts.

The message name specifies the name of the element and that name is used in the operation definition in the WSDL document. The message name defines the name of the message.

The WSDL message part element is used to define the actual format of the message. Although there can be multiple parts in a WSDL message, this document follows the WS-I basic profile [WS-I BP 2.0] and does not allow more than one part element in a message. Hence the same name (“parameters”) is always used for the message part name.

The following WSDL notation is used for this document:

```
<message name="'Operation_Name'Request">
      <part name="parameters" element="'prefix':'Operation_Name'"/>
</message>
```

respectively,

```
<message name="'Operation_Name'Response">
      <part name="parameters" element="'prefix':'Operation_Name'Response"/>
</message>
```

where 'prefix' is the prefix for the namespace in which the message is defined.

This document uses message specific types that encapsulate multiple parts to allow multiple arguments (or data) in messages.

5.7 Operations

5.7.1 General

Operations are defined within the WSDL portType declaration. An operation can be one of these two types:

- One-way – The service provider receives a message.
- Request-response – The service provider receives a message and sends a corresponding message.

Depending on the operation, different port types can be used.

The operation name defines the name of the operation.

Operations in this document are defined using the following table format outlined in Table 4.

Table 4 – Operation description outline used in this document

Operation_Name	Access_Class_Name
Message name	**Description**
'Operation_Name'Request	*Description of the request message.* *Type*$_{r1}$ **Name**$_{r1}$ [a_{r1}][b_{r1}] *Type*$_{r2}$ **Name**$_{r2}$ [a_{r2}][b_{r2}] : *Type*$_{rn}$ **Name**$_{rn}$ [a_{rn}][b_{rn}]
'Operation_Name'Response	*Description of the response message.* *Type*$_{s1}$ **Name**$_{s1}$ [a_{s1}][b_{s2}] *Type*$_{s2}$ **Name**$_{s2}$ [a_{s2}][b_{s2}] : *Type*$_{sn}$ **Name**$_{sn}$ [a_{sn}][b_{sn}]
'FaultMessage_Name'	*In the case that operation specific faults are defined, this field describes the structure of the defined fault message.*
Fault codes	**Description**
Code *Subcode* *Subcode*	*Description of the operation specific fault.*

The description column includes a list of the elements (if applicable) included in the request and response messages respectively. The value between brackets defines the lower and upper limits of the number of occurrences that can be expected for the element of the specified type. For example, $Name_{s2}$ in the table above occurs at least a_{s2} times and at most b_{s2} times.

Most commands do not define any specific fault messages. If a message is defined, it follows in the table directly after the response message.

The fault codes listed in the tables are the specific fault codes that can be expected from the command, see 5.12.3.3. Any command can return a generic fault, see 5.12.3.3.

The Access_Class_Name defines the access class of the operation. The access class characterizes the impact of the operation, see 5.13.2.3.

5.7.2 One-way operation type

A one-way operation type is used when the service provider receives a control message and does not send any explicit acknowledgement message or confirmation. This document makes use of one-way operations for discovery and event purposes only.

This operation type is defined by a single input message.

Use the following table format to describe one-way operations:

Operation_Name	One-way
Message name	**Description**
'Operation_Name'Request	*Description of the request message.* *Type*$_1$ **Name**$_1$ [a_1][b_1] *Type*$_2$ **Name**$_2$ [a_2][b_2] : *Type*$_n$ **Name**$_n$ [a_n][b_n]

This table corresponds to the following WSDL notation in the service specifications:

```
<operation name="'Operation_Name'">
     <input message="'prefix':'Operation_Name'"/>
</operation>
```

5.7.3 Request-response operation type

A request-response operation type is used when a service provider receives a message and responds with a corresponding message.

This operation type is defined by one input, one output and multiple fault messages.

Use the following table format to describe request-response operations:

Operation_Name	Request-Response
Message name	**Description**
'Operation_Name'Request	*Description of the request message.* *Type*$_{r1}$ **Name**$_{r1}$ [a_{r1}][b_{r1}] *Type*$_{r2}$ **Name**$_{r2}$ [a_{r2}][b_{r2}] : *Type*$_{rn}$ **Name**$_{rn}$ [a_{rn}][b_{rn}]
'Operation_Name'Response	*Description of the response message.* *Type*$_{s1}$ **Name**$_{s1}$ [a_{s1}][b_{s2}] *Type*$_{s2}$ **Name**$_{s2}$ [a_{s2}][b_{s2}] : *Type*$_{sn}$ **Name**$_{sn}$ [a_{sn}][b_{sn}]
'"FaultMessage_Name'	*In the case that operation specific faults are defined, this field describes the structure of the defined fault message.*
Fault codes	**Description**
Code *Subcode* *Subcode*	*Description of the operation specific fault.*

This table corresponds to the following WSDL notation:

```
<operation name="'Operation_Name'">
     <input message="'prefix':'Operation_Name'"/>
       <output message="'prefix':'Operation_Name'Response"/>
       <fault name> = "Fault" message = "'prefix':'FaultMessage_Name'">
</operation>
```

5.8 Port types

A port type is a named set of abstract operations and the abstract messages involved. One single port type is a collection of several different operations.

All operation names in this document are sorted into categories. Each operation category contains one or more operations. Each category holds only one type of operation and is grouped into a single port type. A one-way operation and a request response operation can never exist for the same port type.

5.9 Binding

A binding defines a concrete protocol and transport data format specification for a particular port type. There may be any number of bindings for a given port type.

"Port_type" is a previously defined type and "Binding" is a character string starting with an upper case letter that defines the name of the binding.

Binding definitions for a compliant device shall follow the requirements in [WS-I BP 2.0]. This implies that the WSDL SOAP 1.2 bindings shall be used.

The SOAP binding can have different styles. A compliant device shall use the style 'document' specified at the operation level.

The bindings are defined in the WSDL specifications for respective services.

5.10 Ports

The individual endpoint is specified by a single address for a binding. Each port shall be given a unique name. A port definition contains a name and a binding attribute.

This document does not mandate any port naming principles.

5.11 Services

A service is a collection of related ports. This document does not mandate any service naming principles.

5.12 Error handling

5.12.1 General

As with any other protocol, errors can occur during communications, protocol or message processing.

This document classifies error handling into the following categories:

- protocol errors,
- SOAP errors,
- application errors.

5.12.2 Protocol errors

Protocol errors are the result of an incorrectly formed protocol message, which could contain illegal header values, or be received when not expected or experience a socket timeout. To indicate and interpret protocol errors, HTTP and RTSP protocols have defined a set of standard status codes (e.g. 1xx, 2xx, 3xx, 4xx, 5xx). According to this document, devices and clients shall use appropriate RTSP and HTTP protocol defined status codes for error reporting and when received handle them accordingly.

5.12.3 SOAP errors

5.12.3.1 General

SOAP errors are generated as a result of Web services operation errors or during SOAP message processing. All such SOAP errors shall be reported and handled through SOAP fault messages. The SOAP specification provides a well defined common framework to handle errors through SOAP fault.

A SOAP fault message is a normal SOAP message with a single well-known element inside the body (soapenv:Fault). To understand the error in more detail, SOAP has defined a SOAP fault message structure with various components in it.

- Fault code
- Subcode
- Reason
- Node and Role
- Fault Details

Subcode and **Fault Detail** elements information items are intended for carrying application specific error information.

This document uses a separate name space for specific faults, see 5.12.3.3:

ter = “http://www.onvif.org/ver10/error”.

SOAP fault messages for different Web services are defined as part of the different Web services definitions. Server and client shall use SOAP 1.2 fault message handling [W3C SOAP12-PART1] as specified in this document and shall follow the WS-I Basic Profile 2.0 fault handling recommendations.

The following example is an error message (SOAP 1.2 fault message over HTTP). The values in italics are placeholders for actual values.

```
HTTP/1.1 500 Internal Server Error
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: application/soap+xml; charset=”utf-8”
DATE: when response was generated
<?xml version=”1.0” ?>
<soapenv:Envelope        xmlns:soapenv="http://www.w3.org/2003/05/soap-
envelope"
   xmlns:ter="http://www.onvif.org/ver10/error"
   xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
<soapenv:Body>
 <soapenv:Fault>
   <soapenv:Code>
     <soapenv:Value>fault code </soapenv:Value>
     <soapenv:Subcode>
       <soapenv:Value>ter:fault subcode</soapenv:Value>
         <soapenv:Subcode>
           <soapenv:Value>ter:fault subcode</soapenv:Value>
        </soapenv:Subcode>
     </soapenv:Subcode>
    </soapenv:Code>
    <soapenv:Reason>
      <soapenv:Text xml:lang="en">fault reason</soapenv:Text>
    </soapenv:Reason>
    <soapenv:Node>http://www.w3.org/2003/05/soap-
envelope/node/ultimateReceiver</soapenv:Node>
    <soapenv:Role>http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver</soapenv:Role>
    <soapenv:Detail>
      <soapenv:Text>fault detail</soapenv:Text>
    </soapenv:Detail>
  </soapenv:Fault>
</soapenv:Body>
</soapenv:Envelope>
```

The following Table 5 summarizes the general SOAP fault codes (fault codes are defined in SOAP version 1.2 Part 1: Messaging Framework). Server and client may define additional fault subcodes for use by applications.

We distinguish between generic faults and specific faults. Any command can generate a generic fault. Specific faults are related to a specific command or set of commands. Specific faults that apply to a particular command are defined in the command definition tables.

In Table 5, the Fault Code, Subcode and Fault Reason are normative values. The description column is added for information.

5.12.3.2 Generic faults

Table 5 lists the generic fault codes and, if applicable, subcodes. All server and client implementations shall handle all the faults listed below. Any Web service command may return one or several of the generic faults.

The faults listed without subcode do not have any subcode value.

Table 5 – Generic faults

Fault code	Subcode	Fault reason	Description
env:VersionMismatch		SOAP version mismatch	The device found an invalid element information item instead of the expected Envelope element information item.
env:MustUnderstand		SOAP header blocks not understood	One or more mandatory SOAP header blocks were not understood.
env:DataEncodingUnknown		Unsupported SOAP data encoding	SOAP header block or SOAP body child element information item is scoped with data encoding that is not supported by the device.
env:Sender	ter:WellFormed	Well-formed error	XML well-formed violation occurred.
env:Sender	ter:TagMismatch	Tag mismatch	There was a tag name or namespace mismatch.
env:Sender	ter:Tag	No tag	XML element tag was missing.
env:Sender	ter:Namespace	Namespace error	SOAP namespace error occurred.
env:Sender	ter:MissingAttr	Required attribute not present	There was a missing required attribute.
env:Sender	ter:ProhibAttr	Prohibited attribute	A prohibited attribute was present.
env:Sender	ter:InvalidArgs	Invalid arguments	An error due to any of the following: – missing argument – too many arguments – arguments are of the wrong data type.
env:Sender	ter:InvalidArgVal	Argument value invalid	The argument value is invalid.
env:Sender	ter:UnknownAction	Unknown action	An unknown action is specified.
env:Sender	ter:OperationProhibited	Operation not permitted	The requested operation is not permitted by the device.
env:Sender	ter:NotAuthorized	Sender not authorized	The action requested requires authorization and the sender is not authorized.
env:Receiver	ter:ActionNotSupported	Optional action not implemented	The requested action is optional and is not implemented by the device.
env:Receiver	ter:Action	Action failed	The requested SOAP action failed.
env:Receiver	ter:OutofMemory	Out of memory	The device does not have sufficient memory to complete the action.
env:Receiver	ter:CriticalError	Critical error	The device has encountered an error condition which it cannot recover by itself and needs reset or power cycle.

5.12.3.3 Specific faults

Specific faults apply only to a specific command or set of commands. The specific faults are declared as part of the service definitions.

5.12.3.4 HTTP errors

If the server waits for the start of the inbound message and no SOAP message is received, the server shall not generate a SOAP fault and instead sends an HTTP error response (see Table 6.

Table 6 – HTTP errors

HTTP error	HTTP error code	HTTP reason
Malformed request	400	Bad request
Requires authorization	401	Unauthorized
HTTP method is neither POST or GET	405	Method not allowed
Unsupported message encapsulation method	415	Unsupported media

A server should avoid reporting internal errors as this can expose security weaknesses that can be misused.

5.13 Security

5.13.1 Authentication

A device shall support digest authentication according to [RFC 2617] for authentication of web service requests.

Digest authentication gives only a rudimentary level of security. In a system where security is important, it is recommended to always configure the device for TLS-based access (see 11.2). Digest authentication combined with TLS, with client and server authentication, protected transport level security give an acceptable level of security in many systems.

A device should authenticate an RTSP request at the RTSP level. If HTTP is used to tunnel the RTSP request the device shall not authenticate on the HTTP level.

A device shall, when authenticating RTSP and HTTP methods, use user/credentials from the same set of users/credentials that are used for the WS part and digest authentication [RFC 2617] shall be used for RTSP and HTTP.

5.13.2 User-based access control

5.13.2.1 General

The authorization framework described in 5.13 allows for authentication of service requests. Once a service request is authenticated, the device shall decide based on its access policy whether the requestor is authorized to receive the service.

This document defines a set of command for managing the user credentials, see 8.4. These commands allow associating users with the different user levels defined in 5.13.2.2.

A device may support the definition of a custom access policy by the device user through the get and set access policy operations defined in 8.4.

5.13.2.2 User levels

Each user is associated with exactly one of the following user levels:

1) Administrator
2) Operator
3) User
4) Anonymous

Unauthenticated users are placed into the anonymous category and a device shall not allow users to be added to the anonymous user level category.

5.13.2.3 Access classes for service requests

The service requests are classified into access classes based to their impact. The following access classes are defined:

- PRE_AUTH

  The service shall not require user authentication.

  Example: GetEndpointReference
- READ_SYSTEM

  The service reads system configuration information from the device.

  Example: GetNetworkInterfaces
- READ_SYSTEM_SENSITIVE

  The service reads sensitive (but not really confidential) system configuration information from the device.
- READ_SYSTEM_SECRET

  The service reads confidential system configuration information from the device.

  Example: GetSystemLog
- WRITE_SYSTEM

  The service causes changes to the system configuration of the device.

  Example: SetNetworkDefaultGateway
- UNRECOVERABLE

  The service causes unrecoverable changes to the system configuration of the device.

  Example: SetSystemFactoryDefault
- READ_MEDIA

  The service reads data related to recorded media.

  Example: GetRecordings
- ACTUATE

  The service affects the runtime behaviour of the system.

  Example: CreateRecordingJob

Table 7 defines for each access class which user levels are allowed access. A user of level c shall be granted access to a service request associated to access class r if and only if an "X" is present in the cell at column c and row r.

Table 7 – Access class to user level mapping

	Administrator	Operator	User	Anonymous
PRE_AUTH	X	X	X	X
READ_SYSTEM	X	X	X	
READ_SYSTEM_SENSITIVE	X	X		
READ_SYSTEM_SECRET	X			
WRITE_SYSTEM	X			
UNRECOVERABLE	X			
READ_MEDIA	X	X	X	
ACTUATE	X	X		

5.13.2.4 Default access policy

By default the device should enforce the default access policy defined by this document, and other specifications based on this document, which gives an acceptable level of security in many systems.

The default access policy is defined for each service request. It is done by identifying which access class that should be associated with that service request. See 5.7.1 for details on how this is done.

5.14 String representation

5.14.1 Character set

A device shall support the UTF-8 character set and it may support other character sets. If a client sends a request using UTF-8, the device shall always reply using the UTF-8 character set.

5.14.2 Allowed characters in strings

A device shall not have any restriction regarding legal characters in string that are not explicitly stated in this document or other specification based on this document.

5.15 Proprietary extensions

Proprietary extensions to this document shall be handled as follows:

1) It is strongly recommended to add proprietary extensions through defining new web services commands, in a namespace different from the target namespace, and not through extensions of the existing schema elements. This option allows full backward and forward interoperability with extensions from different vendors.
2) If new commands are not the best way to add proprietary extensions, it is recommeneded, whenever possible, to declare proprietary element extensions as attribute declarations instead of adding new schema elements in existing types. New proprietary attributes shall be declared in a namespace different from the target namespace. This option allows full backward and forward interoperability.
3) If it is not possible to declare the proprietary extensions using new commands or attributes, the proprietary extensions shall be declared adding a unique mandatory type of extension element before the extension point and then add all new proprietary elements in this extension element. The new elements shall be declared in a namespace different from the target namespace. Each proprietary extension element shall have minOccurs="0" and the proprietary extension element declarations in the schema shall be followed by an xs:any element declaration in the target namespace with minOccurs="0" and maxOccurs="unbounded".

6 IP configuration

The device and client communicate over an open or closed IP network. This document does not place any general restrictions or requirements on the network type. It shall be possible, however, to establish communication links between the entities according to the architectural framework specified in Clause 4. Device IP configuration includes parameters such as IP addresses and a default gateway.

A device shall have at least one network interface that gives it IP network connectivity. Similarly, the client shall have at least one network interface that gives IP connectivity and allows data communication between the device and the client.

Both device and client shall support IPv4 based network communication. The device and client should support IPv6 based network communication.

It shall be possible to make static IP configuration on the device using a network or local configuration interface.

A device should support dynamic IP configuration of link-local addresses according to [RFC 3927]. A device that supports IPv6 shall support stateless IP configuration according to [RFC 4862] and neighbour discovery according to [RFC 4861].

The device shall support dynamic IP configuration according to [RFC 2131]. A device that supports IPv6 shall support stateful IP configuration via DHCPv6 according to [RFC 3315] if signaled via the corresponding capability.

The device may support any additional IP configuration mechanism.

Network configuration of a device shall be provided via the device management service as specified in 8.3 and may additionally be provided through local interfaces. The latter is outside the scope of this document.

The default device configuration shall have both DHCP and dynamic link-local (stateless) address configuration enabled. Even if the device is configured through a static address configuration it should have the link-local address default enabled.

When a device is connected to an IPv4 network, address assignment priorities (link local versus routable address) should be done as recommended in [RFC 3927].

Further details regarding how the IP connectivity is achieved are outside the scope of this document.

7 Device discovery

7.1 General

A client searches for available devices using the dynamic Web services discovery protocol [XMLSOAP WS-Discovery].

A device compliant with this document shall implement the Target Service role as specified in [XMLSOAP WS-Discovery].

If necessary a client compliant with this document shall implement the Client role as specified in [XMLSOAP WS-Discovery].

[XMLSOAP WS-Discovery] describes the Universally Unique Identifier (UUID): URI format recommendation for endpoint references, but this document overrides this recommendation. Instead, the Uniform Resource Name: Universally Unique Identifier (URN:UUID) format is used [RFC 4122] (see 7.3.1).

7.2 Modes of operation

The device shall be able to operate in two modes:

- discoverable,
- non-discoverable.

A device in discoverable mode sends multicast Hello messages once connected to the network or sends its Status changes according to [XMLSOAP WS-Discovery]. In addition it always listens for Probe and Resolve messages and sends responses accordingly. A device in non-discoverable shall not listen to [XMLSOAP WS-Discovery] messages or send such messages.

The devices default behaviour shall be the discoverable mode. In order to thwart denial-of-service attacks, it shall be possible to set a device into non-discoverable mode through the operation defined in 8.4.19.

7.3 Discovery definitions

7.3.1 Endpoint reference

A device or an endpoint that takes the client role shall use a URN:UUID [RFC 4122] as the address property of its endpoint reference.

The device or an endpoint that takes the client role shall use a stable, globally unique identifier that is constant across network interfaces as part of its endpoint reference property. The combination of an wsadis:Address and wsadis:ReferenceProperties provide a stable and globally-unique identifier.

7.3.2 Hello

7.3.2.1 Types

A device shall include the device management service port type, i.e. tds:Device, in the `<d:Types>` declaration.

The following example shows how the type is encoded in the SOAP Hello body:

```
<d:Types>tds:Device</d:Types>.
```

The Hello message may include additional types.

7.3.2.2 Scopes

7.3.2.2.1 General

A device shall include the scope `<d:Scopes>` attribute with the scopes of the device in the Hello message.

The device scope is set by using [RFC 3986] URIs. This document defines scope attributes as follows:

The scheme attribute:onvif

The authority attribute:www.onvif.org

This implies that all scope URIs defined by this document have the following format:

```
onvif://www.onvif.org/<path>
```

A device may have other scope URIs. These URIs are not restricted by the scope attributes defined in this document.

Table 8 defines a set of scope parameters. Apart from these standardized parameters, it shall be possible to set any scope parameter as defined by the device owner. Scope parameters can be listed and set through the commands defined in 8.4.

Table 8 – Scope parameters

Category	Defined values	Description
Location	Any character string or path value.	The location defines the physical location of the device. The location value might be any string describing the physical location of the device.
Hardware	Any character string or path value.	A string or path value describing the hardware of the device. A device shall include at least one hardware entry into its scope list.
Name	Any character string or path value.	The searchable name of the device. A device shall include at least one name entry into its scope list.

A device shall include at least one fixed entry (defined by the device vendor) of the hardware and name categories respectively in the scopes list. A device may include any other additional scope attributes in the scopes list.

A device might include an arbitrary number of scopes in its scope list. This implies that one unit might for example define several different location scopes. A probe is matched against all scopes in the list.

7.3.2.2.2 Example

The following example illustrates the usage of the scope value. This is just an example, and not at all an indication of what type of scope parameter to be part of a device configuration. In this example we assume that the device is configured with the following scopes:

```
onvif://www.onvif.org/hardware/D1-566
onvif://www.onvif.org/location/country/china
onvif://www.onvif.org/location/city/bejing
onvif://www.onvif.org/location/building/headquarter
onvif://www.onvif.org/location/floor/R5
onvif://www.onvif.org/name/ARV-453
```

A client that probes for the device with scope onvif://www.onvif.org will get a match. Similarly, a probe for the device with scope:

onvif://www.onvif.org/location/country/china

will give a match. A probe with:

```
onvif://www.onvif.org/hardware/D1
```

will not give a match.

7.3.2.3 Addresses

A device shall include the <d:XAddrs> element with the address(es) for the device service in the Hello message. A URI shall be provided for each protocol (http, https) and externally available IP address.

The device should provide a port 80 device service entry in order to allow firewall traversal.

The IP addressing configuration principles for a device are defined in Clause 6.

7.3.3 Probe and probe match

For the device probe match types, scopes and addresses definitions, see 7.3.2 Hello.

A device shall at least support the following scope matching rule:

```
http://schemas.xmlsoap.org/ws/2005/04/discovery/
```

This scope matching definitions differs slightly from the definition in [XMLSOAP WS-Discovery] as [RFC 2396] is replaced by [RFC 3986].

A device shall include the `<d:XAddrs>` element with the addresses for the device service in a matching probe match message. The `<d:XAddrs>` element will in most cases only contain one address to the device management service as defined in 5.2.

7.3.4 Resolve and resolve match

This document requires end point address information to be included into Hello and Probe Match messages. In most cases, there is no need for the resolve and resolve match exchange. To be compatible with the [XMLSOAP WS-Discovery] specification, however, a device should implement the resolve match response.

7.3.5 Bye

A device should send a one-way Bye message when it prepares to leave a network as described in [XMLSOAP WS-Discovery].

7.3.6 SOAP fault messages

If an error exists with the multicast packet, the device and client should silently discard and ignore the request. Sending an error response is not recommended due to the possibility of packet storms if many devices send an error response to the same request. For completeness, unicast packet error handling is described below.

If a device receives a unicast Probe message and it does not support the matching rule, then the device may choose not to send a Probe Match, and instead generate a SOAP fault bound to SOAP 1.2 as follows:

[action] `http://schemas.xmlsoap.org/ws/2005/04/discovery/fault`

[Code] `s12:Sender`

[Subcode] `d:MatchingRuleNotSupported`

[Reason] `E.g., the matching rule specified is not supported`

[Detail]
```
<d: SupportedMatchingRules>
    List of xs:anyURI
</d: SupportedMatchingRules>
```

8 Device management

8.1 General

The device service is divided into five different categories: capabilities, network, system, I/O and security commands. This set of commands can be used to obtain information about the device capabilities and configurations or to set device configurations. A device shall support the device management service as specified in Clause B.1. A basic set of operations are

required for the device management service, other operations are recommended or optional to support. The detailed requirements are listed under the command descriptions.

8.2 Capabilities

8.2.1 Get WSDL URL

It is possible for an endpoint to request a URL that can be used to retrieve the complete schema and WSDL definitions of a device. The command gives in return a URL entry point where all the necessary product specific WSDL and schema definitions can be retrieved. The device shall provide a URL for WSDL and schema download through the GetWsdlUrl command.

Table 9 – GetWSDLUrl command

GetWsdlUrl	Access class: PRE_AUTH
Message name	**Description**
GetWsdlUrlRequest	*This is an empty message*
GetWsdlUrlResponse	*The requested URL.* xs:anyURI **WsdlUrl** [1][1]
Fault codes	**Description**
	No command specific faults!

8.2.2 Capability exchange

8.2.2.1 General

Any endpoint can ask for the capabilities of a device using the capability exchange request response operation. A device shall indicate its capabilities through the GetServices/GetServiceCapabilities commands.

The capability list includes references to the addresses (XAddr) of the service implementing the interface operations in the category.

8.2.2.2 GetServices

Returns a collection of the devices services and possibly their available capabilities. The returned capability response message is untyped to allow future addition of services, service revisions and service capabilities. All returned service capabilities shall be structured by different namespaces which are supported by a device.

A device shall support this method. For complementary information on the structure of GetServices response with capabilities, refer to Annex A.

Table 10 – GetServices command

GetServices	Access class: PRE_AUTH
Message name	**Description**
GetServicesRequest	*The message contains a request for all services in the device and possibly the capabilities for each service. If the Boolean IncludeCapability is set, then the response shall include the services capabilities.* boolean **IncludeCapability**[1][1]
GetServicesResponse	*The capability response message contains the requested information about the services.* tt:ServiceList [1][unbounded]
Fault codes	**Description**
	No command specific faults!

8.2.2.3 GetServiceCapabilities

This command returns the capabilities of the device service. A device shall support this command.

Table 11 and Table 12 describe how to interpret the indicated capabilities.

Table 11 – GetServiceCapabilities command

GetServiceCapabilities	Access class: PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*This is an empty message.*
GetServiceCapabilitiesResponse	*The capability response message contains the requested device capabilities using a hierarchical XML capability structure.* tds:DeviceServiceCapabilities **Capabilities** [1][1]
Fault codes	**Description**
	No command specific faults!

Table 12 – Capabilities in the GetServiceCapabilities command

Category	Capability	Description
Network	IPFilter	Indication if the device supports IP filtering control using the commands in 8.3.18, 8.3.19, 8.3.20 and 8.3.21.
	ZeroConfiguration	Indication if the device supports zero configuration according to the commands in 8.3.16 and 8.3.17.
	IPVersion6	Indication if the device supports IP version 6.
	DynDNS	Indication if the device supports Dynamic DNS configuration according to 8.3.8 and 8.3.9 .
	Dot11Configuration	Indication if the device supports IEEE802.11 configuration as specified in 8.3.22
	HostnameFromDHCP	Indicates whether retrieval of hostname from DHCP is supported by the device.
	NTP	Indicates the maximum number of supported NTP servers by the devices SetNTP command.
	Dot1XConfigurations	Indicates the maximum number of Dot1X configurations supported by the device, see 8.5.8
	DHCPv6	Indicates support for Stateful IPv6 DHCP.
System	DiscoveryResolve	Indication if the device responses to resolve requests as described in 7.3.4.
	DiscoveryBye	Indication if the device sends Bye messages as described in 7.3.5
	SystemBackup	Indication if the device supports system backup and restore as specified in 8.4.3 and 8.4.5
	FirmwareUpgrade	Indication if the device supports firmware upgrade as specified in 8.4.10.
	SystemLogging	Indication if the device supports system log retrieval as specified in 8.4.11.
	HttpSystemBackup	Indication if the device supports system backup and restore using HTTP GET and POST.
	HttpFirmwareUpgrade	Indication if the device supports firmware upgrade using HTTP POST.
	HTTPSystemLogging	Indication if the device supports retrieval of system log using HTTP Get, see 8.4.2.
	HTTPSupportInformation	Indication if the device supports retrieval of support information using HTTP Get, see 8.4.2.
Security	TLS1.0	Support of TLS 1.0.

Category	Capability	Description
	TLS1.1	Support of TLS 1.1.
	TLS1.2	Support of TLS 1.2.
	OnboardKeyGeneration	Indication if the device supports onboard key generation and creation of self-signed certificates as specified in 8.5.9.
	AccessPolicyConfig	Indication if the device supports retrieving and loading device access control policy according to 8.5.2 and 8.5.3.
	DefaultAccessPolicy	Indicates if the device supports the default access policies as defined in 5.13.2.4.
	HttpDigest	Indication if the device supports the HTTP digest authentication.
	Dot1X	Indication if the device supports IEEE 802.1X port-based network authentication
	SupportedEAPMethod	List of supported EAP method types. The numbers correspond to the IANA [EAP-Registry].
	RemoteUserHandling	Indication if the device supports remote user handling and the corresponding methods defined in 8.5.22 and 8.5.23.
	MaxUsers	The maximum number of users that the device supports

8.3 Network

8.3.1 Get hostname

This operation is used by an endpoint to get the hostname from a device. The device shall return its hostname configurations through the GetHostname command as described in Table 13.

Table 13 – GetHostname command

GetHostname	Access Class: PRE_AUTH
Message name	**Description**
GetHostnameRequest	*This is an empty message.*
GetHostnameResponse	*This message contains:* • *"FromDHCP": True if the hostname is obtained via DHCP* • *"Name": The hostname. In case of DHCP the hostname has been obtained from the DHCP server.* xs:boolean **FromDHCP** [1][1] xs:token **Name** [0][1]
Fault codes	**Description**
	No command specific faults!

8.3.2 Set hostname

This operation sets the hostname on a device. It shall be possible to set the device hostname configurations through the SetHostname command as described in Table 14. Attention: a call to SetDNS may result in overriding a previously set hostname.

A device shall accept strings formated according to RFC 1123:1989, Section 2.1 or alternatively to RFC 952; other string shall be considered as invalid strings.

A device shall try to retrieve the name via DHCP when the HostnameFromDHCP capability is set and an empty name string is provided.

Table 14 – SetHostname command

SetHostname	Access class: WRITE_SYSTEM
Message name	**Description**
SetHostnameRequest	*This message contains:* • *"Name": The hostname. If Name is an empty string hostname should be retrieved from DHCP, otherwise the specified Name shall be used.* xs:token **Name** [1][1]
SetHostnameResponse	*This is an empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidHostname	*The requested hostname cannot be accepted by the device.*

8.3.3 Set hostname from DHCP

This operation, as described in Table 15, controls whether the hostname shall be retrieved from DHCP.

A device shall support this command if support is signalled via the HostnameFromDHCP capability. Depending on the device implementation the change may only become effective after a device reboot.

Table 15 – SetHostnameFromDHCP command

SetHostnameFromDHCP	Access class: WRITE_SYSTEM
Message name	**Description**
SetHostnameFromDHCPRequest	*This message contains:* • *"FromDHCP": True if the hostname shall be obtained via DHCP.* xs:boolean **FromDHCP**[1][1]
SetHostnameFromDHCPResponse	*An indication if a reboot is needed in case of changes in the hostname settings.* xs:boolean **RebootNeeded** [1][1]
Fault codes	**Description**
	No command specific faults!

8.3.4 Get DNS settings

This operation gets the DNS settings from a device as described in Table 16. The device shall return its DNS configurations through the GetDNS command.

Table 16 – GetDNS command

GetDNS	Access class: READ_SYSTEM
Message name	**Description**
GetDNSRequest	*This is an empty message.*
GetDNSResponse	*This message contains:* • *"FromDHCP": True if the DNS servers are obtained via DHCP.* • *"SearchDomain": The domain(s) to search if the hostname is not fully qualified.* • *"DNSFromDHCP": A list of DNS servers obtained via DHCP in case FromDHCP is equal to true. This means that the resolved addresses in the field DNSFromDHCP are coming from DHCP and describe the configuration status.* • *"DNSManual": A list of manually given DNS servers* xs:boolean **FromDHCP** [1][1] xs:token **SearchDomain** [0][unbounded] tt:IPAddress **DNSFromDHCP** [0][unbounded] tt:IPAddress **DNSManual** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.5 Set DNS settings

This operation sets the DNS settings on a device as described in Table 17. It shall be possible to set the device DNS configurations through the SetDNS command.

Table 17 – SetDNS command

SetDNS	Access class: WRITE_SYSTEM
Message name	**Description**
SetDNSRequest	*This message contains:* • *"FromDHCP": True if the DNS servers are obtained via DHCP* • *"SearchDomain": The domain(s) to search if the hostname is not fully qualified.* • *"DNSManual": A list of manually given DNS servers* xs:boolean **FromDHCP** [1][1] xs:token **SearchDomain** [0][unbounded] tt:IPAddress **DNSManual** [0][unbounded]
SetDNSResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*

8.3.6 Get NTP settings

This operation gets the NTP settings from a device as described in Table 18. If the device supports NTP, it shall be possible to get the NTP server settings through the GetNTP command.

Table 18 – GetNTP command

GetNTP	Access class: READ_SYSTEM
Message name	**Description**
GetNTPRequest	*This is an empty message.*
GetNTPResponse	*This message contains:* • *"FromDHCP": True if the NTP servers are obtained via DHCP.* • *"NTPFromDHCP": A list of NTP servers obtained via DHCP in case FromDHCP is equal to true. This means that the NTP server addresses in the field NTPFromDHCP are coming from DHCP and describe the current configuration status.* • *"NTPManual": A list of manually given NTP servers* xs:boolean **FromDHCP** [1][1] tt:NetworkHost **NTPFromDHCP** [0][unbounded] tt:NetworkHost **NTPManual** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.7 Set NTP settings

This operation sets the NTP settings on a device as described in Table 19. If support for NTP is signalled via the NTP capability, it shall be possible to set the NTP server settings through the SetNTP command.

A device shall accept string formated according to RFC 1123 section 2.1; other strings shall be considered as invalid strings.

Changes to the NTP server list shall not affect the clock mode DateTimeType. Use SetSystemDateAndTime to activate NTP operation.

Table 19 – SetNTP command

SetNTP	Access class: WRITE_SYSTEM
Message name	**Description**
SetNTPRequest	*This message contains:* • *"FromDHCP": True if the NTP servers are obtained via DHCP.* • *"NTPManual": A list of manually given NTP servers when they are not obtained via DHCP.* xs:boolean **FromDHCP** [1][1] tt:NetworkHost **NTPManual** [0][unbounded]
SetNTPResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidDnsName	*The suggested NTP server name is invalid.*
env:Sender ter:InvalidArgVal ter:TimeSyncedToNtp	*Current DateTimeType requires an NTP server.*

8.3.8 Get dynamic DNS settings

This operation gets the dynamic DNS settings from a device as described in Table 20. If the device supports dynamic DNS as specified in [RFC 2136] and [RFC 4702], it shall be possible to get the type, name and TTL through the GetDynamicDNS command.

Table 20 – GetDynamicDNS command

GetDynamicDNS	Access class: READ_SYSTEM
Message name	**Description**
GetDynamicDNSRequest	*This is an empty message.*
GetDynamicDNSResponse	*This message contains:* • *"Type": The type of update. There are three possible types: the device desires no update (NoUpdate), the device wants the DHCP server to update (ServerUpdates) and the device does the update itself (ClientUpdates).* • *"Name": The DNS name in case the device does the update.* • *"TTL": Time to live.* *tt:DynamicDNSType **Type** [1][1]* *tt:DNSName **Name** [0][1]* *xs:duration **TTL** [0][1]*
Fault codes	**Description**
	No command specific faults!

8.3.9 Set dynamic DNS settings

This operation sets the dynamic DNS settings on a device as described in Table 21. If the device supports dynamic DNS as specified in [RFC 2136] and [RFC 4702], it shall be possible to set the type, name and TTL through the SetDynamicDNS command.

Table 21 – SetDynamicDNS command

SetDynamicDNS	Access class: WRITE_SYSTEM
Message name	**Description**
SetDynamicDNSRequest	*This message contains:* • *"Type": The type of update. There are three possible types: the device desires no update (NoUpdate), the device wants the DHCP server to update (ServerUpdates) and the device does the update itself (ClientUpdates).* • *"Name": The DNS name in case the device does the update.* • *"TTL": Time to live.* *tt:DynamicDNSType **Type** [1][1]* *tt:DNSName **Name** [0][1]* *xs:duration **TTL** [0][1]*
SetDynamicDNSResponse	*This is an empty message.*
Fault codes	**Description**
	No command specific faults!

8.3.10 Get network interface configuration

This operation gets the network interface configuration from a device as described in Table 22. The device shall support return of network interface configuration settings as defined by the NetworkInterface type through the GetNetworkInterfaces command.

Table 22 – GetNetworkInterfaces command

GetNetworkInterfaces	Access class: READ_SYSTEM
Message name	**Description**
GetNetworkInterfacesRequest	*This is an empty message.*
GetNetworkInterfacesResponse	*This message contains an array of device network interfaces.* tt:NetworkInterface **NetworkInterfaces** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.11 Set network interface configuration

This operation sets the network interface configuration on a device as described in Table 23. The device shall support network configuration of supported network interfaces through the SetNetworkInterfaces command.

If a device responds with RebootNeeded set to false, the device can be reached via the new IP address without further action. A client should be aware that a device may not be responsive for a short period of time until it signals availability at the new address via the discovery Hello messages as defined in 7.3.2.

If a device responds with RebootNeeded set to true, it will be further available under its previous IP address. The settings will only be activated when the device is rebooted via the SystemReboot command.

For interoperability with a client unaware of the IEEE 802.11 extension a device shall retain its IEEE 802.11 configuration if the IEEE 802.11 configuration element is not present in the request.

Table 23 – SetNetworkInterfaces command

SetNetworkInterfaces	Access class: WRITE_SYSTEM
Message name	**Description**
SetNetworkInterfacesRequest	*This message contains:* • *"InterfaceToken": The token of the network interface to operate on.* • *"NetworkInterface": The network interface to configure.* tt:ReferenceToken **InterfaceToken** [1][1] tt:NetworkInterfaceSetConfiguration **NetworkInterface** [1][1]
SetNetworkInterfacesResponse	*This message contains:* • *"RebootNeeded": An indication if a reboot is needed in case of changes in the network settings.* xs:boolean RebootNeeded [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidMtuValue	*The MTU value is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceSpeed	*The suggested speed is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceType	*The suggested network interface type is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 Configuration is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidSecurityMode	*The selected security mode is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidStationMode	*The selected station mode is not supported.*
env:Sender ter:InvalidArgVal ter:MissingDot11	*IEEE 802.11 value is missing in the security configuration.*
env:Sender ter:InvalidArgVal ter:MissingPSK	*PSK value is missing in security configuration.*
env:Sender ter:InvalidArgVal ter:MissingDot1X	*IEEE 802.1X value in security configuration is missing or non existing.*
env:Sender ter:InvalidArgVal ter:IncompatibleDot1X	*IEEE 802.1X value in security configuration is incompatible with the network interface.*
env:Receiver ter:ActionNotSupported ter:InvalidDHCPv6	*The requested stateful DHCPv6 mode is not supported.*

8.3.12 Get network protocols

This operation gets defined network protocols from a device as described in Table 24. The device shall support the GetNetworkProtocols command returning configured network protocols.

Table 24 – GetNetworkProtocols command

GetNetworkProtocols	Access class: READ_SYSTEM
Message name	**Description**
GetNetworkProtocolsRequest	*This is an empty message.*
GetNetworkProtocolsResponse	*This message returns an array of defined protocols supported by the device. There are three protocols defined, HTTP, HTTPS and RTSP. The following parameters can be retrieved for each protocol:* • *Port* • *Enable/disable* tt:NetworkProtocol **NetworkProtocols** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.13 Set network protocols

This operation configures defined network protocols on a device as described in Table 25. The device shall support configuration of defined network protocols through the SetNetworkProtocols command.

Table 25 – SetNetworkProtocols command

SetNetworkProtocols	Access class: WRITE_SYSTEM
Message name	**Description**
SetNetworkProtocolsRequest	*This message configures one or more defined network protocols supported by the device. There are currently three protocols defined, HTTP, HTTPS and RTSP. The following parameters can be set for each protocol:* • *Port* • *Enable/disable* tt:NetworkProtocol **NetworkProtocols** [1][unbounded]
SetNetworkProtocolsResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:ServiceNotSupported	*The supplied network service is not supported.*
env:Sender ter:InvalidArgVal ter:PortAlreadyInUse	*The selected port is already in use.*
env:Receiver ter:ActionNotSupported ter:EnablingTlsFailed	*The device doesn't support TLS or TLS is not configured appropriately.*

8.3.14 Get default gateway

This operation gets the default gateway settings from a device as described in Table 26. The device shall support the GetNetworkDefaultGateway command returning configured default gateway address(es).

Table 26 – GetNetworkDefaultGateway command

GetNetworkDefaultGateway	Access class: READ_SYSTEM
Message name	**Description**
GetNetworkDefaultGateway-Request	*This is an empty message.*
GetNetworkDefaultGateway-Response	*This message contains:* • *"IPv4Address": The default IPv4 gateway address(es).* • *"IPv6Address": The default IPv6 gateway address(es).* tt:IPv4Address **IPv4Address** [0][unbounded] tt:IPv6Address **IPv6Address** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.15 Set default gateway

This operation sets the default gateway settings on a device as described in Table 27. The device shall support configuration of default gateway through the SetNetworkDefaultGateway command.

Table 27 – SetNetworkDefaultGateway command

SetNetworkDefaultGateway	Access class: WRITE_SYSTEM
Message name	**Description**
SetNetworkDefaultGateway-Request	*This message contains:* • *"IPv4Address": The default IPv4 gateway address(es).* • *"IPv6Address": The default IPv6 gateway address(es).* tt:IPv4Address **IPv4Address** [0][unbounded] tt:IPv6Address **IPv6Address** [0][unbounded]
SetNetworkDefaultGateway-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidGatewayAddress	*The supplied gateway address was invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*

8.3.16 Get zero configuration

This operation gets the zero-configuration from a device as described in Table 28. If the device supports dynamic IP configuration according to [RFC 3927], it shall support the return of IPv4 zero configuration address and status through the GetZeroConfiguration command

Table 28 – GetZeroConfiguration command

GetZeroConfiguration	Access class: READ_SYSTEM
Message name	**Description**
GetZeroConfigurationRequest	*This is an empty message.*
GetZeroConfigurationResponse	*This message contains:* • *"InterfaceToken": The token of the network interface* • *"Enabled": If zero configuration is enabled or not.* • *"Addresses": The IPv4 zero configuration address(es).* tt:ReferenceToken **InterfaceToken** [1][1] xs:boolean **Enabled** [1][1] tt:IPv4Addresses **Address** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.17 Set zero configuration

This operation sets the zero-configuration on the device as described in Table 29. If the device supports dynamic IP configuration according to [RFC 3927], it shall support the configuration of IPv4 zero configuration address and status through the SetZeroConfiguration command.

Table 29 – SetZeroConfiguration command

SetZeroConfiguration	Access class: WRITE_SYSTEM
Message name	**Description**
SetZeroConfigurationRequest	*This message contains:* • *"InterfaceToken": The token of the network interface to operate on.* • *"Enabled": If zero configuration is enabled or not.* tt:ReferenceToken **InterfaceToken** [1][1] xs:boolean **Enabled** [1][1]
SetZeroConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist*

8.3.18 Get IP address filter

This operation gets the IP address filter settings from a device as described in Table 30. If the device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses), the device shall support the GetIPAddressFilter command.

Table 30 – GetIPAddressFilter command

GetIPAddressFilter	Access class: READ_SYSTEM
Message name	**Description**
GetIPAddressFilterRequest	*This is an empty message.*
GetIPAddressFilterResponse	*This message contains:* • *"Type": Sets if the filter should deny or allow access.* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:IPAddressFilterType **Type** [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.19 Set IP address filter

This operation sets the IP address filter settings on a device as described in Table 31. If the device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses), the device shall support configuration of IP filtering rules through the SetIPAddressFilter command.

Table 31 – SetIPAddressFilter command

SetIPAddressFilter	Access class: WRITE_SYSTEM
Message name	**Description**
SetIPAddressFilterRequest	*This message contains:* • *"Type": Sets if the filter should deny or allow access.* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:IPAddressFilterType **Type** [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
SetIPAddressFilterResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*

8.3.20 Add an IP filter address

This operation adds an IP filter address to a device as described in Table 32. If the device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses), the device shall support adding of IP filtering addresses through the AddIPAddressFilter command.

The value of the Type field shall be ignored by the device. Use SetIPAddressFilter to set the type.

Table 32 – AddIPAddressFilter command

AddIPAddressFilter	Access class: WRITE_SYSTEM
Message name	**Description**
AddIPAddressFilterRequest	*This message contains:* • *Type": Sets if the filter should deny or allow access.* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:IPAddressFilterType Type [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
AddIPAddressFilterResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:IPFilterListIsFull	*It is not possible to add more IP filters since the IP filter list is full.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*

8.3.21 Remove an IP filter address

This operation deletes an IP filter address from a device as described in Table 33. If the device supports device access control based on IP filtering rules (denied or accepted ranges of IP addresses), the device shall support deletion of IP filtering addresses through the RemoveIPAddressFilter command.

The value of the Type field shall be ignored by the device.

Table 33 – RemoveIPAddressFilter command

RemoveIPAddressFilter	Access class: WRITE_SYSTEM
Message name	**Description**
RemoveIPAddressFilterRequest	*This message contains:* • *"Type": Value of this field is ignored in this command.* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:IPAddressFilterType Type [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
RemoveIPAddressFilter-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:NoIPv6Address	*The IPv6 address to be removed does not exist.*
env:Sender ter:InvalidArgVal ter:NoIPv4Address	*The IPv4 address to be removed does not exist.*

8.3.22 IEEE 802.11 configuration

8.3.22.1 General

Requirements in 8.3.22 are only valid for a device that signals IEEE 802.11 support via its Network Dot11Configuration capability. In 8.3.22 the term "the device" is used to indicate a device with IEEE 802.11 support.

The device shall support IEEE 802.11 configuration and shall as a response to the GetNetworkInterfaces method return "ieee80211 (71)" as the IANA-IfTypes for the 802.11 interface(s).

A device shall not return any link element in the GetNetworkInterfaces reply and it shall ignore any Link element in the SetNetworkInterfaces request.

The device should support that each IEEE 802.11 network interface can have more than one alternative IEEE 802.11 configurations attached to it.

IEEE 802.11 configuration is supported through an optional IEEE 802.11 configuration element in the get and set network configuration element. The following information is handled:

- SSID,
- Station mode,
- Multiple wireless network configuration,
- Security configuration.

The following operations are used to help manage the wireless configuration:

- Get IEEE802.11 capabilities,
- Get IEEE802.11 status,
- Scan available IEEE802.11 networks.

8.3.22.2 SSID

The device shall support configuration of the SSID.

8.3.22.3 Station mode

The device shall support the infrastructure station mode.

The device may support the ad-hoc network station mode. The actual configuration needed for ad-hoc network station mode, including manual configuration of the channel number, is outside the scope of this document. However, to allow for devices that support ad-hoc network station modes, the specification allows for selecting (and reporting) this mode.

8.3.22.4 Multiple wireless network configuration

Each IEEE 802.11 configuration shall be identified with an alias (identifier). The alias shall be unique within a network interface configuration. The client shall supply the alias in the SetNetworkInterfaces request. If the client wants to update an existing wireless configuration the same alias shall be used. A wireless configuration, including the alias, shall only exist while it is part of a network interface configuration.

For the device to be able to prioritize between multiple alternative IEEE802.11 configurations an optional priority value can be used, a higher value means a higher priority. If several wireless configurations have the same priority value the order between those configurations is undefined.

The actual algorithm used by the device to enable an IEEE 802.11 network from the prioritized list of IEEE 802.11 configurations is outside the scope of this document.

8.3.22.5 Security configuration

8.3.22.5.1 General

The security configuration contains the chosen security mode and the configuration needed for that mode. The following security modes are supported:

- None,
- PSK (Pre Shared Key) (WPA- and WPA2-Personal),
- IEEE 802.1X-2004 (WPA- and WPA2-Enterprise).

Configuration of WEP security mode is outside the scope of this document however, to allow for devices that support WEP security mode this document allows for selecting (and reporting) this mode.

For data confidentiality and integrity the device shall, in accordance with the [IEEE 802.11-2007] specification, support the CCMP algorithm and the device may support the TKIP algorithm.

The algorithm can either be manually (CCMP, TKIP) or automatically (Any) selected. In manual selected mode the same algorithm shall be used for both the pairwise and group cipher. To be able to support other algorithms an “extended” value is available.

The device shall support both the manually and the automatically selected mode.

8.3.22.5.2 None mode

The device shall support the “None” security mode.

8.3.22.5.3 PSK mode

The device shall support the PSK security mode.

To minimize the risk of compromizing the PSK, the device should not transmit any PSK to a client, furthermore it shall not return the PSK in a response to a GetNetworkInterfaces operation call.

For adding a wireless configuration with the PSK security mode the following rules apply:

- A client shall include a PSK value in the SetNetworkInterfaces request.
- The device shall check that a PSK value was supplied, if not the device shall return an error.

For updating wireless configuration with the PSK security mode the following rules apply:

- If the client wants to retain the PSK value it should not include the PSK value in the SetNetworkInterfaces request.
- The device receiving a SetNetworkInterfaces request without a PSK value shall retain its PSK value.

The [IEEE 802.11-2007] standard states that the PSK should be distributed to the STA with some out-of-band method.

8.3.22.5.4 IEEE 802.1X-2004 mode

The device should support the IEEE 802.1X security mode. For more detailed requirements about the IEEE 802.1X-2004 security mode see [IEEE 802.1X configuration].

8.3.22.6 Get Dot11 capabilities

This operation returns the IEEE802.11 capabilities, see Table 34 and Table 35. The device shall support this operation.

Table 34 – GetDot11Capabilities

GetDot11Capabilities	Access class: READ_SYSTEM
Message name	**Description**
GetDot11CapabilitiesRequest	*This is an empty message*
GetDot11CapabilitesResponse	*This message contains:* *tt:Dot11Capabilities* ***Capabilities*** *[1][1]*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 configuration is not supported.*

Table 35 – IEEE 802.11 capabilities

Capability	Description
TKIP	Indication if the device supports the TKIP algorithm.
ScanAvailableNetworks	Indication if the device supports the ScanAvailableIEEE802.11Networks operation.
MultipleConfiguration	Indication if the device supports multiple alternative IEEE 802.11 configurations.
AdHocStationMode	Indication if the device supports the Ad-Hoc station mode.
WEP	Indication if the device supports the WEP security mode.

8.3.22.7 Get IEEE 802.11 status

This operation returns the status of a wireless network interface. The device shall support this command as described in Table 36. The following status can be returned:

- SSID (shall),
- BSSID (should),
- Pair cipher (should),
- Group cipher (should),
- Signal strength (should),
- Alias of active wireless configuration (shall).

Table 36 – GetDot11Status

GetDot11Status	Access class: READ_SYSTEM
Message name	**Description**
GetDot11StatusRequest	*This message contains:* *tt:ReferenceToken* ***InterfaceToken*** *[1][1]*
GetDot11StatusResponse	*This message contains:* *tt:Dot11Status* ***Status*** *[1][1]*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 configuration is not supported.*
env:Sender ter:InvalidArgVal ter:NotDot11	*The interface is not an IEEE 802.11 interface.*
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist.*
env:Receiver ter:Action ter:NotConnectedDot11	*IEEE 802.11 network is not connected.*

8.3.22.8 Scan available IEEE 802.11 networks

This operation returns a list of the wireless networks in the range of the device. A device should support this operation as described in Table 37. The following status can be returned for each network:

- SSID (shall),
- BSSID (should),
- Authentication and key management suite(s) (should),
- Pair cipher(s) (should),
- Group cipher(s) (should),
- Signal strength (should).

Table 37 – ScanAvailableDot11Networks

ScanAvailableDot11Networks	Access class: READ_SYSTEM
Message name	**Description**
ScanAvailableDot11-NetworksRequest	*This message contains:* *tt:ReferenceToken* ***InterfaceToken*** *[1][1]*
ScanAvailableDot11-NetworksResponse	*This message contains:* *tt:Dot11AvailableNetworks* ***Networks*** *[0][unbounded]*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 configuration is not supported.*
env:Sender ter:InvalidArgVal ter:NotDot11	*The interface is not an IEEE 802.11 interface.*
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist.*
env;Receiver ter:ActionNotSupported ter:NotScanAvailable	*ScanAvailableDot11Networks is not supported.*

8.4 System

8.4.1 Device information

This operation gets device information, such as manufacturer, model and firmware version from a device. The device shall support the return of device information through the GetDeviceInformation command as described in Table 38.

Table 38 – GetDeviceInformation command

GetDeviceInformation	Access class: READ_SYSTEM
Message name	**Description**
GetDeviceInformationRequest	*This is an empty message.*
GetDeviceInformationResponse	*This message contains:* xs:string **Manufacturer** [1][1] xs:string **Model** [1][1] xs:string **FirmwareVersion** [1][1] xs:string **SerialNumber** [1][1] xs:string **HardwareId** [1][1]
Fault codes	**Description**
	No command specific faults!

8.4.2 Get system URIs

This operation is used to retrieve URIs from which system information may be downloaded using HTTP. URIs may be returned for the following system information:

- System logs. Multiple system logs may be returned, of different types. The exact format of the system logs is outside the scope of this document.
- Support information. This consists of arbitrary device diagnostics information from a device. The exact format of the diagnostic information is outside the scope of this document.
- System backup. The received file is a backup file that can be used to restore the current device configuration at a later date. The exact format of the backup configuration file is outside the scope of this document.

If the device allows retrieval of system logs, support information or system backup data, it should make them available via HTTP GET. If it does, it shall support the GetSystemUris command as described in Table 39.

Table 39 – GetSystemUris command

GetSystemUris	Access class: READ_SYSTEM
Message name	**Description**
GetSystemUrisRequest	*This is an empty message.*
GetSystemUrisResponse	*This message contains the URIs from which the various system information components may be downloaded.* tt:SystemLogUriList **SystemLogUris [0][1]** xs:anyURI **SupportInfoUri [0][1]** xs:anyURI **SystemBackupUri [0][1]**
Fault codes	**Description**
	No command specific faults!

8.4.3 Backup

This operation retrieves system backup configuration file(s) from a device as described in Table 40. The device should support return of backup configuration file(s) through the GetSystemBackup command. The backup is returned with reference to a name and mime-type together with binary data. The exact format of the backup configuration files is outside the scope of this document.

The backup configuration file(s) shall be transmitted through MTOM [W3C SOAP-MTOM].

Table 40 – GetSystemBackup command

GetSystemBackup	Access class: READ_SYSTEM_SECRET
Message name	**Description**
GetSystemBackupRequest	*This is an empty message.*
GetSystemBackupResponse	*The get system backup response message contains the system backup configuration files(s).* tt:BackupFile **BackupFiles** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

8.4.4 Restore

This operation restores the system backup configuration files(s) previously retrieved from a device as described in Table 41. The device should support restore of backup configuration file(s) through the RestoreSystem command. The exact format of the backup configuration file(s) is outside the scope of this document. If the command is supported, it shall accept backup files returned by the GetSystemBackup command.

The backup configuration file(s) shall be transmitted through MTOM [W3C SOAP-MTOM].

Table 41 – RestoreSystem command

RestoreSystem	Access class: UNRECOVERABLE
Message name	**Description**
RestoreSystemRequest	*This message contains the system backup file(s).* tt:BackupFile **BackupFiles** [1][unbounded]
RestoreSystemResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidBackupFile	*The backup file(s) are invalid.*

8.4.5 Start system restore

This operation initiates a system restore from backed up configuration data using the HTTP POST mechanism. The response to the command includes an HTTP URL to which the backup file may be uploaded. The actual restore takes place as soon as the HTTP POST operation has completed. Devices should support system restore through the StartSystemRestore command. The exact format of the backup configuration data is outside the scope of this document.

System restore over HTTP may be achieved as described in Table 42 using the following steps:

1) Client calls StartSystemRestore.
2) Device service responds with upload URI.
3) Client transmits the configuration data to the upload URI using HTTP POST.
4) Server applies the uploaded configuration, then reboots if necessary.

If the system restore fails because the uploaded file was invalid, the HTTP POST response shall be “415 Unsupported Media Type”. If the system restore fails due to an error at the device, the HTTP POST response shall be “500 Internal Server Error”.

The value of the Content-Type header in the HTTP POST request shall be “application/octet-stream”.

Table 42 – StartSystemRestore command

StartSystemRestore	Access class: UNRECOVERABLE
Message name	**Description**
StartSystemRestoreRequest	*This is an empty message*
StartSystemRestoreResponse	*This message contains* ▪ *A URL to which the system configuration file may be uploaded.* ▪ *An optional duration that indicates how long the device expects to be unavailable after the upload is complete.* xs:anyURI **UploadUri [1][1]** xs:duration **ExpectedDownTime [0][1]**
Fault codes	**Description**
	No command-specific faults.

8.4.6 Get system date and time

This operation gets the device system date and time as described in Table 43. The device shall support the return of the daylight saving setting and of the manual system date and time (if applicable) or indication of NTP time (if applicable) through the GetSystemDateAndTime command.

A device shall provide the UTCDateTime information although the item is marked as optional to ensure backward compatibility.

Table 43 – GetSystemDateAndTime command

GetSystemDateAndTime	Access class: PRE_AUTH
Message name	**Description**
GetSystemDateAndTime-Request	*This is an empty message.*
GetSystemDateAndTime-Response	*This message contains the date and time information of the device.* • *"DateTimeType": If the system time and date are set manually or by NTP* • *"DaylightSavings": Daylight savings on or off* • *"TimeZone": The time zone as it is defined in POSIX 1003.1 section 8.3* • *"UTCDateTime": The time and date in UTC.* • *"LocalDateTime": The local time and date of the device* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1] tt:DateTime **UTCDateTime** [0][1] tt:DateTime **LocalDateTime** [0][1]
Fault codes	**Description**
	No command specific faults!

8.4.7 Set system date and time

This operation sets the device system date and time as described in Table 44. The device shall support the configuration of the daylight saving setting and of the manual system date and time (if applicable) or indication of NTP time (if applicable) through the SetSystemDateAndTime command. A device shall consider a Timezone which is not formed according to the rules of [IEEE 1003.1]:2004, Section 8.3 as invalid.

If system time and date are set manually, the client shall include UTCDateTime or LocalDateTime in the request.

The DayLightSavings flag should be set to true to activate any DST settings of the TimeZone string. Clear the DayLightSavings flag if the DST portion of the TimeZone settings should be ignored.

Table 44 – SetSystemDateAndTime command

SetSystemDateAndTime	Access class: WRITE_SYSTEM
Message name	**Description**
SetSystemDateAndTime-Request	*This message contains the date and time information of the device.* • *"DateTimeType": If the system time and date are set manually or by NTP* • *"DaylightSavings": Automatically adjust daylight savings if defined in TimeZone.* • *"TimeZone": The time zone is defined in POSIX 1003.1 section 8.3* • *"UTCDateTime": The time and date in UTC. If DateTimeType is NTP, UTCDateTime has no meaning.* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1] tt:DateTime **UTCDateTime** [0][1]
SetSystemDateAndTime-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidTimeZone	*An invalid time zone was specified.*
env:Sender ter:InvalidArgVal ter:InvalidDateTime	*An invalid date or time was specified.*
env:Sender ter:InvalidArgVal ter:NtpServerUndefined	*Cannot switch DateTimeType to NTP because no NTP server is defined.*

8.4.8 Factory default

This operation reloads parameters of a device to their factory default values as described in Table 45. The device shall support hard and soft factory default through the SetSystemFactoryDefault command. The meaning of soft factory default is device product-specific and vendor-specific. The effect of a soft factory default operation is not fully defined. However, it shall be guaranteed that after a soft reset the device is reachable on the same IP address as used before the reset. This means that basic network settings like IP address, subnet and gateway or DHCP settings are kept unchanged by the soft reset.

Table 45 – SetSystemFactoryDefault command

SetSystemFactoryDefault	Access class: UNRECOVERABLE
Message name	**Description**
SetSystemFactoryDefault-Request	*This message contains the types of factory default to perform.* • *"Hard": All parameters are set to their factory default value* • *"Soft": All parameters except device vendor specific parameters are set to their factory default values* tt:FactoryDefaultType **FactoryDefault** [1][1]
SetSystemFactoryDefault-Response	*This is an empty message.*
Fault codes	**Description**
	No command specific faults!

8.4.9 Firmware upgrade

This operation upgrades a device firmware version as described in Table 46. After a successful upgrade the response message is sent before the device reboots. The device should support firmware upgrade through the UpgradeSystemFirmware command. The exact format of the firmware data is outside the scope of this document.

The firmware shall be transmitted through MTOM [W3C SOAP-MTOM].

Table 46 – UpgradeSystemFirmware command

UpgradeSystemFirmware	Access class: UNRECOVERABLE
Message name	**Description**
UpgradeSystemFirmware-Request	*This message contains the firmware used for the upgrade. The firmware upgrade is "soft" meaning that all parameters keep their current value.* tt:AttachmentData **Firmware** [1][1]
UpgradeSystemFirmware-Response	*This message contains a "Message" string allowing the device to report back a message to the client as for example "Upgrade successful, rebooting in x seconds."* xs:string **Message** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:InvalidFirmware	*The firmware was invalid, i.e., not supported by this device.*
env:Receiver ter:Action ter:FirmwareUpgrade-Failed	*The firmware upgrade failed.*

8.4.10 Start firmware upgrade

This operation initiates a firmware upgrade using the HTTP POST mechanism as described in Table 47. The response to the command includes an HTTP URL to which the upgrade file may be uploaded. The actual upgrade takes place as soon as the HTTP POST operation has completed. The device should support firmware upgrade through the StartFirmwareUpgrade command. The exact format of the firmware data is outside the scope of this document.

Firmware upgrade over HTTP may be achieved using the following steps:

1) Client calls StartFirmwareUpgrade.
2) Device service responds with upload URI and optional delay value.
3) Client waits for delay duration if specified by server.
4) Client transmits the firmware image to the upload URI using HTTP POST.
5) Server reprograms itself using the uploaded image, then reboots.

If the firmware upgrade fails because the upgrade file was invalid, the HTTP POST response shall be "415 Unsupported Media Type". If the firmware upgrade fails due to an error at the device, the HTTP POST response shall be "500 Internal Server Error".

The value of the Content-Type header in the HTTP POST request shall be "application/octet-stream".

Table 47 – StartFirmwareUpgrade command

StartFirmwareUpgrade	Access class: UNRECOVERABLE
Message name	**Description**
StartFirmwareUpgradeRequest	*This is an empty message*
StartFirmwareUpgrade-Response	*This message contains:* ▪ *A URL to which the firmware file may be uploaded.* ▪ *An optional delay; the client shall wait for this amount of time before initiating the firmware upload.* ▪ *A duration that indicates how long the device expects to be unavailable after the firmware upload is complete.* xs:anyURI **UploadUri [1][1]** xs:duration **UploadDelay [0][1]** xs:duration **ExpectedDownTime [0][1]**
Fault codes	**Description**
	No command-specific faults.

8.4.11 Get system logs

This operation gets a system log from a device as described in Table 48. The device should support system log information retrieval through the GetSystemLog command. The exact format of the system logs is outside the scope of this document.

The system log information shall be transmitted through MTOM [W3C SOAP-MTOM] or as a string.

Table 48 – GetSystemLog command

GetSystemLog	Access class: READ_SYSTEM_SECRET
Message name	**Description**
GetSystemLogRequest	*This message contains the type of system log to retrieve. The types of supported log information is defined in two different types:* • *"System": The system log* • *"Access": The client access log* tt:SystemLogType **LogType** [1][1]
GetSystemLogResponse	*This message contains the requested system log information. The device can choose if it wants to return the system log information as binary data in an attachment or as a common string.* tt:AttachmentData **Binary** [0][1] xs:string **String** [0][1]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:AccesslogUnavailable	*There is no access log information available*
env:Sender ter:InvalidArgs ter:SystemlogUnavailable	*There is no system log information available*

8.4.12 Get support information

This operation gets arbitrary device diagnostics information from a device as described in Table 49. The device may support retrieval of diagnostics information through the GetSystemSupportInformation command. The exact format of the diagnostic information is outside the scope of this document.

The diagnostics information shall be transmitted as an attachment through MTOM [W3C SOAP-MTOM] or as a string.

Table 49 – GetSystemSupportInformation command

GetSystemSupportInformation	Access class: READ_SYSTEM
Message name	**Description**
GetSystemSupport-InformationRequest	*This is an* empty *message.*
GetSystemSupport-Information Response	*The message contains the support information. The device can choose if it wants to return the support information as binary data or as a common string.* tt:AttachmentData **BinaryFormat** [0][1] xs:string **StringFormat** [0][1]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:SupportInformation-Unavailable	*There is no support information available.*

8.4.13 Reboot

This operation reboots a device. Before the device reboots the response message shall be sent. The device shall support reboot through the SystemReboot command as described in Table 50.

Table 50 – SystemReboot command

SystemReboot	Access class: ACTUATE
Message name	**Description**
SystemReboot	*This is an* empty *message.*
SystemRebootResponse	*This message contains a "Message" string allowing the device to report back a message to the client as for example "Rebooting in x seconds."* Xs:string **Message** [1][1]
Fault codes	**Description**
	No command specific faults!

8.4.14 Get scope parameters

This operation requests the scope parameters of a device as described in Table 51. The scope parameters are used in the device discovery to match a probe message, see Clause 7. The scope parameters are of two different types:

- fixed,
- configurable.

Fixed scope parameters cannot be altered through the device management interface but are permanent device characteristics part of the device firmware configurations. The scope type is indicated in the scope list returned in the get scope parameters response. Configurable scope parameters can be set through the set and add scope parameters operations, see 8.4.15 and 8.4.16. The device shall support retrieval of discovery scope parameters through the GetScopes command. As some scope parameters are mandatory, the client always expects a scope list in the response.

Table 51 – GetScopes command

GetScopes	Access class: READ_SYSTEM
Message name	**Description**
GetScopesRequest	This is an *empty* message.
GetScopesResponse	*The scope response message contains a list of URIs defining the device scopes.* tt:Scope: **Scopes** [1][unbounded]
Fault codes	**Description**
env:Receiver ter:Action ter:EmptyScope	*Scope list is empty.*

8.4.15 Set scope parameters

This operation sets the scope parameters of a device as described in Table 52. The scope parameters are used in the device discovery to match a probe message, see Clause 7.

This operation replaces all existing configurable scope parameters (not fixed parameters). If this shall be avoided, one should use the scope add command instead. The device shall support configuration of discovery scope parameters through the SetScopes command.

Table 52 – SetScopes command

SetScopes	Access class: WRITE_SYSTEM
Message name	**Description**
SetScopesRequest	*The set scope contains a list of URIs defining the device scope.* Xs:anyURI: **Scopes** [1][unbounded]
SetScopesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:ScopeOverwrite	*Scope parameter overwrites fixed device scope setting, command rejected.*
env:Receiver ter:Action ter:TooManyScopes	*The requested scope list exceeds the supported number of scopes.*

8.4.16 Add scope parameters

This operation adds new configurable scope parameters to a device as described in Table 53. The scope parameters are used in the device discovery to match a probe message, see Clause 7. The device shall support addition of discovery scope parameters through the AddScopes command.

Table 53 – AddScopes command

AddScopes	Access class: WRITE_SYSTEM
Message name	**Description**
AddScopesRequest	*The add scope contains a list of URIs to be added to the existing configurable scope list.* xs:anyURI:**ScopeItem** [1][unbounded]
AddScopesResponse	*This is an empty message.*
Fault codes	**Description**
env:Receiver ter:Action ter:TooManyScopes	*The requested scope list exceeds the supported number of scopes.*

8.4.17 Remove scope parameters

This operation deletes scope-configurable scope parameters from a device as described in Table 54. The scope parameters are used in the device discovery to match a probe message, see Clause 7. The device shall support deletion of discovery scope parameters through the RemoveScopes command.

Note that the response message will always match the request or an error will be returned. The use of the response is for that reason deprecated.

Table 54 – RemoveScopes command

RemoveScopes	Access class: WRITE_SYSTEM
Message name	**Description**
RemoveScopesRequest	*The remove scope contains a list of URIs that should be removed from the device scope.* xs:anyURI: **ScopeItem** [1][unbounded]
RemoveScopesResponse	*The scope response message contains a list of URIs that has been removed from the device scope.* *xs:anyURI*: ScopeItem [0][unbounded]
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:FixedScope	*Trying to remove fixed scope parameter, command rejected.*
env:Sender ter:InvalidArgVal ter:NoScope	*Trying to remove scope which does not exist.*

8.4.18 Get discovery mode

This operation gets the discovery mode of a device as described in Table 55. See 7.2 for the definition of the different device discovery modes. The device shall support retrieval of the discovery mode setting through the GetDiscoveryMode command.

Table 55 – GetDiscoveryMode command

GetDiscoveryMode	Access class: READ_SYSTEM
Message name	**Description**
GetDiscoveryModeRequest	*This is an empty message.*
GetDiscoveryModeResponse	*This message contains the current discovery mode setting, i.e. discoverable or non-discoverable.* tt:DiscoveryMode: **DiscoveryMode** [1][1]
Fault codes	**Description**
	No command specific faults!

8.4.19 Set discovery mode

This operation sets the discovery mode operation of a device as described in Table 56. See 7.2 for the definition of the different device discovery modes. The device shall support configuration of the discovery mode setting through the SetDiscoveryMode command.

Table 56 – SetDiscoveryMode command

SetDiscoveryMode	Access class: WRITE_SYSTEM
Message name	**Description**
SetDiscoveryModeRequest	*This message contains the requested discovery mode setting, i.e. discoverable or non-discoverable.* tt:DiscoveryMode: **DiscoveryMode** [1][1]
SetDiscoveryMode-Response	*This is an empty message.*
Fault codes	**Description**
	No command specific faults!

8.5 Security

8.5.1 General

Subclause 8.5 contains a set of security management operations. Such operations are sensitive to network attacks and shall be protected using appropriate authorization levels in order not to compromise the device.

8.5.2 Get access policy

Access to different services and sub-sets of services should be subject to access control. Subclause 5.13 gives the prerequisite for end-point authentication. Authorization decisions can then be taken using an access security policy. This document does not mandate any particular policy description format or security policy. It is up to the device manufacturer or system provider to choose a policy and policy description format. However, an access policy (in arbitrary format) can be requested using this command. If the device supports access policy settings, then the device shall support this command as described in Table 57.

Table 57 – GetAccessPolicy command

GetAccessPolicy	Access class: READ_SYSTEM_SECRET
Message name	**Description**
GetAccessPolicyRequest	*This is an empty message.*
GetAccessPolicyResponse	*This message contains the requested policy file.* tt:BinaryData **PolicyFile** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:EmptyPolicy	*The device policy file does not exist or it is empty.*

8.5.3 Set access policy

This command sets the device access security policy (for more details on the access security policy see the Get command, 8.5.2). If the device supports access policy settings, then the device shall support this command as described in Table 58.

Table 58 – SetAccessPolicy command

SetAccessPolicy	Access class: WRITE_SYSTEM
Message name	**Description**
SetAccessPolicyRequest	*This message contains the policy file to set.* tt:BinaryData **PolicyFile** [1][1]
SetAccessPolicyResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:PolicyFormat	*The requested policy cannot be set due to unknown policy format.*

8.5.4 Get users

This operation lists the registered users along with their user levels. The device shall support retrieval of registered device users through the GetUsers command as described in Table 59.

Furthermore a device shall not return the credentials (password) in the reply.

Table 59 – GetUsers command

GetUsers	Access class: READ_SYSTEM_SECRET
Message name	**Description**
GetUsersRequest	*This is an empty message.*
GetUsersResponse	*This message contains the list of users and corresponding user level. Each entry includes:* • *Username* • *User level* *NOTE The password is not included in the response even if it is present in the tt:User type.* tt:User: **User** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.5.5 Create users

This operation creates new device users and corresponding credentials on a device for authentication, see 5.13 for details. The device shall support the creation of device users and their credentials for authentication through the CreateUsers command, as described in Table 60, as long as the number of existing users does not exceed the capability value MaxUsers. Either all users are created successfully or a fault message shall be returned without creating any user.

A device is recommended to support password length of at least 28 bytes.

Table 60 – CreateUsers command

CreateUsers	Access class: WRITE_SYSTEM
Message name	**Description**
CreateUsersRequest	*This message contains a user parameters element for a new user. Each user entry includes:* • *Username* • *Password* • *UserLevel* tt:User: **User** [1][unbounded]
CreateUsersResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:UsernameClash	*Username already exists.*
env:Sender ter:OperationProhibited ter:PasswordTooLong	*The password is too long*
env:Sender ter:OperationProhibited ter:UsernameTooLong	*The username is too long*
env:Sender ter:OperationProhibited ter:Password	*Too weak password.*
env:Receiver ter:Action ter:TooManyUsers	*Maximum number of supported users exceeded.*
env:Sender ter:OperationProhibited ter:AnonymousNotAllowed	*User level anonymous is not allowed.*
env:Sender ter:OperationProhibited ter:UsernameTooShort	*The username is too short*

8.5.6 Delete users

This operation deletes users on a device as described in Table 61. The device shall support deletion of device users and their credentials for authentication through the DeleteUsers command. A device may have one or more fixed users that cannot be deleted to ensure access to the unit. Either all users are deleted successfully or a fault message shall be returned and no users shall be deleted.

Table 61 – DeleteUsers command

DeleteUsers	Access class: WRITE_SYSTEM
Message name	**Description**
DeleteUsersRequest	*This message contains the name of the user or users to be deleted.* xs:string: **Username** [1][unbounded]
DeleteUsersResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UsernameMissing	*Username not recognized.*
env:Sender ter:InvalidArgVal ter:FixedUser	*Username may not be deleted*

8.5.7 Set users settings

This operation updates the settings for one or several users on a device for authentication, see 5.13 for details. The device shall support update of device users and their credentials through the SetUser command as described in Table 62. Either all change requests are processed successfully or a fault message shall be returned and no change requests shall be processed.

In case the optional password value is omitted the device will consider to clear the password. If the device cannot accept the password of zero length, the fault message "ter:PasswordTooWeak" will be returned.

Table 62 – SetUser command

SetUser	Access class: WRITE_SYSTEM
Message name	**Description**
SetUserRequest	*This message contains a list of users and corresponding parameters to be updated.* • *Username* • *Password* • *UserLevel* tt:User: User [1][unbounded]
SetUserResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UsernameMissing	*Username not recognized.*
env:Sender ter:OperationProhibited ter:PasswordTooLong	*The password is too long*
env:Sender ter:OperationProhibited ter:PasswordTooWeak	*Too weak password.*
env:Sender ter:OperationProhibited ter:AnonymousNotAllowed	*User level anonymous is not allowed.*
env:Sender ter:InvalidArgVal ter:FixedUser	*Password or user level may not be changed.*

8.5.8 IEEE 802.1X configuration

8.5.8.1 General

This document defines the following parameters as a set of IEEE 802.1X configuration parameters.

- Configuration token

  This parameter indicates a reference token of IEEE 802.1X configuration parameters and is defined as 'Dot1XConfigurationToken' in the common schema, see Clause B.4. This naming convention of 'Dot1X', which actually represents 'IEEE 802.1X' is used for better readability of the schema element in the generated source code.

- EAP identity

  This parameter indicates the user name of the supplicant which connects to IEEE 802.1X managed network. This is defined as 'Identity' 'Identity' in the common schema, see Clause B.4.

- EAP method

  This parameter indicates the authentication method used. This is defined as 'EAPMethod' in the common schema, see Clause B.4.

- CA certificate ID

  This parameter indicates the ID of the CA certificate used for authentication server verification. This is defined as 'CACertificateID' in the common schema, see Clause B.4.

- Respective configuration parameters for selected EAP method

  Depending on the selected EAP method, some specific parameters are needed as follows:

  – **[EAP-MD5], [EAP-PEAP/MSCHAP-V2], [EAP-TTLS types]**: Identity password so that the authentication server can verify the user (the device) by using the specified password. [EAP-MD5] method is not applicable for the purposes of 802.11 (WPA-Enterprise) usage.

  – **[EAP-TLS]**: Client certificate ID so that the RADIUS server can verify the user (the device) by using the specified certificate.

These IEEE 802.1X parameters will be referred by security configuration as part of a certain network interface configuration. For more details, refer to 8.3.21.

This document assumes that the IEEE 802.1X configuration on device will be done outside the IEEE 802.1X managed network. In case of reconfiguration of the IEEE 802.1X settings, it is also assumed that it will be done outside the 802.1X managed network.

Note that support for IEEE 802.1X is limited to IEEE 802.11 interfaces.

8.5.8.2 Create IEEE 802.1X configuration

This operation creates a new IEEE 802.1X configuration parameter set of the device as described in Table 63. The device shall support this command if support for IEEE 802.1X is signaled via the Security Dot1X capability. If the device receives this request with an already existing configuration token (Dot1XConfigurationToken) specification, the device should respond with a '*ter:ReferenceToken'* error to indicate there is some configuration conflict.

Table 63 – CreateDot1XConfiguration command

CreateDot1XConfiguration	Access class: WRITE_SYSTEM
Message name	**Description**
CreateDot1XConfigurationRequest	*This message contains:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
CreateDot1XConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:EAPMethodNotSupported	*The suggested EAP method is not supported.*
env:Receiver ter:Action ter:MaxDot1X	*Maximum number of IEEE 802.1X configurations reached.*
env:Sender ter:OperationProhibited ter:CertificateID	*Invalid Certificate ID error.*
env:Sender ter:InvalidArgVal ter:ReferenceToken	*Dot1XConfigurationToken already exists.*
env:Sender ter:InvalidArgVal ter:InvalidDot1X	*Invalid IEEE 802.1X configuration.*

8.5.8.3 Set IEEE 802.1X configuration

While the CreateDot1XConfiguration command is trying to create a new configuration parameter set, this operation modifies the existing IEEE 802.1X configuration parameter set of the device as described in Table 64. The device shall support this command if support for IEEE 802.1X is signaled via the Security Dot1X capability.

Table 64 – SetDot1XConfigurationRequest command

SetDot1XConfiguration	Access class: WRITE_SYSTEM
Message name	**Description**
SetDot1XConfigurationRequest	*This message contains:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
SetDot1XConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:EAPMethodNotSupported	*The suggested EAP method is not supported.*
env:Sender ter:OperationProhibited ter:CertificateID	*Invalid certificate ID error.*
env:Sender ter:OperationProhibited ter:ReferenceToken	*Invalid Dot1XConfigurationToken error*
env:Sender ter:InvalidArgVal ter:InvalidDot1X	*Invalid IEEE 802.1X configuration.*

8.5.8.4 Get IEEE 802.1X configuration

This operation gets one IEEE 802.1X configuration parameter set from the device by specifying the configuration token (Dot1XConfigurationToken).

The device shall support this command if support for IEEE 802.1X is signaled via the Security Dot1X capability as described in Table 65.

Regardless of whether the 802.1X method in the retrieved configuration has a password or not, the device shall not include the Password element in the response.

Table 65 – GetDot1XConfiguration command

GetDot1XConfiguration	Access class: READ_SYSTEM
Message name	**Description**
GetDot1XConfigurationRequest	*This is message contains:* tt:ReferenceToken **Dot1XConfigurationToken**[1][1]
GetDot1XConfigurationResponse	*This message contains:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:ReferenceToken	*Invalid Dot1XConfigurationToken error*

8.5.8.5 Get IEEE 802.1X configurations

This operation gets all the existing IEEE 802.1X configuration parameter sets from the device. The device shall respond with all the IEEE 802.1X configurations so that the client can get to know how many IEEE 802.1X configurations are existing and how they are configured.

The device shall support this command if support for IEEE 802.1X is signaled via the Security Dot1X capability as described in Table 66.

Regardless of whether the 802.1X method in the retrieved configuration has a password or not, the device shall not include the Password element in the response.

Table 66 – GetDot1XConfigurations command

GetDot1XConfigurations	Access class: READ_SYSTEM
Message name	**Description**
GetDot1XConfigurationsRequest	*This is an empty message.*
GetDot1XConfigurationsResponse	*This message contains:* tt:Dot1XConfiguration **Dot1XConfiguration**[0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.5.8.6 Delete IEEE 802.1X configuration

This operation deletes an IEEE 802.1X configuration parameter set from the device as described in Table 67. Which configuration should be deleted is specified by the 'Dot1XConfigurationToken' in the request. The device shall support this command if support for IEEE 802.1X is signaled via the Security Dot1X capability.

Table 67 – DeleteDot1XConfigurations command

DeleteDot1XConfigurations	Access class: WRITE_SYSTEM
Message name	**Description**
DeleteDot1XConfigurationRequest	*This message contains:* tt:ReferenceToken **Dot1XConfigurationToken**[1][1]
DeleteDot1XConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:ReferenceToken	*Invalid Dot1XConfigurationToken error*
env:Receiver ter:OperationProhibited ter:ReferenceToken	*Cannot delete specified IEEE 802.1X configuration.*

8.5.9 Create self-signed certificate

This operation generates a private/public key pair and can also create a self-signed device certificate as a result of key pair generation as described in Table 68. The certificate is created using a suitable onboard key pair generation mechanism.

If a device supports onboard key pair generation, the device that supports TLS shall support this certificate creation command. Likewise, if a device supports onboard key pair generation, the device that signals support for IEEE 802.1X via the Security Dot1X capability shall support this command for the purpose of key pair generation. Certificates and key pairs are identified using certificate IDs. These IDs are either chosen by the certificate generation requester or by the device (in case no ID value is given).

Table 68 – CreateCertificate command

CreateCertificate	Access class: WRITE_SYSTEM
Message name	**Description**
CreateCertificateRequest	*This message contains (if applicable) requested Certificate ID and additional other requested parameters: subject, valid not before and valid not after.* xs:token **CertificateID** [0][1] xs:string **Subject** [0][1] xs:dateTime **ValidNotBefore** [0][1] xs:dateTime **ValidNotAfter** [0][1]
CreateCertificateResponse	*This message contains the generated self-signed certificate.* tt:Certificate **NvtCertificate** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:KeyGeneration	*The private/public key generation failed.*
env:Sender ter:InvalidArgVal ter:CertificateID	*CertificateID already exists.*
env:Sender ter:InvalidArgVal ter:InvalidDateTime	*Specified ValidNotBefore or ValidNotAfter parameter is not valid.*

8.5.10 Get certificates

This operation gets all device server certificates (including self-signed) for the purpose of TLS authentication and all device client certificates for the purpose of IEEE 802.1X authentication. This command lists only the TLS server certificates and IEEE 802.1X client certificates for the device (neither trusted CA certificates nor trusted root certificates). The certificates are returned as binary data. A device that supports TLS shall support this command as described in Table 69 and the certificates shall be encoded using ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1] encoding rules.

Table 69 – GetCertificates command

GetCertificates	Access class: READ_SYSTEM
Message name	**Description**
GetCertificatesRequest	*This is an empty message.*
GetCertificatesResponse	*This message contains a list of the device certificates.* tt:Certificate **NvtCertificate** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.5.11 Get CA certificates

CA certificates will be loaded into a device and be used for the sake of following two cases. The one is for the purpose of TLS client authentication in the TLS server function. The other one is for the purpose of Authentication Server authentication in the IEEE 802.1X function.

This operation gets all CA certificates loaded into a device. A device that supports either TLS client authentication or IEEE 802.1X shall support this command as described in Table 70 and the returned certificates shall be encoded using ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1] encoding rules.

Table 70 – GetCACertificates command

GetCACertificates	Access class: READ_SYSTEM
Message name	**Description**
GetCACertificatesRequest	*This is an empty message.*
GetCACertificatesResponse	*This message contains a list of the CA certificates.* tt:Certificate **CACertificate** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.5.12 Get certificate status

This operation is specific to TLS functionality. This operation gets the status (enabled/disabled) of the device TLS server certificates. A device that supports TLS shall support this command as described in Table 71.

Table 71 – GetCertificatesStatus command

GetCertificatesStatus	Access class: READ_SYSTEM
Message name	**Description**
GetCertificatesStatusRequest	*This is an empty message.*
GetCertificatesStatusResponse	*This message contains a list of the device server certificates referenced by ID and their status. The status is defined as a Boolean value (true = enabled, false = disabled).* tt:CertificateStatus **CertificateStatus** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.5.13 Set certificate status

This operation is specific to TLS functionality. This operation sets the status (enable/disable) of the device TLS server certificates. A device that supports TLS shall support this command as described in Table 72. Typically only one device server certificate is allowed to be enabled at a time.

Table 72 – SetCertificatesStatus command

SetCertificatesStatus	Access class: WRITE_SYSTEM
Message name	**Description**
SetCertificatesStatus-Request	*This message contains a list of device server certificates referenced by ID and the requested certificate status, i.e., enabled or disabled.* tt:CertificateStatus **CertificateStatus** [0][unbounded]
SetCertificatesStatus-Response	*This is an empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Unknown certificate reference.*

8.5.14 Get certificate request

This operation requests a PKCS #10 certificate signature request from the device. The returned information field shall be either formatted exactly as specified in [RFC 2986] or PEM

encoded [RFC 2986] format. In order for this command to work, the device shall already have a private/public key pair. This key pair should be referred by CertificateID as specified in the input parameter description. This CertificateID refers to the key pair generated using CreateCertificate command defined in 8.5.9.

A device that support onboard key pair generation and that supports either TLS or IEEE 802.1X using the client certificate shall support this command as described in Table 73.

Table 73 – GetPkcs10Request command

GetPkcs10Request	Access class: READ_SYSTEM
Message name	**Description**
GetPkcs10RequestRequest	*This message contains a reference to the certificate (key pair) and optional certificate parameters for the certificate request. These attributes need to be encoded as DER ASN.1 objects.* xs:token **CertificateID** [1][1] xs:string **Subject** [0][1] xs:BinaryData **Attributes** [0][1]
GetPkcs10RequestResponse	*This message contains the PKCS#10 request data structure.* tt:BinaryData **Pkcs10Request** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Invalid CertificateID*
env:Receiver ter:Action ter:Signature	*PKCS#10 signature creation failed.*

8.5.15 Get client certificate status

This operation is specific to TLS functionality. This operation gets the status (enabled/disabled) of the device TLS client authentication. A device that supports TLS shall support this command as described in Table 74.

Table 74 – GetClientCertificateMode command

GetClientCertificateMode	Access class: READ_SYSTEM
Message name	**Description**
GetClientCertificateMode-Request	*This is an empty message.*
GetClientCertificateMode-Response	*This message contains the device client authentication status, i.e., enabled or disabled.* xs:boolean **Enabled** [1][1]
Fault codes	**Description**
	No command specific faults!

8.5.16 Set client certificate status

This operation is specific to TLS functionality. This operation sets the status (enabled/disabled) of the device TLS client authentication. A device that supports TLS shall support this command as described in Table 75.

Table 75 – SetClientCertificateMode command

SetClientCertificateMode	Access class: WRITE_SYSTEM
Message name	**Description**
SetClientCertificateMode-Request	*This message contains the requested device client authentication status, i.e., enabled or disabled.* xs:boolean **Enabled** [1][1]
SetClientCertificateMode-Response	*This is an empty message*
Fault codes	**Description**
env:Receiver ter:InvalidArgVal ter:ClientAuth	*Trying to enable client authentication, but client authentication is not supported or not configured.*

8.5.17 Load device certificate

TLS server certificate(s) or IEEE 802.1X client certificate(s) created using the PKCS#10 certificate request command can be loaded into the device using this command (see 8.5.14). The certificate ID in the request shall be present. The device may sort the received certificate(s) based on the public key and subject information in the certificate(s).

The certificate ID in the request will be the ID value the client wishes to have. The device is supposed to scan the generated key pairs present in the device to identify which is the correspondent key pair with the loaded certificate and then make the link between the certificate and the key pair.

A device that supports onboard key pair generation that support either TLS or IEEE 802.1X shall support this command as described in Table 76.

The certificates shall be encoded using ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1] encoding rules.

This command is applicable to any device type, although the parameter name is called for historical reasons NVTCertificate.

Table 76 – LoadCertificates command

LoadCertificates	Access class: WRITE_SYSTEM
Message name	**Description**
LoadCertificatesRequest	*This message contains a list of the device certificates to upload.* tt:Certificate **NVTCertificate** [1][unbounded]
LoadCertificatesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateFormat	*Bad certificate format or the format is not supported by the device.*
env:Sender ter:InvalidArgVal ter:CertificateID	*Certificate ID already exists.*
env:Sender ter:InvalidArgVal ter:InvalidCertificate	*Invalid Certificate.*

8.5.18 Load device certificates in conjunction with its private key

There might be some cases where a certificate authority or some other equivalent organization creates a certificate without having a PKCS#10 certificate signing request. In such cases, the certificate will be bundled in conjunction with its private key. This command will be used in such a case scenario. The certificate ID in the request is optionally set to the

ID value the client wishes to have. If the certificate ID is not specified in the request, the device can choose the ID accordingly.

This operation imports a private/public key pair into the device.

The certificates shall be encoded using ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1] encoding rules.

A device that does not support onboard key pair generation and supports either TLS or IEEE 802.1X using the client certificate should support this command. A device that supports onboard key pair generation may support this command as described in Table 77. The security policy of a device that supports this operation should make sure that the private key is sufficiently protected.

Table 77 – LoadCertificateWithPrivateKey command

LoadCertificateWithPrivateKey	Access class: WRITE_SYSTEM
Message name	**Description**
LoadCertificateWithPrivateKeyRequest	*This message contains a private/public key pair to import.* tt:CertificateWithPrivateKey **CertificateWithPrivateKey**[1][unbounded]
LoadCertificateWithPrivateKeyResponse	*This is an empty message.*
Fault codes	**Description**
env: Sender ter:InvalidArgVal ter:CertificateFormat	*Bad certificate format or the format is not supported by the device.*
env:Sender ter:InvalidArgVal ter:CertificateID	*CertificateID already exists.*
env: Sender ter:InvalidArgVal ter: KeysNotMatching	The public and private key are not matching.

8.5.19 Get certificate information request

This operation requests the information of a certificate specified by the certificate ID. The device should respond with its “Issuer DN”, “Subject DN”, “Key usage”, "Extended key usage”, “Key Length”, “Version”, “Serial Number”, “Signature Algorithm” and “Validity” data as the information of the certificate, as long as the device can retrieve such information from the specified certificate. IssuerDN and SubjectDN shall be encoded using the rules in [RFC 4514].

A device that supports either TLS or IEEE 802.1X should support this command as described in Table 78.

Table 78 – GetCertificateInformation command

GetCertificateInformation	Access class: READ_SYSTEM
Message name	**Description**
GetCertificateInformationRequest	*This message contains:* *CertificateID: The token of the certificate.* xs: *token CertificateID* [1][1]
GetCertificateInformationResponse	*This message contains:* tt:CertificateInformation **CertificateInformation**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Invalid Certificate ID*

8.5.20 Load CA certificates

This command is used when it is necessary to load trusted CA certificates or trusted root certificates for the purpose of verification of its counterpart i.e. client certificate verification in TLS function or server certificate verification in IEEE 802.1X function.

A device that supports either TLS or IEEE 802.1X shall support this command as described in Table 79. The device shall support the DER format; other formats may be supported by the device.The device may sort the received certificate(s) based on the public key and subject information in the certificate(s). Either all CA certificates are loaded successfully or a fault message shall be returned without loading any CA certificate.

Table 79 – LoadCACertificates command

LoadCACertificates	Access class: WRITE_SYSTEM
Message name	**Description**
LoadCACertificatesRequest	*This message contains a list of the device CA certificates to upload.* tt:Certificate **CACertificate** [1][unbounded]
LoadCACertificatesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateFormat	*Bad certificate format or the format is not supported by the device.*
env:Sender ter:InvalidArgVal ter:CACertificateID	*CA Certificate ID already exists.*
env:Receiver ter:OperationProhibited ter:MaxCertificates	*Maximum number of certificates already loaded.*

8.5.21 Delete certificate

This operation deletes a certificate or multiple certificates. The device may also delete a private/public key pair which is coupled with the certificate to be deleted. The device that support either TLS or IEEE 802.1X shall support the deletion of a certificate or multiple certificates through this command as described in Table 80. Either all certificates are deleted successfully or a fault message shall be returned without deleting any certificate.

Table 80 – DeleteCertificates command

DeleteCertificates	Access class: WRITE_SYSTEM
Message name	**Description**
DeleteCertificatesRequest	*This message deletes certificates identified with the CertificateID parameter.* xs:token **CertificateID**[1][unbounded]
DeleteCertificatesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Unknown certificate reference.*
env:Receiver ter:OperationProhibited ter:CertificateID	*Cannot delete specified certificates.*

8.5.22 Get remote user

This operation returns the configured remote user (if any) as described in Table 81. A device that signals support for remote user handling via the security capability RemoteUserHandling shall support this operation. The user is only valid as a HTTP/RTSP user.

The UseDerivedPassword is included for backward compatibility reason and should always be false.

Table 81 – GetRemoteUser command

GetRemoteUser	Access class: READ_SYSTEM
Message name	**Description**
GetRemoteUserRequest	*This is an empty message.*
GetRemoteUserResponse	*This message contains the configured remote user (if any). The value returned are:* • *xs:string* ***Username*** *[1][1]* • *xs:boolean* ***UseDerivedPassword*** *[1][1]* *A device shall never return the Password field in* ***RemoteUser****.* tt:RemoteUser: **RemoteUser** [0][1]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:NotRemoteUser	*Remote User handling is not supported*

8.5.23 Set remote user

This operation sets the remote user as described in Table 82. A device that signals support for remote user handling via the security capability RemoteUserHandling shall support this operation. The user is only valid as a HTTP/RTSP user.

To remove the remote user SetRemoteUser should be called without the RemoteUser parameter.

The UseDerivedPassword is included for backward compatibility reason and should always be false.

Table 82 – SetRemoteUser command

SetRemoteUser	Access class: WRITE_SYSTEM
Message name	**Description**
SetRemoteUserRequest	*This message contains the remote user. The values that can be set are:* • *xs:string* ***Username*** *[1][1]* • *xs:string* ***Password*** *[0][1]* • *xs:boolean* ***UseDerivedPassword*** *[1][1]* tt:RemoteUser: **RemoteUser** [0][1]
SetRemoteUserResponse	*This is an empty message*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:NotRemoteUser	*Remote User handling not supported*

8.5.24 Get endpoint reference

A client can ask for the device service endpoint reference address property that can be used to derive the password equivalent for remote user operation. The device should support the GetEndpointReference command as described in Table 83 returning the address property of the device service endpoint reference.

Table 83 – GetEndpointReference command

GetEndpointReference	Access class: PRE_AUTH
Message name	**Description**
GetEndpointReferenceRequest	*This is an empty message.*
GetEndpointReference-Response	*The requested URL.* xs:string **GUID** [1][1]
Fault codes	**Description**
	No command specific faults!

8.6 Auxiliary operation

Subclause 8.6 describes operations to manage auxiliary commands supported by a device as described in Table 84, such as controlling an Infrared (IR) lamp, a heater or a wiper or a thermometer that is connected to the device.

The commands supported by the device are reported in the Misc – AuxiliaryCommands attribute returned by the GetServiceCapabilites commands, see 8.2.2.3. The command transmitted by using this command should match one of the commands supported by the device. If, for example, the GetServiceCapabilites command response lists only the irlampon command, then the SendAuxiliaryCommand argument will be irlampon, which may indicate turning the connected IR lamp on.

Although the name of the auxiliary commands can be freely defined, commands starting with the prefix tt: are reserved to define frequently used commands and these reserved commands shall all share the "tt:command|parameter" syntax.

- tt:Wiper|On – Request to start the wiper.
- tt:Wiper|Off – Request to stop the wiper.
- tt:Washer|On – Request to start the washer.
- tt:Washer|Off – Request to stop the washer.
- tt:WashingProcedure|On – Request to start the washing procedure.
- tt:WashingProcedure|Off – Request to stop the washing procedure.

A device that indicates auxiliary service capability shall support this command.

Table 84 – SendAuxiliary command

SendAuxiliaryCommand	Access class: ACTUATE
Message name	**Description**
SendAuxiliaryCommandRequest	*This message contains the auxiliary command.* tt:AuxiliaryData **AuxiliaryCommand**[1][1]
SendAuxiliaryCommandResponse	*The response contains the auxiliary response.* tt:AuxiliaryData **AuxiliaryCommandResponse**[0][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:AuxiliaryDataNotSupported	*The requested AuxiliaryCommand is not supported.*

8.7 Monitoring events

8.7.1 Processor usage

If a device supports monitoring of the processing unit usage, it should provide the processing unit usage monitoring event to inform a client about its current processing unit usage in percent:

```
Topic: tns1:Monitoring/ProcessorUsage
<tt:MessageDescription IsProperty="true">
  <tt:Source>
  <tt:SimpleItemDescription Name=”Token” Type=”tt:ReferenceToken”/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="Value" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.2 Link status

If a device supports monitoring of the link status, it should provide the link status monitoring event to inform a client about its current link status.

```
Topic: tns1:Monitoring/LinkStatus
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Link"
Type="tt:NetworkInterfaceConnectionSetting"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.3 Upload status

If a device supports monitoring of its upload firmware (upload of firmware or system information) it should provide the status in percent of an ongoing update using the UploadStatus event.

```
Topic: tns1:Monitoring/UploadStatus
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.4 Operating time

The set of events defined in this subclause relates to operating time. A device supporting operation time events should provide the following events.

The following event should be generated with a true value when the operating time limit is reached.

```
Topic: tns1:Monitoring/OperatingTime/DefinedLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with a true value when the devices MTBF default limit has been reached.

```
Topic:
tns1:Monitoring/OperatingTime/MeanTimeBetweenFailuresDefaultLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with a true value when the devices MTBF operation limit has been reached.

```
Topic:
tns1:Monitoring/OperatingTime/MeanTimeBetweenFailuresOperationLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies when the device has been reset to factory settings the last time.

```
Topic: tns1:Monitoring/OperatingTime/LastReset
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies when the device has been rebooted the last time.

```
Topic: tns1:Monitoring/OperatingTime/LastReboot
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies when the device clock has been synchronized the last time either via an NTP message or via a SetSystemDateAndTime call.

```
Topic: tns1:Monitoring/OperatingTime/LastClockSynchronization
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies the last maintenance activity on the device.

```
Topic: tns1:Monitoring/Maintenance/Last
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies the next maintenance activity on the device.

```
Topic: tns1:Monitoring/Maintenance/NextScheduled
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies when the last backup of the device configuration has been retrieved.

```
Topic: tns1:Monitoring/Backup/Last
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with a true value when the area of operation the device is certified for is not adhered to due to outside influences.

```
Topic: tns1:Monitoring/AreaOfOperation/OutsideCertifiedArea
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with a true value when the area of operation the device is configured for is not adhered to due to outside influences.

```
 Topic: tns1:Monitoring/AreaOfOperation/OutsideConfiguredArea
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.5 Environmental conditions

If measurements of environmental conditions are supported a device should provide the following events.

The following event specifies the relative humidity in percent.

```
Topic: tns1:Monitoring/EnvironmentalConditions/RelativeHumidity
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event specifies the operating temperature of the device in degree Celsius.

```
Topic: tns1:Monitoring/EnvironmentalConditions/Temperature
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.6 Battery capacity

If measurements of the battery level are supported, a device should provide the data using the BatteryCapacity event.

```
Topic: tns1:Monitoring/BatteryCapacity
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="PercentageRemainingCapacity"
                              Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.7 Device management

The following topics signal important device status information:

```
tns1:Device/OperationMode/ShutdownInitiated
tns1:Device/OperationMode/UploadInitiated
tns1:Device/HardwareFailure/FanFailure
tns1:Device/HardwareFailure/PowerSupplyFailure
tns1:Device/HardwareFailure/StorageFailure
tns1:Device/HardwareFailure/TemperatureCritical
```

8.8 Service specific fault codes

Table 85 lists the device service-specific fault codes. In addition, each command can also generate a generic fault, see Table 5.

The specific faults are defined as subcode of a generic fault, see 5.12.3.2. The parent generic subcode is the subcode at the top of each row below and the specific fault subcode is at the bottom of the cell.

Table 85 – Device service specific fault codes

Fault code	Parent subcode Subcode	Fault reason	Description
env:Receiver	ter:Action ter:EmptyPolicy	The policy is empty	The device policy file does not exist or it is empty.
env:Receiver	ter:Action ter:EmptyScope	The scope list is empty	Scope list is empty.
env:Receiver	ter:Action ter:FirmwareUpgradeFailed	Upgrade failed	The firmware upgrade failed.
env:Receiver	ter:Action ter:KeyGeneration	Generating a key failed	The private/public key generation failed.
env:Receiver	ter:Action ter:Signature	Creating a signature failed	PKCS#10 signature creation failed.
env:Receiver	ter:InvalidArgVal ter:ClientAuth	Client authentication not supported	Trying to enable client authentication, but client authentication is not supported or not configured
env:Receiver	ter:Action ter:TooManyUsers	Too many users	Maximum number of supported users exceeded.
env:Receiver	ter:Action ter:TooManyScopes	Too large scope list	The scope list exceeds the supported number of scopes.
env:Receiver	ter:ActionNotSupported ter:NoSuchService	The service is not supported	The requested WSDL service category is not supported by the device.
env:Sender	ter: InvalidArgs ter:AccesslogUnavailable	No access log available	There is no access log information available.
env:Sender	ter:InvalidArgVal ter:CertificateFormat	Invalid format	Bad certificate format or the format is not supported by the device.

Fault code	Parent subcode Subcode	Fault reason	Description
env:Sender	ter:InvalidArgVal ter:CertificateID	Invalid certificate ID	Unknown certificate reference or the certificate ID already exists.
env:Sender	ter:InvalidArgVal ter:CACertificateID	Invalid CA certificate ID	Unknown CA certificate reference or the CA certificate ID already exists.
env:Sender	ter:InvalidArgVal ter:InvalidBackupFile	Invalid file	The backup file(s) are invalid.
env:Sender	ter:InvalidArgVal ter:InvalidDateTime	Invalid date and time.	An invalid date or time was specified.
env:Sender	ter:InvalidArgVal ter:NtpServerUndefined	NTP server undefined.	Cannot switch DateTimeType to NTP because no NTP server is defined.
env:Sender	ter:InvalidArgVal ter:InvalidDnsName	Invalid name	The suggested NTP server name is invalid.
env:Sender	ter:InvalidArgVal ter:TimeSyncedToNtp	Time synced to NTP	Current DateTimeType requires an NTP server.
env:Sender	ter: InvalidArgs ter:InvalidFirmware	Invalid firmware	The firmware was invalid i.e. not supported by this device.
env:Sender	ter:InvalidArgVal ter:InvalidGatewayAddress	Invalid address	The supplied gateway address was invalid.
env:Sender	ter:InvalidArgVal ter:InvalidHostname	Invalid name	The requested hostname cannot be accepted by the device.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceSpeed	Invalid speed	The suggested speed is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceType	Invalid type	The suggested network interface type is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidIPv4Address	Invalid address	The suggested IPv4 address is invalid.
env:Sender	ter:InvalidArgVal ter:NoIPv4Address	Address does not exist	The IPv4 address to be removed does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidIPv6Address	Invalid address	The suggested IPv6 address is invalid.
env:Sender	ter:InvalidArgVal ter:NoIPv6Address	Address does not exist	The IPv6 address to be removed does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidMtuValue	Invalid data	The MTU value is invalid.
env:Sender	ter:InvalidArgVal ter:InvalidNetworkInterface	Invalid token	The supplied network interface token does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidTimeZone	Invalid data	An invalid time zone was specified.
env:Sender	ter:InvalidArgVal ter:IPFilterListIsFull	The list is full	It is not possible to add more IP filters since the IP filter list is full.
env:Sender	ter:InvalidArgs ter:PolicyFormat	Invalid format	The requested policy cannot be set due to unknown policy format.

Fault code	Parent subcode Subcode	Fault reason	Description
env:Sender	ter:InvalidArgVal ter:ServiceNotSupported	The service is not supported	The supplied network service is not supported.
env:Sender	ter:InvalidArgVal ter:SupportInformationUnavailable	No support information available	There is no support information available.
env:Sender	ter: InvalidArgs ter:SystemlogUnavailable	No system log available	There is no system log information available.
env:Sender	ter:InvalidArgVal ter:UsernameMissing	Username not recognized	Username not recognized.
env:Sender	ter:OperationProhibited ter:FixedScope	Trying to delete fixed scope parameter	Trying to delete fixed scope parameter, command rejected.
env:Sender	ter:InvalidArgVal ter:NoScope	Scope does not exist	Trying to Remove scope which does not exist.
env:Sender	ter:OperationProhibited ter:Password	Too weak password	Too weak password.
env:Sender	ter:OperationProhibited ter:PasswordTooLong	Too long password	The password is too long.
env:Sender	ter:OperationProhibited ter:UsernameTooShort	Too short password	The password is too short.
env:Sender	ter:OperationProhibited ter:ScopeOverwrite	Trying overwriting permanent device scope setting	Scope parameter overwrites permanent device scope setting, command rejected.
env:Sender	ter:OperationProhibited ter:UsernameClash	Username already exists	Username already exists.
env:Sender	ter:OperationProhibited ter:UsernameTooLong	Too long username	The username is too long.
env:Receiver	ter:ActionNotSupported ter:InvalidDot11	Not supported	IEEE 802.11 configuration is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidSecurityMode	Not Supported	The selected security mode is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidStationMode	Not Supported	The selected station mode is not supported.
env:Sender	ter:InvalidArgVal ter:MissingDot11	IEEE 802.11 value missing	IEEE 802.11 value is missing in the security configuration.
env:Sender	ter:InvalidArgVal ter:MissingPSK	PSK value missing	PSK value is missing in security configuration.
env:Sender	ter:InvalidArgVal ter:MissingDot1X	IEEE 802.1X value is missing	IEEE 802.1X value in security configuration is missing or non existing.
env:Sender	ter:InvalidArgVal ter:IncompatibleDot1X	IEEE 802.!X value is incompatible	IEEE 802.1X value in security configuration is incompatible with the network interface.
env:Sender	ter:InvalidArgVal ter:NotDot11	Not IEEE 802.11	The interface is not an IEEE 802.11 interface.
env:Sender	ter:InvalidArgVal ter:InvalidDot1X	Invalid IEEE 802.1X configuration	Specified IEEE 802.1X configuration is not valid.

Fault code	Parent subcode Subcode	Fault reason	Description
env:Receiver	ter:Action ter:NotConnectedDot11	IEEE 802.11 not connected	IEEE 802.11 network is not connected.
env:Receiver	ter:ActionNotSupported ter:NotScanAvailable	ScanAvailableIEEE802.11Networks is not supported.	ScanAvailableIEEE802.11Networks is not supported.
env:Receiver	ter:ActionNotSupported ter:NotRemoteUser	Remote User handling is not supported.	Remote User handling is not supported.
env:Receiver	ter:ActionNotSupported ter:EAPMethodNotSupported	The suggested EAP method is not supported.	The suggested EAP method is not supported.
env:Receiver	ter:Action ter:MaxDot1X	Maximum number of IEEE 802.1X configurations reached.	Device reached maximum number of IEEE 802.1X configurations.
env:Receiver	ter:OperationProhibited ter:ReferenceToken	Cannot delete specified IEEE 802.1X configuration.	It is not possible to delete specified IEEE 802.1X configuration.
env:Receiver	ter:OperationProhibited ter:CertificateID	Cannot delete specified certificate(s).	It is not possible to delete specified certificate(s).
env:Sender	ter:OperationProhibited ter:ReferenceToken	Invalid Dot1XConfigurationToken error.	Specified IEEE 802.1X configuration token is invalid.
env:Sender	ter:OperationProhibited ter:CertificateID	Invalid certificate ID error.	Specified certificate ID is invalid.
env:Sender	ter:InvalidArgVal ter:ReferenceToken	Dot1XConfigurationToken already exists.	Specified Dot1XConfigurationToken already exists in the device.
env:Sender	ter:InvalidArgVal ter:InvalidCertificate	Invalid certificate.	Specified certificate is invalid.
env:Receiver	ter:OperationProhibited ter:MaxCertificates	Maximum number of certificates already loaded.	Device reached maximum number of loaded certificates.
env:Sender	ter:OperationProhibited ter:PasswordTooWeak	Too weak password	Too weak password
env:Sender	ter:InvalidArgVal ter:AuxiliaryDataNotSupported	The requested AuxiliaryCommand is not supported.	The requested AuxiliaryCommand is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidTimeOutValue	Invalid Timeout value specified.	Specified TimeOut value is invalid.
env:Receiver	ter:OperationProhibited ter:InvalidMode	Device is not ready to operate in command mode.	Device is not ready to operate in command mode.
env:Sender	ter:InvalidArgVal ter:FixedUser	Removing fixed user	Client trying to remove fixed user.
env:Sender	ter:OperationProhibited ter:AnonymousNotAllowed	User level anonymous is not allowed.	User level anonymous is not allowed.
env:Sender	ter:InvalidArgVal ter:FixedUser	Username or user level may not be changed.	Username or user level toward the specified user(s) may not be changed.
env:Sender	ter:InvalidArgVal ter:KeysNotMatching	Keys not matching	The public and private key is not matching.
env:Sender	ter:InvalidArgVal	Port in use	The selected port is already in

Fault code	Parent subcode Subcode	Fault reason	Description
	ter:PortAlreadyInUse		use.
env:Receiver	ter:ActionNotSupported	Enabling TLS failed	The device doesn't support TLS or TLS is not configured appropriately.
	ter:EnablingTlsFailed		
env:Receiver	ter:ActionNotSupported	Enabling DHCPv6 failed	The requested stateful DHCPv6 mode is not supported.
	ter:InvalidDHCPv6		

9 Device I/O

9.1 General

This service offers commands to retrieve and configure the physical inputs and outputs of a device.

Commands to request the available in- and outputs are defined as well as commands to request the available relays. This service also offers functions to request and change the configuration of these entities.

A device that has inputs or outputs shall support this service as described in Clause B.2.

9.2 Relay outputs

9.2.1 Overview

The Input/Output (I/O) commands are used to control the state or observe the status of the I/O ports. If the device has I/O ports, then it shall support the I/O commands.

9.2.2 Get relay outputs

This operation gets a list of all available relay outputs and their settings as described in Table 86.

Table 86 – GetRelayOutputs command

GetRelayOutputs	Access class: READ_MEDIA
Message name	**Description**
GetRelayOutputsRequest	*This is an empty message.*
GetRelayOutputsResponse	*This message contains an array of relay outputs.* tt:RelayOutput **RelayOutputs** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

9.2.3 Get relay output options

Requests the available settings and ranges for one or all relay outputs as described in Table 87. The method shall return the information for exactly one output when a RelayOutputToken is provided as request parameter. Otherwise the method shall return the information for all relay outputs.

Two examples:

1) Device supports PT1S to PT120S:

```
<tmd:RelayOutputOptions token='44'>
```

```
    <tmd:Mode>Monostable</tmd:Mode>
    <tmd:DelayTimes>1 120</tmd:DelayTimes>
  </tmd:RelayOutputOptions>
```

2) Device supports values PT0.5S, PT1S, PT2s and PT1M:

```
  <tmd:RelayOutputOptions token='123'>
    <tmd:Mode>Monostable</tmd:Mode>
    <tmd:DelayTimes Discrete='True'>0.5 1 2 60</tmd:DelayTimes>
  </tmd:RelayOutputOptions>
```

Table 87 – GetRelayOutputOptions command

GetRelayOutputOptions	Access class: PRE_AUTH
Message name	**Description**
GetRelayOutputOptionsRequest	*This message contains:* • *"RelayOutputToken": Optional token reference to the requested relay output* tt:ReferenceToken **RelayOutputToken** [0][1]
GetRelayOutputOptionsResponse	*This message contains an array of relay output options.* tmd:RelayOutputOptions **RelayOutputOptions**[0][unbounded]
Fault codes	**Description**
	No command specific faults!

9.2.4 Set relay output settings

This operation sets the settings of a relay output as described in Table 88.

The relay can work in two relay modes:

- Bistable – After setting the state, the relay remains in this state.
- Monostable – After setting the state, the relay returns to its idle state after the specified time.

The physical idle state of a relay output can be configured by setting the IdleState to 'open' or 'closed' (inversion of the relay behaviour).

Idle State 'open' means that the relay is open when the relay state is set to 'inactive' through the trigger command (see 9.2.5) and closed when the state is set to 'active' through the same command.

Idle State 'closed' means that the relay is closed when the relay state is set to 'inactive' through the trigger command (see 9.2.5) and open when the state is set to 'active' through the same command.

The duration parameter of the properties field "DelayTime" describes the time after which the relay returns to its idle state if it is in monostable mode. If the relay is set to bistable mode the value of the parameter shall be ignored.

Table 88 – SetRelayOutputSettings command.

SetRelayOutputSettings	Access class: ACTUATE
Message name	**Description**
SetRelayOutputSettingsRequest	*This message contains:* • *"RelayOutputToken": Token reference to the requested relay output.* • *"RelayOutputSettings": The settings of the relay* . tt:ReferenceToken **RelayOutputToken**[1][1] tt:RelayOutputSettings **RelayOutputSettings**[1][1]
SetRelayOutputSettings-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:RelayToken	*Unknown relay token reference.*
Env:Sender ter:InvalidArgVal ter:ModeError	*Monostable delay time not valid*

9.2.5 Trigger relay output

This operation triggers a relay output as described in Table 89.

NOTE There is no GetRelayState command; the current logical state of the relay output is transmitted via notification and their properties.

Table 89 – SetRelayOutputState command

SetRelayOutputState	Access class: ACTUATE
Message name	**Description**
SetRelayOutputStateRequest	*This message contains:* • *RelayOutputToken": Token reference to the requested relay output.* • *"LogicalState": Trigger request, i.e., active or inactive.* tt:ReferenceToken **RelayOutputToken** [1][1] tt:RelayLogicalState **LogicalState** [1][1]
SetRelayOutputStateResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:RelayToken	*Unknown relay token reference.*

9.3 Digital inputs

9.3.1 Overview

The DigitalInput type represents the integrated physical digital inputs of a device which monitor the state of external items that can be toggled between an open and closed circuit. Examples are doorbell switches, movement detectors and door status monitor switches.

9.3.2 GetDigitalInputs

This command lists all available digital inputs of a device. A device that signals support for digital inputs via its capabilities shall support listing of available inputs through the GetDigitalInputs command as described in Table 90.

Table 90 – GetDigitalInputs command

GetDigitalInputs	Access class: READ_MEDIA
Message name	**Description**
GetDigitalInputsRequest	*This is an empty message.*
GetDigitalInputsResponse	*Contains a list of structures describing all available digital inputs of the device. If a device has no digital inputs an empty list is returned.* tt:DigitalInput **DigitalInputs** [0][unbounded]
Fault codes	**Description**
No specific fault codes.	

9.4 SerialPorts

9.4.1 Overview

The SerialPort type represents the physical serial port on the device and allows serial data to be read and written.

9.4.2 GetSerialPorts

This command lists all available serial ports of a device. A device that has one or more physical serial ports shall support listing of available serial ports through the GetSerialPorts command as described in Table 91.

Table 91 – GetSerialPorts command

GetSerialPorts	Access class: READ_SYSTEM
Message name	**Description**
GetSerialPortsRequest	*This is an empty message.*
GetSerialPortsResponse	*Contains a list of structures describing all available serial ports of the device. If a device has no serial ports an empty list is returned* tmd:SerialPort **SerialPort**[0][unbounded]
Fault codes	**Description**
No specific fault codes.	

9.4.3 GetSerialPortConfiguration

This operation gets a list of all available serial ports and their settings as described in Table 92.

Table 92 – GetSerialPortConfiguration command

GetSerialPortConfiguration	Access class: READ_SYSTEM
Message name	**Description**
GetSerialPortConfiguration-Request	*This message contains the token of the serial port.* tt:ReferenceToken **SerialPortToken**[1][1]
GetSerialPortConfiguration-Response	*This message contains an array of SerialPortConfiguration.* Tmd:SerialPortConfiguration **SerialPortConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*The supplied serial port token does not exist.*

9.4.4 SetSerialPortConfiguration

This operation sets the setting of serial port as described in Table 93.

Table 93 – SetSerialPortConfiguration command

SetSerialPortConfiguration	Access class: WRITE_SYSTEM
Message name	**Description**
SetSerialPortConfiguration-Request	*The **SerialPortToken** element specifies the serial port whose configuration is to be modified.* *The **SerialPortConfiguration** element contains the modified serial port configuration.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:ReferenceToken **SerialPortToken**[1][1] tmd:SerialPortConfiguration **SerialPortConfiguration** [1][1] xs:boolean **ForcePersistence**[1][1]
SetSerialPortConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*The supplied serial port token does not exist.*
Env:Sender ter:InvalidArgVal ter:ConfigModify	*It is not possible to set the the configuration parameters.*

9.4.5 GetSerialPortConfigurationOptions

This operation requests the SerialPortConfigurationOptions of a serial port. A device that has one or more serial ports shall support this command as described in Table 94.

Table 94 – GetSerialPortConfigurationOptions command

GetSerialPortConfigurationOptions	Access class: READ_SYSTEM
Message name	**Description**
GetSerialPortConfigurationOptions-Request	*The **SerialPortToken** element specifies the serial port whose options are requested.* tt:ReferenceToken **SerialPortToken**[1][1]
GetSerialPortConfigurationOptions-Response	*This message contains:* tmd:SerialPortConfigurationOptions **SerialPortConfigurationOptions** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*The supplied serial port token does not exist.*

9.4.6 Send and/or Receive serial command

Subclause 9.4.6 describes operations to transmit/receive generic controlling data to/from a serial device that is connected to the serial port of the device.

This operation can be used for the following purposes.

- Transmitting arbitrary data to the connected serial device.
- Receiving data from the connected serial device.
- Transmitting arbitrary data to the connected serial device and then receiving its response data.

In order to make use of this command for the above purpose, this document defines the input parameter structure as follows.

- token

  This element shall be present in the request. It indicates the physical serial port reference to be used when this request is invoked.

- SerialData

  Putting this element in the request is optional. When transmitting serial data is needed, the request should contain the element.

- TimeOut

  Putting this element in the request is optional. Depending on the specified value, it is possible for various configurations as follows.

  i) TimeOut > PT0S: Indicates that the command should get a response back within the specified period of time. In the case the device received the data which meets one of the following conditions of DataLength and Delimiter, the device should respond back with the received data instead of waiting for the specified time.

  ii) TimeOut = PT0S: Indicates that the command should get a response back immediately (Non-blocking). It will be used in the case of only transmitting data.

  iii) TimeOut = -PT1S: Indicates that the command should get a response after one of the following conditions (DataLength/Delimiter) is met. How long the device can hold the blocking state is vendor specific.

  If this element is not present in the request, the command should be responded after one of the following conditions (DataLength/Delimiter) is met. How long the device can hold the blocking state is *vendor specific*.

- DataLength

  Putting this element in the request is optional. This element may be put in the case that the data length returned from the connected serial device is already determined as some fixed bytes length. It indicates the length of received data which can be regarded as available.

- Delimiter

  Putting this element in the request is optional. This element may be put in the case that the delimiter codes returned from the connected serial device is already known. It indicates the termination data sequence of the responded data. In case the string has more than one character a device shall interpret the whole string as a single delimiter. Furthermore a device shall return the delimiter character(s) to the client.

A device that indicates generic serial communication service capability shall support this command as described in Table 95.

Table 95 – Send and/or Receive serial command

SendReceiveSerialCommand	Access class: ACTUATE
Message name	**Description**
SendReceiveSerialCommandRequest	*See above for information about the paramaters. This message contains:* tmd:SerialData**SerialData** [0][1] xs:duration **TimeOut** [0][1] xs:integer **DataLength** [0][1] xs:string **Delimiter** [0][1]
SendReceiveSerialCommandResponse	*This message contains the serial data.* tmd:SerialData **SerialData** [0][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*The supplied serial port token does not exist.*
Env:Sender ter:OperationProhibited ter:DataLengthOver	*Number of available bytes exceeded.*
Env:Sender ter:OperationProhibited ter:DelimiterNotSupported	*Sequence of character (delimiter) is not supported.*

9.5 Capabilities

The capabilities reflect optional functions and functionality of a service as described in Table 96. The information is static and does not change during device operation. The following capabilites are available:

RelayOutputs: Number of relay outputs (defaults to none).

DigitalInputs: Number of digital inputs (defaults to none).

SerialPorts: Number of serial ports (defaults to none).

Table 96 – GetServiceCapabilities command

GetServiceCapabilities	Access class: PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*This is an empty message.*
GetServiceCapabilitiesResponse	*The capability response message contains the requested service capabilities using a hierarchical XML capability structure.* tmd:Capabilities **Capabilities** [1][1]
Fault codes	**Description**
	No command specific faults!

9.6 Events

9.6.1 DigitalInput state change

A device that signals support for digital inputs in its capabilities shall provide the following event whenever one of its input state changes:

```
Topic: tns1:Device/Trigger/DigitalInput

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="InputToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Digital Input LogicalState can be either set at "true" to represent the circuit in the closed state or set at "false" to represent the circuit in the open state.

9.6.2 Relay output trigger

A device that signals RelayOutputs in its capabilities shall provide the Trigger event whenever a relay output state is changed. A device shall use the following topic and message format:

```
Topic: tns1:Device/Trigger/Relay

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RelayToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState"
type="tt:RelayLogicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

9.7 Service specific fault codes

Table 97 below lists the DeviceIO service specific fault codes. Additionally, each command can also generate a generic fault as defined in this document.

Table 97 – DeviceIO service specific fault codes

Fault code	Parent subcode Subcode	Fault reason	Description
env:Sender	ter:InvalidArgVal Ter:ConfigModify	Invalid configuration parameters	It is not possible to set the configuration parameters.
env:Sender	ter:InvalidArgVal Ter:RelayToken	Unknown relay token reference	The requested RelayOutput indicated with RelayOutputToken does not exist.
env:Sender	ter:InvalidArgVal ter:ModeError	Monostable delay time not valid	The requested Monostable delay time is not valid.
env:Sender	ter:InvalidArgVal Ter:InvalidSerialPort	Serial port token not valid	The supplied serial port token does not exist.
env:Sender	ter:OperationProhibited Ter:DataLengthOver	Data length over	Number of available bytes exceeded.
env:Sender	ter:OperationProhibited	Delimiter is not supported	Sequence of character (delimiter) is not supported.

10 Event handling

10.1 General

An event is an action or occurrence detected by a device that a client can subscribe to. Events are handled through the event service. This document defines event handling based on the [OASIS WS-BaseNotification] and [OASIS WS-Topics] specifications. It extends the event notion to allow clients to track object properties (such as digital input and motion alarm properties) through events. Properties are defined in 10.4.

The description of event payload and their filtering within subscriptions is discussed in 10.5. Subclause 10.6 describes how a synchronization point can be requested by clients using one of the notification interfaces. Subclause 10.7 describes the integration of topics and 10.10 discusses the handling of faults.

Subclause 10.11 demonstrates the usage of the Real-Time Pull-Point notification interface including Message Filtering and Topic Set. Examples for the basic notification interface can be found in the corresponding [OASIS WS-BaseNotification] specification.

A device shall provide an event service as defined in Clause B.3. Both device and client shall support [W3C WS-Addressing] for the event services.

10.2 Real-time Pull-Point notification interface

10.2.1 General

Subclause 10.2 introduces the Real-time Pull-Point notification interface. This interface provides a firewall friendly notification interface that enables real-time polling and initiates all client communications.

This interface is used in the following way:

1) The client asks the device for a pull point with the CreatePullPointSubscriptionRequest message.
2) The device evaluates the subscription request and returns either a CreatePullPointSubscriptionResponse or one of the fault codes.
3) If the subscription is accepted, the response contains a WS-EndpointReference to the instanciated pull point. This WS-Endpoint provides a PullMessages operation, which is used by the client to retrieve notifications. Additionally it provides the Renew and Unsubscribe operations of the base subscription manager interface. The sequence diagram of the interaction is shown in Figure 2.

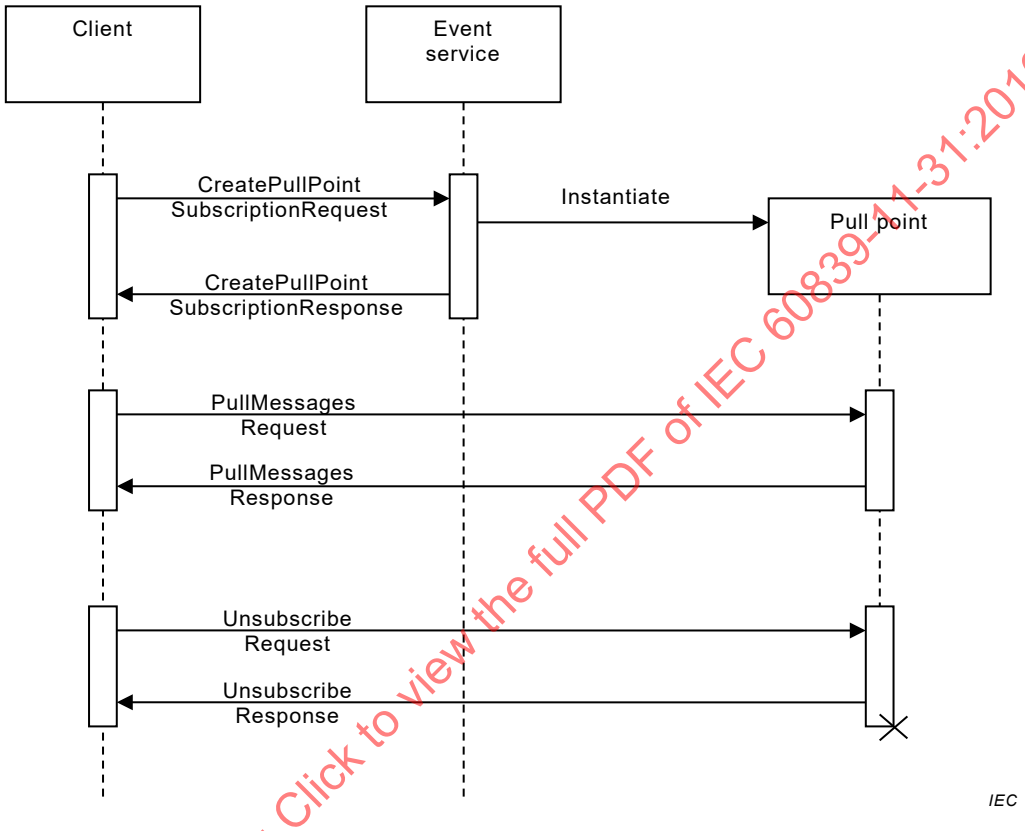


Figure 2 – Sequence diagram for the Real-time Pull-Point notification interface

4) The device shall immediately respond with notifications that have been aggregated on behalf of the client. If there are no aggregated notifications, the device waits to respond until either a notification is produced for the client or the specified Timeout has exceeded. In any case, the response will contain, at most, the number of notifications specified by the MessageLimit parameter. The client can poll the notifications in real-time when it starts a new PullMessagesRequest immediately after each PullMessagesResponse.

For a device implementation it is important to support multiple pull points (including multiple pull points per client) in order to allow precise event generation. If a device were to only support one subscription at a time, a client would need to subscribe without any scope restriction, because changing of event subscription is not possible. Hence this would require the device to serve all available events for which the device would have to activate all subsystems that generate events. This may cause unnecessary load on the device. Additionally the traffic produced by all these events may cause a substantial network load.

If the device supports persistent notification storage, see 10.2.8, the WS-Endpoint also provides a Seek operation. This operation allows to reposition the pull pointer into the past. With the Seek operation it is also possible to reverse the pull direction. A BeginOfBuffer event also exists, as defined in 10.12, that signals the start of the buffer.

A device shall implement the Real Time Pull-Point notification interface.

10.2.2 Create pull point subscription

A device shall provide the CreatePullPointSubscription command as described in Table 98. If no Filter element is specified the pull point shall notify all occurring events to the client.

By default the pull point keep alive is controlled via the PullMessages operation. In this case, after a PullMessages response is returned, the subscription should be active for at least the timeout specified in the PullMessages request.

If a client wants to control the pull point lifetime via Renew calls it shall use the optional parameter InitialTerminationTime. A device shall support an absolute time value specified in UTC as well as a relative time value for the InitialTerminationTime parameter. A device shall respond to both parameters CurrentTime and TerminationTime as UTC using the 'Z' indicator.

The following optional subscription policy elements are defined in tev:SubscriptionPolicy:

- **tev:ChangedOnly** A pull point should not provide initialized nor deleted events for properties.

Table 98 – CreatePullPointSubscription command

CreatePullPointSubscription	Access class: READ_MEDIA
Message name	**Description**
CreatePullPointSubscription-Request	*This message contains the same elements as the SubscriptionRequest of the [OASIS WS-BaseNotification] without the ConsumerReference:* wsnt:FilterType **Filter** [0][1] wsnt:AbsoluteOrRelativeTimeType **InitialTerminationTime** [0][1] xs:any **SubscriptionPolicy** [0][1]
CreatePullPointSubscription-Response	*The response contains the same elements as the SubscriptionResponse of the [OASIS WS-BaseNotification]:* wsa:EndpointReferenceType **SubscriptionReference** [1][1] xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1]
Fault codes	**Description**
	The same faults as for Subscription Request of the [OASIS WS-BaseNotification] are used.

10.2.3 Pull messages

The device shall provide the following PullMessages command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command as described in Table 99.

The command shall at least support a timeout of one minute. In case a device supports retrieval of less messages than requested it shall return these without generating a fault.

A device shall respond both parameters CurrentTime and TerminationTime as UTC using the 'Z' indicator.

After a seek operation the device shall return the messages in strict message UTC time order. Note that this requirement is not applicable to standard realtime message delivery where the delivery order may be affected by device internal computations.

A device should return an error (UnableToGetMessagesFault) when receiving a PullMessages request for a subscription where a blocking PullMessage request already exists.

Table 99 – PullMessages command

PullMessages	Access class: READ_MEDIA
Message name	**Description**
PullMessagesRequest	*This message shall be addressed to a SubscriptionManager in order to pull notifications:* xs:duration **Timeout** [1][1] xs:int **MessageLimit** [1][1]
PullMessagesResponse	*The response contains a list of notifications together with an updated TerminationTime for the SubscriptionManager:* xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1] wsnt:NotificationMessageHolderType **NotificationMessage** [0][unbounded]
PullMessagesFaultResponse	*The Timeout exceeds the upper limit supported by the device. The fault message shall contain the upper limits for both parameters.* xs:duration **MaxTimeout**[1][1] xs:int **MaxMessageLimit**[1][1]
Fault codes	**Description**
	No specific fault codes.

10.2.4 Renew

The device shall provide the following Renew command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command as described in Table 100.

The command shall at least support a timeout of one minute. A device shall respond both parameters CurrentTime and TerminationTime as UTC using the 'Z' indicator.

Table 100 – Renew command

Renew	Access class: READ_MEDIA
Message name	**Description**
RenewRequest	*This message contains the new relative or absolute termination time:* wsnt:AbsoluteOrRelativeTimeType **TerminationTime** [1][1]
RenewResponse	*The response contains a list of notifications together with an updated TerminationTime for the SubscriptionManager:* xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1]
ResourceUnknownFaultResponse	*The pull point reference is invalid* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
UnacceptableTerminationTimeFaultResponse	*The Timeout exceeds the upper limit supported by the device.* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
Fault codes	**Description**
	No specific fault codes.

10.2.5 Unsubscribe

The device shall provide the following Unsubscribe command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command as described in Table 101.

This command shall terminate the lifetime of a pull point.

Table 101 – Unsubscribe command

Unsubscribe	Access class: READ_MEDIA
Message name	**Description**
UnsubscribeRequest	*This message is empty.*
UnsusbscribeResponse	*This message is empty.*
ResourceUnknownFault-Response	*The pull point reference is invalid* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
Fault codes	**Description**
	No specific fault codes.

10.2.6 Seek

A device supporting persistent notification storage as defined in 10.2.8 shall provide the following Seek command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command as described in Table 102.

On a Seek a pull point shall abort any event delivery including any initial states of properties. Furthermore the pull point should flush events not already queued for transmission from the transmit queue.

After a Seek request a pull point shall ignore the behaviour described in 10.6 for properties.

A device shall only set the subscription in reverse pull mode if the Reverse argument is present and set to “true”.

The UtcTime argument of the Seek request shall be matched against the UtcTime attribute of the notifications in the persistent notification storage.

When Seek is used in the forward mode a device shall position the pull pointer to include all NotificationMessages in the persistent storage with a UtcTime attribute greater than or equal to the Seek argument.

When Seek is used in reverse mode a device shall position the pull pointer to include all NotificationMessages in the persistent storage with a UtcTime attribute less than or equal to the Seek argument.

A device shall not provide information of the initial generated property state as response to a call to the Seek method.

Table 102 – Seek command

Seek	Access class: READ_MEDIA
Message name	**Description**
SeekRequest	*This message shall be addressed to a PullPoint in order to readjust the pull position:* xs:datetime Utc**Time**[1][1] xs:bool **Reverse**[0][1]
SeekResponse	*This message is empty*
Fault codes	**Description**
	No specific fault codes.

10.2.7 Pull point lifecycle

Figure 2 depicts the basic operation of a pull point. Subclause 10.2.7 states the requirements on the pull point lifecycle.

A device shall create a new pull point on each CreatePullPointSubscription command as long as the number of instantiated pull points does not exceed the capability MaxPullPoints. Each pull point shall have a unique endpoint reference to which the client can direct its consecutive operations on the pull point.

A pull point shall exist until either its termination time has elapsed or the client has requested its disposal via an Unsubscribe request. There are no requirements regarding persitancy of a pull point across the power cycle of a device.

10.2.8 Persistent notification storage

To ensure that no notifications are lost by a client a device may store its notifications. The stored notifications can at any time be retrieved by a client. The device shall indicate if it supports persistent notification storage with the PersistentNotificationStorage capability. See 10.9.

This document states that the interface to the persistant storage allows linear access via the originating message event time. This holds also for events that are delivered out of order in the live streaming case due to, for example, computational delay.

The details of what notification and how and where those notifications actually are stored are outside the scope of this document. Removal policy of stored notifications to make room for new ones is also outside the scope of this document.

10.3 Basic notification interface

10.3.1 General

Subclause 10.3.2 briefly introduces the basic notification interface of the [OASIS WS-BaseNotification] specification. Subclause 10.3.3 summarizes the mandatory and the optional interfaces of the [OASIS WS-BaseNotification] specification. For a full documentation of the basic notification interface refer to the [OASIS WS-BaseNotification] specification.

10.3.2 Summary

The following logical entities participate in the notification pattern:

- Client: implements the NotificationConsumer interface.
- Event service: implements the NotificationProducer interface.
- Subscription manager: implements the BaseSubscriptionManager interface.

The event service and the subscription manager should be instantiated on a device.

Typical messages exchanged between the entities are shown in the sequence diagram in Figure 3. First, the client establishes a connection to the event service. The client can then subscribe to certain notifications by sending a SubscriptionRequest. If the event service accepts the subscription, it dynamically instantiates a SubscriptionManager representing the subscription. The event service shall return the WS-Endpoint-Address of the SubscriptionManager in the SubscriptionResponse.

In order to transmit notifications matching the subscription, another connection is established from the event service to the client. Via this connection, the event service sends a one-way Notify message to the NotificationConsumer interface of the client. Corresponding

notifications can be sent at any time by the event service to the client, while the subscription is active.

To control the subscription, the client directly addresses the SubscriptionManager returned in the SubscriptionResponse. In the SubscriptionRequest the client can specify a termination time. The SubscriptionManager is automatically destroyed when the termination time is reached. RenewRequests can be initiated by the client in order to postpone the termination time. The client can also explicitly terminate the SubscriptionManager by sending an UnsubscribeRequest. After a successful Unsubscription, the SubscriptionManager no longer exists.

The interaction between EventService and SubscriptionManager is not further specified by the [OASIS WS-BaseNotification] and is up to the implementation of the device.

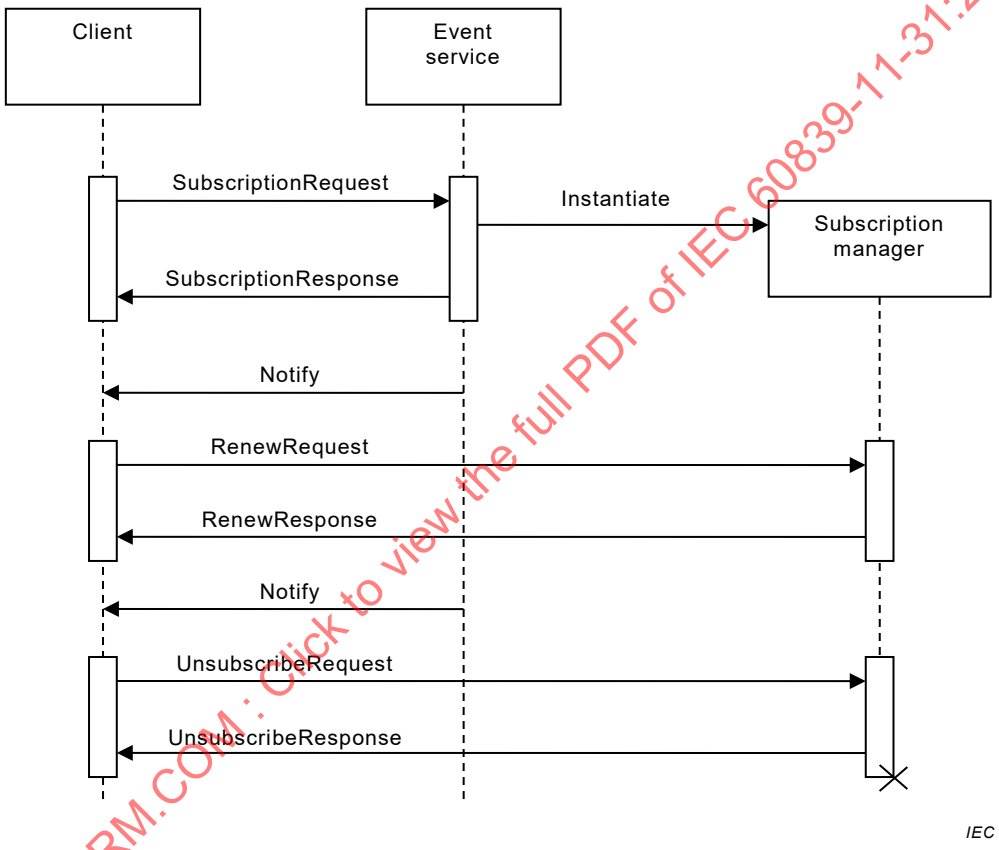


Figure 3 – Sequence diagram for the base notification interface

10.3.3 Requirements

Subclause 10.3.3 details those interfaces of the [OASIS WS-BaseNotification] that a device shall provide.

A device shall support the NotificationProducer interface of the [OASIS WS-BaseNotification] if the capability MaxNotificationProducers is non-zero. The device shall support TopicExpression filters with the dialects described in 10.7.4. The support for MessageContent filters is signalled via the GetEventProperties method. If the device does not accept the InitialTerminationTime of a subscription, it shall provide a valid InitialTerminationTime within the fault message. The device shall be able to provide notifications using the Notify wrapper of the [OASIS WS-BaseNotification] specification. The SubscriptionPolicy `wsnt:UseRaw` is optional for the device. Although the [OASIS WS-BaseNotification] has CurrentTime and TerminationTime as optional elements in a SubscribeResponse and RenewResponse, a device shall list them in both SubscribeResponses and RenewResponse. The device may

respond to any GetCurrentMessage request with a fault message indicating that no current message is available on the requested topic.

The implementation of the Pull-Point interface of the [OASIS WS-BaseNotification] on a device is optional.

A device shall implement the base subscription manager interface of the [OASIS WS-BaseNotification] specification consisting of the Renew and Unsubscribe operations. The pausable subscription manager interface is optional. The implementation of subscriptions as WS-Resources is optional.

A device shall support time values in request parameters that are given in UTC with the 'Z' indicator and respond all time values as UTC including the 'Z' indicator.

10.4 Properties

A property is a collection of name and value pairs representing a unique and addressable set of data. They are uniquely identified by the combination of their topic, source and key values and are packaged like ordinary events. A property also contains an additional flag, stating whether it is newly created, has changed or has been deleted.

When a client subscribes to a topic representing a certain property, the device shall provide notifications informing the client of all objects with the requested property, which are alive at the time of the subscription. A client can also request the values of all currently alive properties the client has subscribed to at any time by asking for a synchronization point (see 10.6).

The property interface is defined in this document in order to group all property related events together and to present uniformly to clients. It is recommended to use the property interface wherever applicable. Subclause 10.5 explains the structure of events and properties in detail.

10.5 Notification structure

10.5.1 General

The following code is the schema for the `wsnt:NotificationMessage` [OASIS WS-BaseNotification]:

```
<xs:complexType name="NotificationMessageHolderType" >
  <xs:sequence>
    <xs:element ref="wsnt:SubscriptionReference" minOccurs="0" />
    <xs:element ref="wsnt:Topic" minOccurs="0" />
    <xs:element ref="wsnt:ProducerReference" minOccurs="0" />
    <xs:element name="Message">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:element name="NotificationMessage"
            type="wsnt:NotificationMessageHolderType"/>
```

This corresponds to the following XML structure:

```
<wsnt:NotificationMessage>
  <wsnt:SubscriptionReference>
    wsa:EndpointReferenceType
  </wsnt:SubscriptionReference>
  <wsnt:Topic Dialect="xs:anyURI">
```

```
    …
  </wsnt:Topic>?
  <wsnt:ProducerReference>
    wsa:EndpointReferenceType
  </wsnt:ProducerReference>
  <wsnt:Message>
    …
  </wsnt:Message>
</wsnt:NotificationMessage>
```

where the `wsnt:Message` element contains the actual notification payload. The XML type of the Message element can be specified within a TopicTree definition, see 10.7.

Subclause 10.5.2 gives an overview of the information a client retrieves through notifications. Subclause 10.5.3 gives a detailed formatting of the message payload, and 10.5.4 introduces a description language for the message payload. Subclause 10.5.5 defines the grammar used in a subscription to filter notifications by their message content.

10.5.2 Notification information

10.5.2.1 General

A notification answers at least the following questions:

- When did it happen?
- Who produced the event?
- What happened?

The “when” question is answered by adding a time attribute to the Message element of the NotificationMessage. A device shall include the time attribute to the Message element.

The “who” question is split into two parts. One part is the WS-Endpoint which identifies the device or a service within the device where the notification has been produced. Therefore, the WS-Endpoint should be specified within the ProducerReference element of the NotificationMessage. The second part is the identification of the component within the WS-Endpoint, which is responsible for the production of the notification. Depending on the component multiple parameters or none may be needed to identify the component uniquely. These parameters are placed as Items within the Source element of the Message container.

The “what” question is answered in two steps. First, the Topic element of the NotificationMessage is used to categorize the event. Second, items are added to the Data element of the Message container in order to describe the details of the event.

When the topic points to properties, see 10.4, the client uses the NotificationProducer, the Topic, the Source items and optional Key items, see 10.5.3, in order to identify the property. These values shall result in a unique identifier.

10.5.2.2 Event example

The subsequent example demonstrates the different parts of the notification:

```
<wsnt:NotificationMessage>
  ...
  <wsnt:Topic Dialect="...Concrete">
    tns1:Device/Trigger/DigitalInput
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="..." PropertyOperation=”...”>
      <tt:Source>
        <tt:SimpleItem Name="InputToken"       Value="2"/>
      </tt:Source>
      <tt:Data>
```

```
        <tt:SimpleItem Name="LogicalState"     Value="true"/>
      </tt:Data>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

The item “InputToken” identifies uniquely the component, which is responsible for the detection of the event. In this example, the component is a digal input referenced by the “2” token. The event `tns1:Device/Trigger/DigitalInput` indicates the digial input state has changed. The data block contains the state.

10.5.3 Message format

The Message element of the NotificationMessage is defined in the common schema, see Clause B.4. The definition is presented below:

NOTE The schema is included here for information only. Clause B.4 contains the normative schema definition.

```
<xs:element name="Message" type="Message">

<xs:element name="Message">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Source" type="tt:ItemList" minOccurs="0"/>
      <xs:element name="Key" type="tt:ItemList" minOccurs="0"/>
      <xs:element name="Data" type="tt:ItemList" minOccurs="0"/>
      ...
    </xs:sequence>

    <xs:attribute name="UtcTime" type="xs:dateTime" use="required"/>
    <xs:attribute name="PropertyOperation" type="tt:PropertyOperationType"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="ItemList">
  <xs:sequence>
    <xs:element name="SimpleItem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Value" type="xs:anySimpleType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ElementItem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any"/>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="PropertyOperationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Initialized"/>
    <xs:enumeration value="Deleted"/>
    <xs:enumeration value="Changed"/>
  </xs:restriction>
</xs:simpleType>
```

The items within the Message element are grouped into three categories: Source, Key, and Data. The Key group shall not be used by notifications which are not related to properties. Multiple Simple and Element items can be placed within each group. Each item has a name and a value. In the case of an ElementItem, the value is expressed by one XML element within the ElementItem element. In the case of a SimpleItem, the value shall be specified by

the value attribute. The name of all items shall be unique within all items contained in any group of this message.

Vendor specific extensions shall express the SimpleItem and ElementItem Name attributes as qname. This avoids potential name clashes between vendor specific extensions and future extensions to this document.

It is recommended to use SimpleItems instead of ElementItems whenever applicable, since SimpleItems ease the integration of messages into a generic client. The exact type information of both Simple and ElementItems can be extracted from the TopicSet (see 10.7), where each topic can be augmented by a description of the message payload.

The PropertyOperation shall be present when the notification relates to a property. The operation mode “Initialized” shall be used to inform a client about the creation of a property. The operation mode “Initialized” shall be used when a synchronization point has been requested.

10.5.4 Message description language

10.5.4.1 General

The structure of the message payload was introduced in 10.5.3. The structure contains three groups: Source, Key, and Data. Each group contains a set of Simple and ElementItems. For each topic, a device can describe which item will be part of a notification produced by this topic using a message description language. The following description language describes the mandatory message items:

NOTE The schema is included here for information only. Clause B.4 contains the normative schema definition.

```
<xs:complexType name="MessageDescription">
  <xs:sequence>
    <xs:element name="Source" type="tt:ItemListDescription"
                minOccurs="0"/>
    <xs:element name="Key" type="tt:ItemListDescription" minOccurs="0"/>
    <xs:element name="Data" type="tt:ItemListDescription" minOccurs="0"/>
    ...
  </xs:sequence>
  <xs:attribute name="IsProperty" type="xs:boolean"/>
</xs:complexType>

<xs:complexType name="ItemListDescription">
  <xs:sequence>
    <xs:element name="SimpleItemDescription"
                minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:QName" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ElementItemDescription"
                minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:QName" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

The Name attribute of an item shall be unique within all items independent from the group (Source, Key, Data) they are coming from. The IsProperty attribute shall be set to true when the described message relates to a property. If the message, however, does not relate to a property, the Key group shall not be present. The Type attribute of a SimpleItemDescriptor

shall use simple type defined in XML schema (built in simple types), schemas defined by this document, or vendor schemas. Similarly, the Type attribute of an ElementItemDescriptor shall match a global element declaration of an XML schema.

The message description language does not mandate the order of the items in each of the categories Source, Key and Data. Additionally items documented as optional by an event definition are not required to be present in a message. This applies also to optional items that are described in the related MessageDescription.

The location of all schema files used to describe message payloads are listed in the GetEventPropertiesResponse message in 10.8.

10.5.4.2 Message description example

The following code is an example of a message description corresponding to the event example of 10.5.2.2:

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="InputToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

10.5.5 Message content filter

In the subscription request, a client can filter notifications by TopicExpression (see 10.7.4) and by MessageContent. For the latter, the [OASIS WS-BaseNotification] proposes the XPath 1.0 dialect. Due to the specific message structure required by this document, the specification requires a subset of the [W3C Xpath] syntax. The corresponding dialect can be referenced with the following URI:

```
Dialect=http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter
```

Precedence and associativity:

The 'and' operation has higher precedence than the 'or' operation. Both 'and' and 'or' operations are left associative.

The precedence and associativity of 'and' and 'or' operations in the following grammar definition are identical to XPath 1.0 specifications.

The structure of the expressions is as follows:

[1] Expression::= BoolExpr | Expression ‘and’ Expression
| Expression ‘or’ Expression | ‘(‘ Expression ‘)’ | ‘not’ ‘(‘ Expression ‘)’

[2] BoolExpr::= ‘boolean’ ‘(‘ PathExpr ‘)’

[3] PathExpr::= [‘//’Prefix?’SimpleItem‘ | ’//’Prefix?’ElementItem‘] NodeTest

[4] Prefix::= NamespacePrefix’:’ | ‘’

[5] NodeTest::= ‘[‘ AttrExpr ‘]’

[6] AttrExpr::= AttrComp | AttrExpr ‘and’ AttrExpr | AttrExpr ‘or‘ AttrExpr | ‘(‘ AttrExpr ‘)‘ | ‘not’ ‘(‘ AttrExpr ‘)‘

[7] AttrComp::= Attribute ‘=’ ‘”’ String ‘”’

[8] Attribute::= ‘@Name’ | ‘@Value’

This grammar allows testing the presence of Simple or ElementItems independently of the group they belong to (Source, Key or Data). Furthermore, the value of SimpleItems can be checked. The SimpleItem and ElementItem prefix namespace shall correspond to “http://www.onvif.org/ver10/schema.

Finally, arbitrary boolean combinations of these tests are possible. The following expressions can be formulated:

- Return only notifications which contain a reference to InputToken “1”

```
boolean(//tt:SimpleItem[@Name=” InputToken” and @Value=”1”] )
```

- Return only notifications which do not contain a reference to a RelayToken

```
not(boolean(//tt:SimpleItem[@Name=” RelayToken”]))
```

- Return only notifications which do relate to InputToken “2” with the LogicalState “true”

```
boolean(//tt:SimpleItem[@Name=”  InputToken”  and  @Value=”2”])
and
boolean(//tt:SimpleItem[@Name=”LogicalState” and @Value=”true”])
```

10.6 Synchronization point

Note that 10.2.6 defines rules for devices supporting persistent notification storage that override the behaviour defined in this subclause.

Properties, introduced in 10.4, inform a client about property creation, changes and deletion in a uniform way. When a client wants to synchronize its properties with the properties of the device, it can request a synchronization point which repeats the current status of all properties to which a client has subscribed as described in Table 103. The PropertyOperation of all produced notifications is set to “Initialized” (see 10.5). The synchronization point is requested directly from the SubscriptionManager which was returned in either the SubscriptionResponse or in the CreatePullPointSubscriptionResponse. The property update is transmitted via the notification transportation of the notification interface. The following operation shall be provided by all subscription manager endpoints:

Table 103 – SetSynchronizationPoint command

SetSynchronizationPoint	Access class: READ_MEDIA
Message name	**Description**
SetSynchronizationPointRequest	*This message is empty.*
SetSynchronizationPointResponse	*This message is empty.*
Fault codes	**Description**
	No command specific faults!

10.7 Topic structure

10.7.1 General

This document extends the Topic framework defined in the [OASIS WS-Topics] specification.

Subclause 10.7.2 describes the ONVIF topic namespace. Subclause 10.7.3 incorporates the message description language defined in 10.5.4 into the TopicSet structure, furthermore 10.8

defines an interface that allows a client to get this information from a device. A topic expression dialects to be supported by a device is defined in 10.7.4.

Concrete event definitions are specified in the events sections of the service specifications.

10.7.2 ONVIF topic namespace

The [OASIS WS-Topics] specification distinguishes between the definition of a topic tree belonging to a certain topic namespace and the topic set supported by a certain Web service. This distinction allows vendors to refer to a common topic namespace while only using a portion of the defined topics.

If the topic tree of an existing topic namespace covers only a subset of the topics available by a device, the topic tree can be grown by defining a new topic namespace. A new topic namespace is defined by appending a new topic to an existing topic namespace as described in the [OASIS WS-Topics] specification.

All notifications referring to topics in the ONVIF topic namespace, http://www.onvif.org/ver10/topics, shall use the message format as described in 10.5.3.

10.7.3 Topic type information

A device shall add a MessageDescription element, of type MessageDescriptionType defined in 10.5.4, below all elements representing topics in the topic set supported by the device. Furthermore a device shall, in accordance with the notification specification, identify all elements representing topics in the topic set by including the wstop:topic attribute with value "true".

The following example demonstrates how topics of a TopicSet are augmented with message descriptions:

```
<wstop:TopicSet xmlns="">
  <tns1:Device>
    <Trigger>
      <DigitalInput wstop:topic="true">
        <tt:MessageDescription IsProperty="true">
          <tt:Source>
            <tt:SimpleItemDescription Name="InputToken"
                                      Type="tt:ReferenceToken"/>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="LogigalState" Type="xs:boolean"/>
          </tt:Data>
        </tt:MessageDescription>
      </DigitalInput>
      <Relay wstop:topic="true">
        <tt:MessageDescription IsProperty="true">
          <tt:Source>
            <tt:SimpleItemDescription Name="RelayToken"
                                      Type="tt:ReferenceToken"/>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="LogicalState"
                                      Type="xs:RelayLogicalState"/>
          </tt:Data>
        </tt:MessageDescription>
      </Relay>
    </Trigger>
  </tns1:Device>
</wstop:TopicSet>
```

NOTE xmlns="" is included in the example to make sure that there is no default namespace in scope for any of the descendants of the TopicSet element, see the [WS-Topics] specification for more information.

10.7.4 Topic filter

A device shall support the concrete topic expressions defined in the [OASIS WS-Topics] specification. This document defines the identification of a specific topic within topic trees. The following dialect shall be specified when a concrete topic expression is used as TopicExpression of a subscription filter:

```
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete
```

The following topic expression syntax shall be supported by a device.

The syntax extends the concrete topic expressions by an “or” operation and topic sub-tree matching string. This extended syntax allows selection of an arbitrary TopicSet within a single subscription. The grammar is described in the same way as the topic expressions of the [OASIS WS-Topics] specification:

[3] TopicExpression::= TopicPath (‘|’ TopicPath)*

[4] TopicPath::= RootTopic ChildTopicExpression* *('//.')?*

[5] RootTopic::= QName

If a namespace prefix is included in the RootTopic, it shall correspond to a valid topic namespace definition and the local name shall correspond to the name of a root topic defined in that namespace.

[6] ChildTopicExpression::= ‘/’ ChildTopicName

[7] ChildTopicName::= QName | NCName

The NCName or local part of the QName shall correspond to the name of a topic within the descendant path from the RootTopic, where each forward slash denotes another level of child topic elements in the path.

In order to reference this TopicExpression dialect, the following URI shall be used:

```
Dialect=http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
```

If the TopicExpression ends with the characters “//.” this indicates that the TopicExpression matches a topic sub-tree. For example:

“tns1:Device/Trigger //.”

This identifies the sub-tree consisting of tns1:Device/Trigger and all its descendants.

The following examples demonstrate the usage of the ConcreteSet topicExpression:

Look for notifications which have the tns1:Monitoring topic as parent topic:

```
<wsnt:TopicExpression Dialect =
```

- "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet" >
- tns1:Monitoring//.

```
</wsnt:TopicExpression>
```

Look for notifications which have the tns1:Monitoring topic or the tns1:Device topic as parent topic:

```
<wsnt:TopicExpression Dialect =
   • "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet" >
      • tns1:Monitoring//|tns1:Device//.
</wsnt:TopicExpression>
```

Look for notifications matching either the tns1:Monitoring/ProcessorUsage or the tns1:Device/Trigger/DigitalInput topics:

```
<wsnt:TopicExpression Dialect =
   • "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
      • tns1:Monitoring/ProcessosUsage|tns1:Device/Trigger/DigitalInputs
</wsnt:TopicExpression>
```

10.8 Get event properties

The [OASIS WS-BaseNotification] specification defines a set of optional WS-ResouceProperties. This document does not require the implementation of the WS-ResourceProperty interface. Instead, the subsequent direct interface shall be implemented by a device in order to provide information about the FilterDialects, schema files and topics supported by the device as described in Table 104.

Table 104 – GetEventProperties command

GetEventProperties	Access class: READ_MEDIA
Message name	**Description**
GetEventPropertiesRequest	*This is an empty message.*
GetEventPropertiesResponse	*This message contains:* xs:anyURI **TopicNamespaceLocation** [1][unbounded] xs:boolean **FixedTopicSet** [1][1] wstop:TopicSetType **TopicSet** [1][1] xs:anyURI **TopicExpressionDialect** [1][unbounded] xs:anyURI **MessageContentFilterDialect** [1][unbounded] xs:anyURI **ProducerPropertiesFilterDialect** [0][unbounded] xs:anyURI **MessageContentSchemaLocation** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

A device shall respond and declare if its TopicSet is fixed or not, which topics are provided, and which dialects are supported.

The following TopicExpressionDialects are mandatory for a device (see 10.7.4):

- `http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete`
- `http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet`

A device that does not support any MessageContentFilterDialect shall return a single empty URL.

This document does not require the support of any ProducerPropertiesDialect by a device.

10.9 Capabilities

The message content description language, introduced in 10.5.4, allows referencing of vendor-specific types. In order to ease the integration of such types into a client application, the GetEventPropertiesResponse as described in Table 105 shall list all URI locations to schema files whose types are used in the description of notifications, with

MessageContentSchemaLocation elements. This list shall at least contain the URI of the common schema file.

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilites are available:

WSSubscriptionPolicySupport: Indication if the device supports the WS Subscription policy according to 10.3.3.

WSPullPointSupport: Indication if the device supports the WS Pull Point according to 10.3.3.

WSPausableSubscriptionManagerInterfaceSupport: Indication if the device supports the WS Pausable Subscription Manager interface according to 10.3.3.

MaxNotificationProducers: Maximum number of supported notification producers as defined by [OASIS WS-BaseNotification].

MaxPullPoints: Maximum supported number of notification pull points.

PersistenNotificationStorage: Indication if the device supports persistent notification storage according to 10.2.8.

Table 105 – GetServiceCapabilities command

GetServiceCapabilities	Access class: PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*This is an empty message.*
GetServiceCapabilitiesResponse	*The capability response message contains the requested service capabilities using a hierarchical XML capability structure.* tev:Capabilities **Capabilities** [1][1]
Fault codes	**Description**
	No command specific faults!

10.10 SOAP fault messages

If a device encounters a failure while processing [OASIS WS-BaseNotification] messages from either a client or subscription manager, then the device shall generate a SOAP 1.2 fault message.

All SOAP 1.2 fault messages shall be generated according to [OASIS WS-BaseNotification] and [OASIS WS-Topics] specifications with one exception; All faults shall use the following URI for the WS-Addressing [action] Message Addressing Property:

`http://www.w3.org/2005/08/addressing/soap/fault`

Furthermore the error should be sent as a SOAP receiver fault (env:Receiver), i.e. the HTTP error code shall be 500.

10.11 Notification example

10.11.1 General

The following example is a complete communication pattern for notifications. It uses the Real-time Pull-Point notification interface to receive notifications.

10.11.2 GetEventPropertiesRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesReq
uest
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:GetEventProperties>
    </tet:GetEventProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.3 GetEventPropertiesResponse

In this example, the device response uses the ONVIF topic namespace. The topic set does not change over time and consists of the single topic `tns1:Device/Trigger/DigitalInput`. The device supports two TopicExpressionDialects.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
  xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
  xmlns:tns1="http://www.onvif.org/ver10/topics"
  xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRespons
e
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:GetEventPropertiesResponse>
      <tet:TopicNamespaceLocation>
        http://www.onvif.org/onvif/ver10/topics/topicns.xml
      </tet:TopicNamespaceLocation>
      <wsnt:FixedTopicSet>
        true
      </wsnt:FixedTopicSet>
      <wstop:TopicSet xmlns="">
        <tns1:Device>
          <Trigger>
            <DigitalInput wstop:topic="true">
              <tt:MessageDescription IsProperty="true">
                <tt:Source>
                  <tt:SimpleItemDescription Name="InputToken"
                                            Type="tt:ReferenceToken"/>
                </tt:Source>
                <tt:Data>
                  <tt:SimpleItemDescription Name="LogigalState"
                                            Type="xs:boolean"/>
                </tt:Data>
              </tt:MessageDescription>
```

```
            </ DigitalInput>
          </Trigger>
        </tns1:Device>
      </wstop:TopicSet>
      <wsnt:TopicExpressionDialect>
        http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
      </wsnt:TopicExpressionDialect>
      <wsnt:TopicExpressionDialect>
        http://docs.oasis-open.org/wsnt/t-1/TopicExpression/ConcreteSet
      </wsnt:TopicExpressionDialect>
      <wsnt:MessageContentFilterDialect>
        http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter
      </wsnt:MessageContentFilterDialect>
      <tt:MessageContentSchemaLocation>
        http://www.onvif.org/onvif/ver10/schema/onvif.xsd
      </tt:MessageContentSchemaLocation>
    </tet:GetEventPropertiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.4 CreatePullPointSubscription

A client can subscribe to specific notifications with the information from the TopicProperties. The following XML example shows the subscription for notifications produced below the device root topic of the device. The subscription has a timeout of one minute. If the subscription is not explicitly renewed or messages are not pulled regularly, it will be terminated automatically after this time.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
  xmlns:tns1="http://www.onvif.org/ver10/topics">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscripti
onRequest
    </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
    <tet:CreatePullPointSubscription>
      <tet:Filter>
        <wsnt:TopicExpression
         Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
          tns1:Device//.
        </wsnt:TopicExpression>
      </tet:Filter>
      <tet:InitialTerminationTime>
        PT1M
      </tet:InitialTerminationTime>
    </tet:CreatePullPointSubscription>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.5 CreatePullPointSubscriptionResponse

When the device accepts the subscription, it returns the `http://160.10.64.10/Subscription?Idx=0` URI which represents the endpoint of this subscription. Additionally, the client is informed about the CurrentTime of the device and the TerminationTime of the created subscription.

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl">
```

```
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscr
iptionResponse
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:CreatePullPointSubscriptionResponse>
      <tet:SubscriptionReference>
        <wsa:Address>
          http://160.10.64.10/Subscription?Idx=0
        </wsa:Address>
      </tet:SubscriptionReference>
      <wsnt:CurrentTime>
        2008-10-09T13:52:59
      </wsnt:CurrentTime>
      <wsnt:TerminationTime>
        2008-10-09T13:53:59
      </wsnt:TerminationTime>
    </tet:CreatePullPointSubscriptionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.6 PullMessagesRequest

The client sends a PullMessagesRequest to the endpoint given in the CreatePullPointSubscriptionResponse to get notifications corresponding to a certain subscription. The following sample request contains a timeout of five (5) seconds and limits the total number of messages in the response to two (2).

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl" >
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesR
equest        </wsa:Action>
    <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:PullMessages>
      <tet:Timeout>
       PT5S
      </tet:Timeout>
      <tet:MessageLimit>
        2
      </tet:MessageLimit>
    </tet:PullMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.7 PullMessagesResponse

The following PullMessageResponse contains notifications which match the subscription. The response informs the client that the state of DigitalInput “1” has changed to “true”.

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
    xmlns:tns1="http://www.onvif.org/ver10/topics"
    xmlns:tt="http://www.onvif.org/ver10/schema">
    <SOAP-ENV:Header>
```

```
      <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRespo
nse
      </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tet:PullMessagesResponse>
        <tet:CurrentTime>
          2008-10-10T12:24:58
        </tet:CurrentTime>
        <tet:TerminationTime>
          2008-10-10T12:25:58
        </tet:TerminationTime>
        <wsnt:NotificationMessage>
          <wsnt:Topic
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
              tns1:Device/Trigger/DigitalInput
          </wsnt:Topic>
          <wsnt:Message>
            <tt:Message UtcTime="2008-10-10T12:24:57.321Z"
                        PropertyOperation="Changed">
              <tt:Source>
                <tt:SimpleItem Name="InputToken" Value="1"/>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItem Name="LogicalState" Value="true"/>
                </tt:Data>
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
      </tet:PullMessagesResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.8 UnsubscribeRequest

A client has to terminate a subscription explicitly with an UnsubscribeRequest so that the device can immediately free resources. The request is directed to the subscription endpoint returned in the CreatePullPointSubscriptionResponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2" >
     <SOAP-ENV:Header>
          <wsa:Action>
  http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/UnsubscribeRequest              </wsa:Action>
          <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
          <wsnt:Unsubscribe/>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.9 UnsubscribeResponse

The subscription endpoint is no longer available once the device replies with an UnsubscribeResponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2" >
     <SOAP-ENV:Header>
          <wsa:Action>
http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/UnsubscribeResponse             </wsa:Action>
```

```
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <wsnt:UnsubscribeResponse/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.12 Persistent storage event:BeginingOfBuffer

The beginning of buffer event is a logical event that is connected to each subscription that signals that a subscription is reading passed the beginning of the buffer in either direction.

If a device supports persistent storage notification, it shall support the beginning of buffer event.

A device shall signal the beginning of buffer event when a subscription is reading, i.e. PullMessages, passed the beginning of persistent storage buffer either in forward or reverse direction.

Furthermore when a Seek operation has been done to before the beginning of buffer a device shall regardless of the direction of reading return the beginning of buffer event.

A device shall for each Seek operation on a subscription send the beginning of buffer event one time at most.

```
Topic: tns1:EventBuffer/Begin
<tt:MessageDescription IsProperty=”false”/>
```

10.13 Service specific fault codes

The event service does not define any service specific faults except those defined in [OASIS WS-BaseNotification].

11 Security

11.1 General

As is true for all network-oriented information technology, security is a most important subject for communication within an alarm and electronic security system. The security threat depends on the application. While some applications are most vulnerable to network based attacks, other applications are not at all sensitive. The cost for implementing security countermeasures varies depending on the type of attacks intended to prevent. These facts imply that we cannot list general security requirements for an alarm and electronic security system or its components, but can try to find a reasonable level of security requirements for devices conformant to this document and to define a basic security mechanism that allows building a secure alarm and electronic security system.

The current document defines security mechanisms on the transport level.

This document adopts the port-based authentication mechanism as follows.

- IEEE 802.1X

11.2 Transport level security

11.2.1 General

Transport level security protects the data transfer between the client and the server. Transport Layer Security (TLS) is regarded as a mature standard for encrypted transport connections to provide a basic level of communication security. The TLS protocol allows the configuration of a mutually authenticated transport session as well as preserving the confidentiality and the integrity protected transport.

A device conformant to this document should support TLS 1.0 [RFC 2246] and related specifications. The device should support TLS 1.1 [RFC 4346]. The device may support TLS 1.2 [RFC 5246].

A device should support TLS for protection of all of the services it provides. A device should also support TLS for protection of media streams for the RTP/RTSP/HTTPS tunnel option as defined in Clause 11. This document profiles a particular implementation of TLS and other relevant specifications that can be used with TLS.

A client should support TLS 1.0 [RFC 2246] and TLS 1.1 [RFC 4346]. The client may support TLS 1.2 [RFC 5246].

11.2.2 Supported cipher suites

A device that supports TLS shall support all of the following cipher suites [RFC 2246], [RFC 3268]:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

If a client supports TLS, then it shall support the following cipher suites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

11.2.3 Server authentication

A device that supports TLS shall support server authentication using TLS. The device shall support processing of X.509 server certificates. The RSA key length shall be at least 1 024 bits.

A client should support server authentication using TLS.

This document does not provide a full server certificate generation and Certificate Authority (CA) model. However, device management commands for device certificate retrieval and download are defined in 8.5.

The details of the server private key or keys secure bootstrapping mechanisms are outside the scope of the current document. However, commands for on board key generation are defined in 8.5.

11.2.4 Client authentication

A device that supports TLS should support client authentication. Client authentication can be enabled/disabled with a device management command as described in 8.5

A device that supports TLS shall include the RSA certificate type (rsa_sign, for example) in the certificate request [RFC 2246] for client certificates, and shall support verification of the RSA client certificate and signature.

A client should support client authentication. If client authentication is supported, the client shall support RSA client certificate and signature and shall use an RSA key length of at least 1 024 bits.

The trusted CA bootstrapping mechanisms are outside the scope of the current document. Future versions of the document might define standardized bootstrapping mechanisms.

11.3 IEEE 802.1X

IEEE 802.1X is an IEEE standard for port based network access control for the purpose of providing authentication and authorization of the devices attached to LAN ports. It makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases in which the authentication and authorization process fails.

This document recommends the adoption of IEEE 802.1X for port based authentication for wireless networks. A device that supports IEEE 802.1X shall support EAP-PEAP/MSCHAPv2 type as a supported EAP method. The device may also support other EAP methods such as EAP-MD5, EAP-TLS and EAP-TTLS types.

This document defines a set of commands to configure and manage the IEEE 802.1X configuration, refer to 8.5.8.

Annex A
(informative)

Example for GetServices response with capabilities

The following is an example response for GetServices.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Header>
  </env:Header>
  <env:Body>
    <tds:GetServicesResponse>
      <tds:Service>
        <tds:Namespace>http://www.onvif.org/ver10/device/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif/device_service</tds:XAddr>
        <tds:Capabilities>
          <tds:Capabilities>
            <tds:Network IPFilter="false" ZeroConfiguration="true"
IPVersion6="false" DynDNS="false" Dot11Configuration="false"
HostnameFromDHCP="false" NTP="0" />
            <tds:Security TLS1.0="false" TLS1.1="false" TLS1.2="false"
OnboardKeyGeneration="false" AccessPolicyConfig="false"
DefaultAccessPolicy="false" Dot1X="false" RemoteUserHandling="false"
X.509Token="false" SAMLToken="false" KerberosToken="false"
UsernameToken="false" HttpDigest="false" RELToken="false" />
            <tds:System DiscoveryResolve="true" DiscoveryBye="true"
SystemBackup="false" SystemLogging="false" FirmwareUpgrade="true"
HttpFirmwareUpgrade="false" HttpSystemBackup="false" HttpSystemLogging="false"
HttpSupportInformation="false" />
            <tds:MiscCapabilities AuxiliaryCommands="" />
          </tds:Capabilities>
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
      <tds:Service>
        <tds:Namespace>http://www.onvif.org/ver10/events/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <tev:Capabilities xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
WSSubscriptionPolicySupport="false" WSPullPointSupport="false"
WSPausableSubscription="false" />
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
<tds:Namespace>http://www.onvif.org/ver10/deviceIO/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <tmd:Capabilities
xmlns:tmd="http://www.onvif.org/ver10/deviceIO/wsdl" RelayOutputs="2"
SerialPorts="0" DigitalInputs="4" />
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
```

```
    </tds:GetServicesResponse>
  </env:Body>
</env:Envelope>
```

Note that capabilities can be omitted if a device does not support the capability or a new capability is defined after the device implementation.

Annex B
(normative)

Device IP network liferface XML schemata

B.1 Device management service WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
targetNamespace="http://www.onvif.org/ver10/device/wsdl">
  <wsdl:types>
    <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/device/wsdl" elementFormDefault="qualified"
version="2.4.1">
      <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../../../ver10/schema/onvif.xsd"/>
      <!--================================-->
      <xs:element name="GetServices">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="IncludeCapability" type="xs:boolean"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServicesResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Service" type="tds:Service" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!--================================-->
      <xs:complexType name="Service">
        <xs:sequence>
          <xs:element name="Namespace" type="xs:anyURI"/>
          <xs:element name="XAddr" type="xs:anyURI"/>
          <xs:element name="Capabilities" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:any namespace="##any" processContents="lax"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Version" type="tt:OnvifVersion"/>
          <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <!--================================-->
      <xs:element name="GetServiceCapabilities">
        <xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServiceCapabilitiesResponse">
        <xs:complexType>
```

```
    <xs:sequence>
      <xs:element name="Capabilities" type="tds:DeviceServiceCapabilities"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:complexType name="DeviceServiceCapabilities">
  <xs:sequence>
    <xs:element name="Network" type="tds:NetworkCapabilities"/>
    <xs:element name="Security" type="tds:SecurityCapabilities"/>
    <xs:element name="System" type="tds:SystemCapabilities"/>
    <xs:element name="Misc" type="tds:MiscCapabilities" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Capabilities" type="tds:DeviceServiceCapabilities"/>
<!--================================-->
<xs:complexType name="NetworkCapabilities">
  <xs:attribute name="IPFilter" type="xs:boolean"/>
  <xs:attribute name="ZeroConfiguration" type="xs:boolean"/>
  <xs:attribute name="IPVersion6" type="xs:boolean"/>
  <xs:attribute name="DynDNS" type="xs:boolean"/>
  <xs:attribute name="Dot11Configuration" type="xs:boolean"/>
  <xs:attribute name="Dot1XConfigurations" type="xs:int"/>
  <xs:attribute name="HostnameFromDHCP" type="xs:boolean"/>
  <xs:attribute name="NTP" type="xs:int"/>
  <xs:attribute name="DHCPv6" type="xs:boolean"/>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:simpleType name="EAPMethodTypes">
  <xs:list itemType="xs:int"/>
</xs:simpleType>
<xs:complexType name="SecurityCapabilities">
  <xs:attribute name="TLS1.0" type="xs:boolean"/>
  <xs:attribute name="TLS1.1" type="xs:boolean"/>
  <xs:attribute name="TLS1.2" type="xs:boolean"/>
  <xs:attribute name="OnboardKeyGeneration" type="xs:boolean"/>
  <xs:attribute name="AccessPolicyConfig" type="xs:boolean"/>
  <xs:attribute name="DefaultAccessPolicy" type="xs:boolean"/>
  <xs:attribute name="Dot1X" type="xs:boolean"/>
  <xs:attribute name="RemoteUserHandling" type="xs:boolean"/>
  <xs:attribute name="X.509Token" type="xs:boolean"/>
  <xs:attribute name="SAMLToken" type="xs:boolean"/>
  <xs:attribute name="KerberosToken" type="xs:boolean"/>
  <xs:attribute name="UsernameToken" type="xs:boolean"/>
  <xs:attribute name="HttpDigest" type="xs:boolean"/>
  <xs:attribute name="RELToken" type="xs:boolean"/>
  <xs:attribute name="SupportedEAPMethods" type="tds:EAPMethodTypes"/>
  <xs:attribute name="MaxUsers" type="xs:int"/>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="SystemCapabilities">
  <xs:attribute name="DiscoveryResolve" type="xs:boolean"/>
  <xs:attribute name="DiscoveryBye" type="xs:boolean"/>
  <xs:attribute name="RemoteDiscovery" type="xs:boolean"/>
  <xs:attribute name="SystemBackup" type="xs:boolean"/>
  <xs:attribute name="SystemLogging" type="xs:boolean"/>
  <xs:attribute name="FirmwareUpgrade" type="xs:boolean"/>
  <xs:attribute name="HttpFirmwareUpgrade" type="xs:boolean"/>
  <xs:attribute name="HttpSystemBackup" type="xs:boolean"/>
  <xs:attribute name="HttpSystemLogging" type="xs:boolean"/>
```

```
  <xs:attribute name="HttpSupportInformation" type="xs:boolean"/>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="MiscCapabilities">
  <xs:attribute name="AuxiliaryCommands" type="tt:StringAttrList"/>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:element name="GetDeviceInformation">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <xs:element name="GetDeviceInformationResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Manufacturer" type="xs:string"/>
    <xs:element name="Model" type="xs:string"/>
    <xs:element name="FirmwareVersion" type="xs:string"/>
    <xs:element name="SerialNumber" type="xs:string"/>
    <xs:element name="HardwareId" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="SetSystemDateAndTime">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="DateTimeType" type="tt:SetDateTimeType"/>
    <xs:element name="DaylightSavings" type="xs:boolean"/>
    <xs:element name="TimeZone" type="tt:TimeZone" minOccurs="0"/>
    <xs:element name="UTCDateTime" type="tt:DateTime" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="SetSystemDateAndTimeResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="GetSystemDateAndTime">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <xs:element name="GetSystemDateAndTimeResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="SystemDateAndTime" type="tt:SystemDateTime"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="SetSystemFactoryDefault">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="FactoryDefault" type="tt:FactoryDefaultType"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

```
<xs:element name="SetSystemFactoryDefaultResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="UpgradeSystemFirmware">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Firmware" type="tt:AttachmentData"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="UpgradeSystemFirmwareResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Message" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SystemReboot">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<xs:element name="SystemRebootResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Message" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="RestoreSystem">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="BackupFiles" type="tt:BackupFile" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="RestoreSystemResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetSystemBackup">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<xs:element name="GetSystemBackupResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="BackupFiles" type="tt:BackupFile" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetSystemSupportInformation">
  <xs:complexType>
```

```
    <xs:sequence/>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetSystemSupportInformationResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="SupportInformation" type="tt:SupportInformation"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetSystemLog">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="LogType" type="tt:SystemLogType"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetSystemLogResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="SystemLog" type="tt:SystemLog"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetScopes">
   <xs:complexType>
    <xs:sequence/>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetScopesResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Scopes" type="tt:Scope" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetScopes">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Scopes" type="xs:anyURI" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="SetScopesResponse">
   <xs:complexType>
    <xs:sequence/>
   </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="AddScopes">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="ScopeItem" type="xs:anyURI" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="AddScopesResponse">
   <xs:complexType>
    <xs:sequence/>
```

```
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="RemoveScopes">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ScopeItem" type="xs:anyURI" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="RemoveScopesResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ScopeItem" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetDiscoveryMode">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetDiscoveryModeResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DiscoveryMode" type="tt:DiscoveryMode"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetDiscoveryMode">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DiscoveryMode" type="tt:DiscoveryMode"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetDiscoveryModeResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetEndpointReference">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetEndpointReferenceResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="GUID" type="xs:string"/>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetRemoteUser">
    <xs:complexType>
      <xs:sequence/>
```

```
  </xs:complexType>
 </xs:element>
 <xs:element name="GetRemoteUserResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="RemoteUser" type="tt:RemoteUser" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="SetRemoteUser">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="RemoteUser" type="tt:RemoteUser" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="SetRemoteUserResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="GetUsers">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <xs:element name="GetUsersResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="User" type="tt:User" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="CreateUsers">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="User" type="tt:User" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="CreateUsersResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
 <xs:element name="DeleteUsers">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Username" type="xs:string" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="DeleteUsersResponse">
  <xs:complexType>
   <xs:sequence/>
  </xs:complexType>
 </xs:element>
 <!--================================-->
```

```
<xs:element name="SetUser">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="User" type="tt:User" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetUserResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetWsdlUrl">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<xs:element name="GetWsdlUrlResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="WsdlUrl" type="xs:anyURI"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetHostname">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<xs:element name="GetHostnameResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="HostnameInformation" type="tt:HostnameInformation"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetHostname">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:token"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetHostnameResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetHostnameFromDHCP">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FromDHCP" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetHostnameFromDHCPResponse">
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element name="RebootNeeded" type="xs:boolean"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetDNS">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetDNSResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DNSInformation" type="tt:DNSInformation"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetDNS">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FromDHCP" type="xs:boolean"/>
        <xs:element name="SearchDomain" type="xs:token" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="DNSManual" type="tt:IPAddress" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetDNSResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetNTP">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetNTPResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NTPInformation" type="tt:NTPInformation"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetNTP">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="FromDHCP" type="xs:boolean"/>
        <xs:element name="NTPManual" type="tt:NetworkHost" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetNTPResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
```

```
<!--================================-->
<xs:element name="GetDynamicDNS">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
</xs:element>
<xs:element name="GetDynamicDNSResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="DynamicDNSInformation" type="tt:DynamicDNSInformation"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetDynamicDNS">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Type" type="tt:DynamicDNSType"/>
   <xs:element name="Name" type="tt:DNSName" minOccurs="0"/>
   <xs:element name="TTL" type="xs:duration" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="SetDynamicDNSResponse">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetNetworkInterfaces">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
</xs:element>
<xs:element name="GetNetworkInterfacesResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="NetworkInterfaces" type="tt:NetworkInterface" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetNetworkInterfaces">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="InterfaceToken" type="tt:ReferenceToken"/>
   <xs:element name="NetworkInterface" type="tt:NetworkInterfaceSetConfiguration"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="SetNetworkInterfacesResponse">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="RebootNeeded" type="xs:boolean"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetNetworkProtocols">
 <xs:complexType>
  <xs:sequence/>
 </xs:complexType>
```

```
    </xs:element>
    <xs:element name="GetNetworkProtocolsResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NetworkProtocols" type="tt:NetworkProtocol" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetNetworkProtocols">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NetworkProtocols" type="tt:NetworkProtocol" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetNetworkProtocolsResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetNetworkDefaultGateway">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetNetworkDefaultGatewayResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NetworkGateway" type="tt:NetworkGateway"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetNetworkDefaultGateway">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="IPv4Address" type="tt:IPv4Address" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="IPv6Address" type="tt:IPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetNetworkDefaultGatewayResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetZeroConfiguration">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetZeroConfigurationResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ZeroConfiguration" type="tt:NetworkZeroConfiguration"/>
        </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetZeroConfiguration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="InterfaceToken" type="tt:ReferenceToken"/>
      <xs:element name="Enabled" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetZeroConfigurationResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetIPAddressFilter">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<xs:element name="GetIPAddressFilterResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IPAddressFilter" type="tt:IPAddressFilter"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetIPAddressFilter">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IPAddressFilter" type="tt:IPAddressFilter"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetIPAddressFilterResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="AddIPAddressFilter">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IPAddressFilter" type="tt:IPAddressFilter"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="AddIPAddressFilterResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="RemoveIPAddressFilter">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IPAddressFilter" type="tt:IPAddressFilter"/>
    </xs:sequence>
  </xs:complexType>
```

```
    </xs:element>
    <xs:element name="RemoveIPAddressFilterResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetAccessPolicy">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetAccessPolicyResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="PolicyFile" type="tt:BinaryData"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetAccessPolicy">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="PolicyFile" type="tt:BinaryData"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetAccessPolicyResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="CreateCertificate">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateID" type="xs:token" minOccurs="0"/>
          <xs:element name="Subject" type="xs:string" minOccurs="0"/>
          <xs:element name="ValidNotBefore" type="xs:dateTime" minOccurs="0"/>
          <xs:element name="ValidNotAfter" type="xs:dateTime" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="CreateCertificateResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NvtCertificate" type="tt:Certificate"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetCertificates">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetCertificatesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NvtCertificate" type="tt:Certificate" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
```

```
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetCertificatesStatus">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetCertificatesStatusResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateStatus" type="tt:CertificateStatus" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetCertificatesStatus">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateStatus" type="tt:CertificateStatus" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetCertificatesStatusResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="DeleteCertificates">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateID" type="xs:token" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="DeleteCertificatesResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetPkcs10Request">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateID" type="xs:token"/>
          <xs:element name="Subject" type="xs:string" minOccurs="0"/>
          <xs:element name="Attributes" type="tt:BinaryData" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetPkcs10RequestResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Pkcs10Request" type="tt:BinaryData"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="LoadCertificates">
```

```
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NVTCertificate" type="tt:Certificate" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="LoadCertificatesResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetClientCertificateMode">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetClientCertificateModeResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Enabled" type="xs:boolean"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetClientCertificateMode">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Enabled" type="xs:boolean"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetClientCertificateModeResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetCACertificates">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetCACertificatesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CACertificate" type="tt:Certificate" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="LoadCertificateWithPrivateKey">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CertificateWithPrivateKey" type="tt:CertificateWithPrivateKey"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="LoadCertificateWithPrivateKeyResponse">
      <xs:complexType>
```

```
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetCertificateInformation">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CertificateID" type="xs:token"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="GetCertificateInformationResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CertificateInformation" type="tt:CertificateInformation"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="LoadCACertificates">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CACertificate" type="tt:Certificate" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="LoadCACertificatesResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="CreateDot1XConfiguration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Dot1XConfiguration" type="tt:Dot1XConfiguration"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="CreateDot1XConfigurationResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="SetDot1XConfiguration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Dot1XConfiguration" type="tt:Dot1XConfiguration"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="SetDot1XConfigurationResponse">
  <xs:complexType>
    <xs:sequence/>
  </xs:complexType>
</xs:element>
<!--================================-->
<xs:element name="GetDot1XConfiguration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Dot1XConfigurationToken" type="tt:ReferenceToken"/>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetDot1XConfigurationResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Dot1XConfiguration" type="tt:Dot1XConfiguration"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetDot1XConfigurations">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetDot1XConfigurationsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Dot1XConfiguration" type="tt:Dot1XConfiguration" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="DeleteDot1XConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Dot1XConfigurationToken" type="tt:ReferenceToken" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DeleteDot1XConfigurationResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetRelayOutputs">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetRelayOutputsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RelayOutputs" type="tt:RelayOutput" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetRelayOutputSettings">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RelayOutputToken" type="tt:ReferenceToken"/>
        <xs:element name="Properties" type="tt:RelayOutputSettings"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetRelayOutputSettingsResponse">
```

```
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetRelayOutputState">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="RelayOutputToken" type="tt:ReferenceToken"/>
          <xs:element name="LogicalState" type="tt:RelayLogicalState"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SetRelayOutputStateResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SendAuxiliaryCommand">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AuxiliaryCommand" type="tt:AuxiliaryData"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SendAuxiliaryCommandResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AuxiliaryCommandResponse" type="tt:AuxiliaryData" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDot11Capabilities">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetDot11CapabilitiesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Capabilities" type="tt:Dot11Capabilities"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetDot11Status">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="InterfaceToken" type="tt:ReferenceToken"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetDot11StatusResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Status" type="tt:Dot11Status"/>
        </xs:sequence>
```

```
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="ScanAvailableDot11Networks">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="InterfaceToken" type="tt:ReferenceToken"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="ScanAvailableDot11NetworksResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Networks" type="tt:Dot11AvailableNetworks" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetSystemUris">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetSystemUrisResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="SystemLogUris" type="tt:SystemLogUriList" minOccurs="0"/>
          <xs:element name="SupportInfoUri" type="xs:anyURI" minOccurs="0"/>
          <xs:element name="SystemBackupUri" type="xs:anyURI" minOccurs="0"/>
          <xs:element name="Extension" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="StartFirmwareUpgrade">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="StartFirmwareUpgradeResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="UploadUri" type="xs:anyURI"/>
          <xs:element name="UploadDelay" type="xs:duration"/>
          <xs:element name="ExpectedDownTime" type="xs:duration"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="StartSystemRestore">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
```

```
    <xs:element name="StartSystemRestoreResponse">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="UploadUri" type="xs:anyURI"/>
       <xs:element name="ExpectedDownTime" type="xs:duration"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
  </xs:schema>
</wsdl:types>
<wsdl:message name="GetServicesRequest">
  <wsdl:part name="parameters" element="tds:GetServices"/>
</wsdl:message>
<wsdl:message name="GetServicesResponse">
  <wsdl:part name="parameters" element="tds:GetServicesResponse"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesRequest">
  <wsdl:part name="parameters" element="tds:GetServiceCapabilities"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesResponse">
  <wsdl:part name="parameters" element="tds:GetServiceCapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="GetDeviceInformationRequest">
  <wsdl:part name="parameters" element="tds:GetDeviceInformation"/>
</wsdl:message>
<wsdl:message name="GetDeviceInformationResponse">
  <wsdl:part name="parameters" element="tds:GetDeviceInformationResponse"/>
</wsdl:message>
<wsdl:message name="SetSystemDateAndTimeRequest">
  <wsdl:part name="parameters" element="tds:SetSystemDateAndTime"/>
</wsdl:message>
<wsdl:message name="SetSystemDateAndTimeResponse">
  <wsdl:part name="parameters" element="tds:SetSystemDateAndTimeResponse"/>
</wsdl:message>
<wsdl:message name="GetSystemDateAndTimeRequest">
  <wsdl:part name="parameters" element="tds:GetSystemDateAndTime"/>
</wsdl:message>
<wsdl:message name="GetSystemDateAndTimeResponse">
  <wsdl:part name="parameters" element="tds:GetSystemDateAndTimeResponse"/>
</wsdl:message>
<wsdl:message name="SetSystemFactoryDefaultRequest">
  <wsdl:part name="parameters" element="tds:SetSystemFactoryDefault"/>
</wsdl:message>
<wsdl:message name="SetSystemFactoryDefaultResponse">
  <wsdl:part name="parameters" element="tds:SetSystemFactoryDefaultResponse"/>
</wsdl:message>
<wsdl:message name="UpgradeSystemFirmwareRequest">
  <wsdl:part name="parameters" element="tds:UpgradeSystemFirmware"/>
</wsdl:message>
<wsdl:message name="UpgradeSystemFirmwareResponse">
  <wsdl:part name="parameters" element="tds:UpgradeSystemFirmwareResponse"/>
</wsdl:message>
<wsdl:message name="SystemRebootRequest">
  <wsdl:part name="parameters" element="tds:SystemReboot"/>
</wsdl:message>
<wsdl:message name="SystemRebootResponse">
  <wsdl:part name="parameters" element="tds:SystemRebootResponse"/>
</wsdl:message>
<wsdl:message name="GetSystemBackupRequest">
  <wsdl:part name="parameters" element="tds:GetSystemBackup"/>
</wsdl:message>
<wsdl:message name="GetSystemBackupResponse">
```

```
  <wsdl:part name="parameters" element="tds:GetSystemBackupResponse"/>
 </wsdl:message>
 <wsdl:message name="RestoreSystemRequest">
  <wsdl:part name="parameters" element="tds:RestoreSystem"/>
 </wsdl:message>
 <wsdl:message name="RestoreSystemResponse">
  <wsdl:part name="parameters" element="tds:RestoreSystemResponse"/>
 </wsdl:message>
 <wsdl:message name="GetSystemSupportInformationRequest">
  <wsdl:part name="parameters" element="tds:GetSystemSupportInformation"/>
 </wsdl:message>
 <wsdl:message name="GetSystemSupportInformationResponse">
  <wsdl:part name="parameters" element="tds:GetSystemSupportInformationResponse"/>
 </wsdl:message>
 <wsdl:message name="GetSystemLogRequest">
  <wsdl:part name="parameters" element="tds:GetSystemLog"/>
 </wsdl:message>
 <wsdl:message name="GetSystemLogResponse">
  <wsdl:part name="parameters" element="tds:GetSystemLogResponse"/>
 </wsdl:message>
 <wsdl:message name="GetScopesRequest">
  <wsdl:part name="parameters" element="tds:GetScopes"/>
 </wsdl:message>
 <wsdl:message name="GetScopesResponse">
  <wsdl:part name="parameters" element="tds:GetScopesResponse"/>
 </wsdl:message>
 <wsdl:message name="SetScopesRequest">
  <wsdl:part name="parameters" element="tds:SetScopes"/>
 </wsdl:message>
 <wsdl:message name="SetScopesResponse">
  <wsdl:part name="parameters" element="tds:SetScopesResponse"/>
 </wsdl:message>
 <wsdl:message name="AddScopesRequest">
  <wsdl:part name="parameters" element="tds:AddScopes"/>
 </wsdl:message>
 <wsdl:message name="AddScopesResponse">
  <wsdl:part name="parameters" element="tds:AddScopesResponse"/>
 </wsdl:message>
 <wsdl:message name="RemoveScopesRequest">
  <wsdl:part name="parameters" element="tds:RemoveScopes"/>
 </wsdl:message>
 <wsdl:message name="RemoveScopesResponse">
  <wsdl:part name="parameters" element="tds:RemoveScopesResponse"/>
 </wsdl:message>
 <wsdl:message name="GetDiscoveryModeRequest">
  <wsdl:part name="parameters" element="tds:GetDiscoveryMode"/>
 </wsdl:message>
 <wsdl:message name="GetDiscoveryModeResponse">
  <wsdl:part name="parameters" element="tds:GetDiscoveryModeResponse"/>
 </wsdl:message>
 <wsdl:message name="SetDiscoveryModeRequest">
  <wsdl:part name="parameters" element="tds:SetDiscoveryMode"/>
 </wsdl:message>
 <wsdl:message name="SetDiscoveryModeResponse">
  <wsdl:part name="parameters" element="tds:SetDiscoveryModeResponse"/>
 </wsdl:message>
 <wsdl:message name="GetEndpointReferenceRequest">
  <wsdl:part name="parameters" element="tds:GetEndpointReference"/>
 </wsdl:message>
 <wsdl:message name="GetEndpointReferenceResponse">
  <wsdl:part name="parameters" element="tds:GetEndpointReferenceResponse"/>
 </wsdl:message>
```

```
<wsdl:message name="GetRemoteUserRequest">
  <wsdl:part name="parameters" element="tds:GetRemoteUser"/>
</wsdl:message>
<wsdl:message name="GetRemoteUserResponse">
  <wsdl:part name="parameters" element="tds:GetRemoteUserResponse"/>
</wsdl:message>
<wsdl:message name="SetRemoteUserRequest">
  <wsdl:part name="parameters" element="tds:SetRemoteUser"/>
</wsdl:message>
<wsdl:message name="SetRemoteUserResponse">
  <wsdl:part name="parameters" element="tds:SetRemoteUserResponse"/>
</wsdl:message>
<wsdl:message name="GetUsersRequest">
  <wsdl:part name="parameters" element="tds:GetUsers"/>
</wsdl:message>
<wsdl:message name="GetUsersResponse">
  <wsdl:part name="parameters" element="tds:GetUsersResponse"/>
</wsdl:message>
<wsdl:message name="CreateUsersRequest">
  <wsdl:part name="parameters" element="tds:CreateUsers"/>
</wsdl:message>
<wsdl:message name="CreateUsersResponse">
  <wsdl:part name="parameters" element="tds:CreateUsersResponse"/>
</wsdl:message>
<wsdl:message name="DeleteUsersRequest">
  <wsdl:part name="parameters" element="tds:DeleteUsers"/>
</wsdl:message>
<wsdl:message name="DeleteUsersResponse">
  <wsdl:part name="parameters" element="tds:DeleteUsersResponse"/>
</wsdl:message>
<wsdl:message name="SetUserRequest">
  <wsdl:part name="parameters" element="tds:SetUser"/>
</wsdl:message>
<wsdl:message name="SetUserResponse">
  <wsdl:part name="parameters" element="tds:SetUserResponse"/>
</wsdl:message>
<wsdl:message name="GetWsdlUrlRequest">
  <wsdl:part name="parameters" element="tds:GetWsdlUrl"/>
</wsdl:message>
<wsdl:message name="GetWsdlUrlResponse">
  <wsdl:part name="parameters" element="tds:GetWsdlUrlResponse"/>
</wsdl:message>
<wsdl:message name="GetHostnameRequest">
  <wsdl:part name="parameters" element="tds:GetHostname"/>
</wsdl:message>
<wsdl:message name="GetHostnameResponse">
  <wsdl:part name="parameters" element="tds:GetHostnameResponse"/>
</wsdl:message>
<wsdl:message name="SetHostnameRequest">
  <wsdl:part name="parameters" element="tds:SetHostname"/>
</wsdl:message>
<wsdl:message name="SetHostnameResponse">
  <wsdl:part name="parameters" element="tds:SetHostnameResponse"/>
</wsdl:message>
<wsdl:message name="SetHostnameFromDHCPRequest">
  <wsdl:part name="parameters" element="tds:SetHostnameFromDHCP"/>
</wsdl:message>
<wsdl:message name="SetHostnameFromDHCPResponse">
  <wsdl:part name="parameters" element="tds:SetHostnameFromDHCPResponse"/>
</wsdl:message>
<wsdl:message name="GetDNSRequest">
  <wsdl:part name="parameters" element="tds:GetDNS"/>
```

```
</wsdl:message>
<wsdl:message name="GetDNSResponse">
  <wsdl:part name="parameters" element="tds:GetDNSResponse"/>
</wsdl:message>
<wsdl:message name="SetDNSRequest">
  <wsdl:part name="parameters" element="tds:SetDNS"/>
</wsdl:message>
<wsdl:message name="SetDNSResponse">
  <wsdl:part name="parameters" element="tds:SetDNSResponse"/>
</wsdl:message>
<wsdl:message name="GetNTPRequest">
  <wsdl:part name="parameters" element="tds:GetNTP"/>
</wsdl:message>
<wsdl:message name="GetNTPResponse">
  <wsdl:part name="parameters" element="tds:GetNTPResponse"/>
</wsdl:message>
<wsdl:message name="SetNTPRequest">
  <wsdl:part name="parameters" element="tds:SetNTP"/>
</wsdl:message>
<wsdl:message name="SetNTPResponse">
  <wsdl:part name="parameters" element="tds:SetNTPResponse"/>
</wsdl:message>
<wsdl:message name="GetDynamicDNSRequest">
  <wsdl:part name="parameters" element="tds:GetDynamicDNS"/>
</wsdl:message>
<wsdl:message name="GetDynamicDNSResponse">
  <wsdl:part name="parameters" element="tds:GetDynamicDNSResponse"/>
</wsdl:message>
<wsdl:message name="SetDynamicDNSRequest">
  <wsdl:part name="parameters" element="tds:SetDynamicDNS"/>
</wsdl:message>
<wsdl:message name="SetDynamicDNSResponse">
  <wsdl:part name="parameters" element="tds:SetDynamicDNSResponse"/>
</wsdl:message>
<wsdl:message name="GetNetworkInterfacesRequest">
  <wsdl:part name="parameters" element="tds:GetNetworkInterfaces"/>
</wsdl:message>
<wsdl:message name="GetNetworkInterfacesResponse">
  <wsdl:part name="parameters" element="tds:GetNetworkInterfacesResponse"/>
</wsdl:message>
<wsdl:message name="SetNetworkInterfacesRequest">
  <wsdl:part name="parameters" element="tds:SetNetworkInterfaces"/>
</wsdl:message>
<wsdl:message name="SetNetworkInterfacesResponse">
  <wsdl:part name="parameters" element="tds:SetNetworkInterfacesResponse"/>
</wsdl:message>
<wsdl:message name="GetNetworkProtocolsRequest">
  <wsdl:part name="parameters" element="tds:GetNetworkProtocols"/>
</wsdl:message>
<wsdl:message name="GetNetworkProtocolsResponse">
  <wsdl:part name="parameters" element="tds:GetNetworkProtocolsResponse"/>
</wsdl:message>
<wsdl:message name="SetNetworkProtocolsRequest">
  <wsdl:part name="parameters" element="tds:SetNetworkProtocols"/>
</wsdl:message>
<wsdl:message name="SetNetworkProtocolsResponse">
  <wsdl:part name="parameters" element="tds:SetNetworkProtocolsResponse"/>
</wsdl:message>
<wsdl:message name="GetNetworkDefaultGatewayRequest">
  <wsdl:part name="parameters" element="tds:GetNetworkDefaultGateway"/>
</wsdl:message>
<wsdl:message name="GetNetworkDefaultGatewayResponse">
```

```
  <wsdl:part name="parameters" element="tds:GetNetworkDefaultGatewayResponse"/>
</wsdl:message>
<wsdl:message name="SetNetworkDefaultGatewayRequest">
  <wsdl:part name="parameters" element="tds:SetNetworkDefaultGateway"/>
</wsdl:message>
<wsdl:message name="SetNetworkDefaultGatewayResponse">
  <wsdl:part name="parameters" element="tds:SetNetworkDefaultGatewayResponse"/>
</wsdl:message>
<wsdl:message name="GetZeroConfigurationRequest">
  <wsdl:part name="parameters" element="tds:GetZeroConfiguration"/>
</wsdl:message>
<wsdl:message name="GetZeroConfigurationResponse">
  <wsdl:part name="parameters" element="tds:GetZeroConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="SetZeroConfigurationRequest">
  <wsdl:part name="parameters" element="tds:SetZeroConfiguration"/>
</wsdl:message>
<wsdl:message name="SetZeroConfigurationResponse">
  <wsdl:part name="parameters" element="tds:SetZeroConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetIPAddressFilterRequest">
  <wsdl:part name="parameters" element="tds:GetIPAddressFilter"/>
</wsdl:message>
<wsdl:message name="GetIPAddressFilterResponse">
  <wsdl:part name="parameters" element="tds:GetIPAddressFilterResponse"/>
</wsdl:message>
<wsdl:message name="SetIPAddressFilterRequest">
  <wsdl:part name="parameters" element="tds:SetIPAddressFilter"/>
</wsdl:message>
<wsdl:message name="SetIPAddressFilterResponse">
  <wsdl:part name="parameters" element="tds:SetIPAddressFilterResponse"/>
</wsdl:message>
<wsdl:message name="AddIPAddressFilterRequest">
  <wsdl:part name="parameters" element="tds:AddIPAddressFilter"/>
</wsdl:message>
<wsdl:message name="AddIPAddressFilterResponse">
  <wsdl:part name="parameters" element="tds:AddIPAddressFilterResponse"/>
</wsdl:message>
<wsdl:message name="RemoveIPAddressFilterRequest">
  <wsdl:part name="parameters" element="tds:RemoveIPAddressFilter"/>
</wsdl:message>
<wsdl:message name="RemoveIPAddressFilterResponse">
  <wsdl:part name="parameters" element="tds:RemoveIPAddressFilterResponse"/>
</wsdl:message>
<wsdl:message name="GetAccessPolicyRequest">
  <wsdl:part name="parameters" element="tds:GetAccessPolicy"/>
</wsdl:message>
<wsdl:message name="GetAccessPolicyResponse">
  <wsdl:part name="parameters" element="tds:GetAccessPolicyResponse"/>
</wsdl:message>
<wsdl:message name="SetAccessPolicyRequest">
  <wsdl:part name="parameters" element="tds:SetAccessPolicy"/>
</wsdl:message>
<wsdl:message name="SetAccessPolicyResponse">
  <wsdl:part name="parameters" element="tds:SetAccessPolicyResponse"/>
</wsdl:message>
<wsdl:message name="CreateCertificateRequest">
  <wsdl:part name="parameters" element="tds:CreateCertificate"/>
</wsdl:message>
<wsdl:message name="CreateCertificateResponse">
  <wsdl:part name="parameters" element="tds:CreateCertificateResponse"/>
</wsdl:message>
```

```
<wsdl:message name="GetCertificatesRequest">
  <wsdl:part name="parameters" element="tds:GetCertificates"/>
</wsdl:message>
<wsdl:message name="GetCertificatesResponse">
  <wsdl:part name="parameters" element="tds:GetCertificatesResponse"/>
</wsdl:message>
<wsdl:message name="GetCertificatesStatusRequest">
  <wsdl:part name="parameters" element="tds:GetCertificatesStatus"/>
</wsdl:message>
<wsdl:message name="GetCertificatesStatusResponse">
  <wsdl:part name="parameters" element="tds:GetCertificatesStatusResponse"/>
</wsdl:message>
<wsdl:message name="SetCertificatesStatusRequest">
  <wsdl:part name="parameters" element="tds:SetCertificatesStatus"/>
</wsdl:message>
<wsdl:message name="SetCertificatesStatusResponse">
  <wsdl:part name="parameters" element="tds:SetCertificatesStatusResponse"/>
</wsdl:message>
<wsdl:message name="DeleteCertificatesRequest">
  <wsdl:part name="parameters" element="tds:DeleteCertificates"/>
</wsdl:message>
<wsdl:message name="DeleteCertificatesResponse">
  <wsdl:part name="parameters" element="tds:DeleteCertificatesResponse"/>
</wsdl:message>
<wsdl:message name="GetPkcs10RequestRequest">
  <wsdl:part name="parameters" element="tds:GetPkcs10Request"/>
</wsdl:message>
<wsdl:message name="GetPkcs10RequestResponse">
  <wsdl:part name="parameters" element="tds:GetPkcs10RequestResponse"/>
</wsdl:message>
<wsdl:message name="LoadCertificatesRequest">
  <wsdl:part name="parameters" element="tds:LoadCertificates"/>
</wsdl:message>
<wsdl:message name="LoadCertificatesResponse">
  <wsdl:part name="parameters" element="tds:LoadCertificatesResponse"/>
</wsdl:message>
<wsdl:message name="GetClientCertificateModeRequest">
  <wsdl:part name="parameters" element="tds:GetClientCertificateMode"/>
</wsdl:message>
<wsdl:message name="GetClientCertificateModeResponse">
  <wsdl:part name="parameters" element="tds:GetClientCertificateModeResponse"/>
</wsdl:message>
<wsdl:message name="SetClientCertificateModeRequest">
  <wsdl:part name="parameters" element="tds:SetClientCertificateMode"/>
</wsdl:message>
<wsdl:message name="SetClientCertificateModeResponse">
  <wsdl:part name="parameters" element="tds:SetClientCertificateModeResponse"/>
</wsdl:message>
<wsdl:message name="SendAuxiliaryCommandRequest">
  <wsdl:part name="parameters" element="tds:SendAuxiliaryCommand"/>
</wsdl:message>
<wsdl:message name="SendAuxiliaryCommandResponse">
  <wsdl:part name="parameters" element="tds:SendAuxiliaryCommandResponse"/>
</wsdl:message>
<wsdl:message name="GetCACertificatesRequest">
  <wsdl:part name="parameters" element="tds:GetCACertificates"/>
</wsdl:message>
<wsdl:message name="GetCACertificatesResponse">
  <wsdl:part name="parameters" element="tds:GetCACertificatesResponse"/>
</wsdl:message>
<wsdl:message name="LoadCertificateWithPrivateKeyRequest">
  <wsdl:part name="parameters" element="tds:LoadCertificateWithPrivateKey"/>
```

```
</wsdl:message>
<wsdl:message name="LoadCertificateWithPrivateKeyResponse">
  <wsdl:part name="parameters" element="tds:LoadCertificateWithPrivateKeyResponse"/>
</wsdl:message>
<wsdl:message name="GetCertificateInformationRequest">
  <wsdl:part name="parameters" element="tds:GetCertificateInformation"/>
</wsdl:message>
<wsdl:message name="GetCertificateInformationResponse">
  <wsdl:part name="parameters" element="tds:GetCertificateInformationResponse"/>
</wsdl:message>
<wsdl:message name="LoadCACertificatesRequest">
  <wsdl:part name="parameters" element="tds:LoadCACertificates"/>
</wsdl:message>
<wsdl:message name="LoadCACertificatesResponse">
  <wsdl:part name="parameters" element="tds:LoadCACertificatesResponse"/>
</wsdl:message>
<wsdl:message name="CreateDot1XConfigurationRequest">
  <wsdl:part name="parameters" element="tds:CreateDot1XConfiguration"/>
</wsdl:message>
<wsdl:message name="CreateDot1XConfigurationResponse">
  <wsdl:part name="parameters" element="tds:CreateDot1XConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="SetDot1XConfigurationRequest">
  <wsdl:part name="parameters" element="tds:SetDot1XConfiguration"/>
</wsdl:message>
<wsdl:message name="SetDot1XConfigurationResponse">
  <wsdl:part name="parameters" element="tds:SetDot1XConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetDot1XConfigurationRequest">
  <wsdl:part name="parameters" element="tds:GetDot1XConfiguration"/>
</wsdl:message>
<wsdl:message name="GetDot1XConfigurationResponse">
  <wsdl:part name="parameters" element="tds:GetDot1XConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetDot1XConfigurationsRequest">
  <wsdl:part name="parameters" element="tds:GetDot1XConfigurations"/>
</wsdl:message>
<wsdl:message name="GetDot1XConfigurationsResponse">
  <wsdl:part name="parameters" element="tds:GetDot1XConfigurationsResponse"/>
</wsdl:message>
<wsdl:message name="DeleteDot1XConfigurationRequest">
  <wsdl:part name="parameters" element="tds:DeleteDot1XConfiguration"/>
</wsdl:message>
<wsdl:message name="DeleteDot1XConfigurationResponse">
  <wsdl:part name="parameters" element="tds:DeleteDot1XConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetDot11CapabilitiesRequest">
  <wsdl:part name="parameters" element="tds:GetDot11Capabilities"/>
</wsdl:message>
<wsdl:message name="GetDot11CapabilitiesResponse">
  <wsdl:part name="parameters" element="tds:GetDot11CapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="GetDot11StatusRequest">
  <wsdl:part name="parameters" element="tds:GetDot11Status"/>
</wsdl:message>
<wsdl:message name="GetDot11StatusResponse">
  <wsdl:part name="parameters" element="tds:GetDot11StatusResponse"/>
</wsdl:message>
<wsdl:message name="ScanAvailableDot11NetworksRequest">
  <wsdl:part name="parameters" element="tds:ScanAvailableDot11Networks"/>
</wsdl:message>
<wsdl:message name="ScanAvailableDot11NetworksResponse">
```

```
  <wsdl:part name="parameters" element="tds:ScanAvailableDot11NetworksResponse"/>
 </wsdl:message>
 <wsdl:message name="GetSystemUrisRequest">
  <wsdl:part name="parameters" element="tds:GetSystemUris"/>
 </wsdl:message>
 <wsdl:message name="GetSystemUrisResponse">
  <wsdl:part name="parameters" element="tds:GetSystemUrisResponse"/>
 </wsdl:message>
 <wsdl:message name="StartFirmwareUpgradeRequest">
  <wsdl:part name="parameters" element="tds:StartFirmwareUpgrade"/>
 </wsdl:message>
 <wsdl:message name="StartFirmwareUpgradeResponse">
  <wsdl:part name="parameters" element="tds:StartFirmwareUpgradeResponse"/>
 </wsdl:message>
 <wsdl:message name="StartSystemRestoreRequest">
  <wsdl:part name="parameters" element="tds:StartSystemRestore"/>
 </wsdl:message>
 <wsdl:message name="StartSystemRestoreResponse">
  <wsdl:part name="parameters" element="tds:StartSystemRestoreResponse"/>
 </wsdl:message>
 <wsdl:portType name="Device">
  <wsdl:operation name="GetServices">
   <wsdl:input message="tds:GetServicesRequest"/>
   <wsdl:output message="tds:GetServicesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetServiceCapabilities">
   <wsdl:input message="tds:GetServiceCapabilitiesRequest"/>
   <wsdl:output message="tds:GetServiceCapabilitiesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetDeviceInformation">
   <wsdl:input message="tds:GetDeviceInformationRequest"/>
   <wsdl:output message="tds:GetDeviceInformationResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SetSystemDateAndTime">
   <wsdl:input message="tds:SetSystemDateAndTimeRequest"/>
   <wsdl:output message="tds:SetSystemDateAndTimeResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSystemDateAndTime">
   <wsdl:input message="tds:GetSystemDateAndTimeRequest"/>
   <wsdl:output message="tds:GetSystemDateAndTimeResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SetSystemFactoryDefault">
   <wsdl:input message="tds:SetSystemFactoryDefaultRequest"/>
   <wsdl:output message="tds:SetSystemFactoryDefaultResponse"/>
  </wsdl:operation>
  <wsdl:operation name="UpgradeSystemFirmware">
   <wsdl:input message="tds:UpgradeSystemFirmwareRequest"/>
   <wsdl:output message="tds:UpgradeSystemFirmwareResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SystemReboot">
   <wsdl:input message="tds:SystemRebootRequest"/>
   <wsdl:output message="tds:SystemRebootResponse"/>
  </wsdl:operation>
  <wsdl:operation name="RestoreSystem">
   <wsdl:input message="tds:RestoreSystemRequest"/>
   <wsdl:output message="tds:RestoreSystemResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSystemBackup">
   <wsdl:input message="tds:GetSystemBackupRequest"/>
   <wsdl:output message="tds:GetSystemBackupResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSystemLog">
```

```
  <wsdl:input message="tds:GetSystemLogRequest"/>
  <wsdl:output message="tds:GetSystemLogResponse"/>
</wsdl:operation>
<wsdl:operation name="GetSystemSupportInformation">
  <wsdl:input message="tds:GetSystemSupportInformationRequest"/>
  <wsdl:output message="tds:GetSystemSupportInformationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetScopes">
  <wsdl:input message="tds:GetScopesRequest"/>
  <wsdl:output message="tds:GetScopesResponse"/>
</wsdl:operation>
<wsdl:operation name="SetScopes">
  <wsdl:input message="tds:SetScopesRequest"/>
  <wsdl:output message="tds:SetScopesResponse"/>
</wsdl:operation>
<wsdl:operation name="AddScopes">
  <wsdl:input message="tds:AddScopesRequest"/>
  <wsdl:output message="tds:AddScopesResponse"/>
</wsdl:operation>
<wsdl:operation name="RemoveScopes">
  <wsdl:input message="tds:RemoveScopesRequest"/>
  <wsdl:output message="tds:RemoveScopesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetDiscoveryMode">
  <wsdl:input message="tds:GetDiscoveryModeRequest"/>
  <wsdl:output message="tds:GetDiscoveryModeResponse"/>
</wsdl:operation>
<wsdl:operation name="SetDiscoveryMode">
  <wsdl:input message="tds:SetDiscoveryModeRequest"/>
  <wsdl:output message="tds:SetDiscoveryModeResponse"/>
</wsdl:operation>
<wsdl:operation name="GetEndpointReference">
  <wsdl:input message="tds:GetEndpointReferenceRequest"/>
  <wsdl:output message="tds:GetEndpointReferenceResponse"/>
</wsdl:operation>
<wsdl:operation name="GetRemoteUser">
  <wsdl:input message="tds:GetRemoteUserRequest"/>
  <wsdl:output message="tds:GetRemoteUserResponse"/>
</wsdl:operation>
<wsdl:operation name="SetRemoteUser">
  <wsdl:input message="tds:SetRemoteUserRequest"/>
  <wsdl:output message="tds:SetRemoteUserResponse"/>
</wsdl:operation>
<wsdl:operation name="GetUsers">
  <wsdl:input message="tds:GetUsersRequest"/>
  <wsdl:output message="tds:GetUsersResponse"/>
</wsdl:operation>
<wsdl:operation name="CreateUsers">
  <wsdl:input message="tds:CreateUsersRequest"/>
  <wsdl:output message="tds:CreateUsersResponse"/>
</wsdl:operation>
<wsdl:operation name="DeleteUsers">
  <wsdl:input message="tds:DeleteUsersRequest"/>
  <wsdl:output message="tds:DeleteUsersResponse"/>
</wsdl:operation>
<wsdl:operation name="SetUser">
  <wsdl:input message="tds:SetUserRequest"/>
  <wsdl:output message="tds:SetUserResponse"/>
</wsdl:operation>
<wsdl:operation name="GetWsdlUrl">
  <wsdl:input message="tds:GetWsdlUrlRequest"/>
  <wsdl:output message="tds:GetWsdlUrlResponse"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="GetHostname">
      <wsdl:input message="tds:GetHostnameRequest"/>
      <wsdl:output message="tds:GetHostnameResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetHostname">
      <wsdl:input message="tds:SetHostnameRequest"/>
      <wsdl:output message="tds:SetHostnameResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetHostnameFromDHCP">
      <wsdl:input message="tds:SetHostnameFromDHCPRequest"/>
      <wsdl:output message="tds:SetHostnameFromDHCPResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDNS">
      <wsdl:input message="tds:GetDNSRequest"/>
      <wsdl:output message="tds:GetDNSResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetDNS">
      <wsdl:input message="tds:SetDNSRequest"/>
      <wsdl:output message="tds:SetDNSResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetNTP">
      <wsdl:input message="tds:GetNTPRequest"/>
      <wsdl:output message="tds:GetNTPResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetNTP">
      <wsdl:input message="tds:SetNTPRequest"/>
      <wsdl:output message="tds:SetNTPResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDynamicDNS">
      <wsdl:input message="tds:GetDynamicDNSRequest"/>
      <wsdl:output message="tds:GetDynamicDNSResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetDynamicDNS">
      <wsdl:input message="tds:SetDynamicDNSRequest"/>
      <wsdl:output message="tds:SetDynamicDNSResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetNetworkInterfaces">
      <wsdl:input message="tds:GetNetworkInterfacesRequest"/>
      <wsdl:output message="tds:GetNetworkInterfacesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetNetworkInterfaces">
      <wsdl:input message="tds:SetNetworkInterfacesRequest"/>
      <wsdl:output message="tds:SetNetworkInterfacesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetNetworkProtocols">
      <wsdl:input message="tds:GetNetworkProtocolsRequest"/>
      <wsdl:output message="tds:GetNetworkProtocolsResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetNetworkProtocols">
      <wsdl:input message="tds:SetNetworkProtocolsRequest"/>
      <wsdl:output message="tds:SetNetworkProtocolsResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetNetworkDefaultGateway">
      <wsdl:input message="tds:GetNetworkDefaultGatewayRequest"/>
      <wsdl:output message="tds:GetNetworkDefaultGatewayResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetNetworkDefaultGateway">
      <wsdl:input message="tds:SetNetworkDefaultGatewayRequest"/>
      <wsdl:output message="tds:SetNetworkDefaultGatewayResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetZeroConfiguration">
```

```
  <wsdl:input message="tds:GetZeroConfigurationRequest"/>
  <wsdl:output message="tds:GetZeroConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="SetZeroConfiguration">
  <wsdl:input message="tds:SetZeroConfigurationRequest"/>
  <wsdl:output message="tds:SetZeroConfigurationResponse"/>
</wsdl:operation>
<wsdl:operation name="GetIPAddressFilter">
  <wsdl:input message="tds:GetIPAddressFilterRequest"/>
  <wsdl:output message="tds:GetIPAddressFilterResponse"/>
</wsdl:operation>
<wsdl:operation name="SetIPAddressFilter">
  <wsdl:input message="tds:SetIPAddressFilterRequest"/>
  <wsdl:output message="tds:SetIPAddressFilterResponse"/>
</wsdl:operation>
<wsdl:operation name="AddIPAddressFilter">
  <wsdl:input message="tds:AddIPAddressFilterRequest"/>
  <wsdl:output message="tds:AddIPAddressFilterResponse"/>
</wsdl:operation>
<wsdl:operation name="RemoveIPAddressFilter">
  <wsdl:input message="tds:RemoveIPAddressFilterRequest"/>
  <wsdl:output message="tds:RemoveIPAddressFilterResponse"/>
</wsdl:operation>
<wsdl:operation name="GetAccessPolicy">
  <wsdl:input message="tds:GetAccessPolicyRequest"/>
  <wsdl:output message="tds:GetAccessPolicyResponse"/>
</wsdl:operation>
<wsdl:operation name="SetAccessPolicy">
  <wsdl:input message="tds:SetAccessPolicyRequest"/>
  <wsdl:output message="tds:SetAccessPolicyResponse"/>
</wsdl:operation>
<wsdl:operation name="CreateCertificate">
  <wsdl:input message="tds:CreateCertificateRequest"/>
  <wsdl:output message="tds:CreateCertificateResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificates">
  <wsdl:input message="tds:GetCertificatesRequest"/>
  <wsdl:output message="tds:GetCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetCertificatesStatus">
  <wsdl:input message="tds:GetCertificatesStatusRequest"/>
  <wsdl:output message="tds:GetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="SetCertificatesStatus">
  <wsdl:input message="tds:SetCertificatesStatusRequest"/>
  <wsdl:output message="tds:SetCertificatesStatusResponse"/>
</wsdl:operation>
<wsdl:operation name="DeleteCertificates">
  <wsdl:input message="tds:DeleteCertificatesRequest"/>
  <wsdl:output message="tds:DeleteCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetPkcs10Request">
  <wsdl:input message="tds:GetPkcs10RequestRequest"/>
  <wsdl:output message="tds:GetPkcs10RequestResponse"/>
</wsdl:operation>
<wsdl:operation name="LoadCertificates">
  <wsdl:input message="tds:LoadCertificatesRequest"/>
  <wsdl:output message="tds:LoadCertificatesResponse"/>
</wsdl:operation>
<wsdl:operation name="GetClientCertificateMode">
  <wsdl:input message="tds:GetClientCertificateModeRequest"/>
  <wsdl:output message="tds:GetClientCertificateModeResponse"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="SetClientCertificateMode">
      <wsdl:input message="tds:SetClientCertificateModeRequest"/>
      <wsdl:output message="tds:SetClientCertificateModeResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SendAuxiliaryCommand">
      <wsdl:input message="tds:SendAuxiliaryCommandRequest"/>
      <wsdl:output message="tds:SendAuxiliaryCommandResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetCACertificates">
      <wsdl:input message="tds:GetCACertificatesRequest"/>
      <wsdl:output message="tds:GetCACertificatesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="LoadCertificateWithPrivateKey">
      <wsdl:input message="tds:LoadCertificateWithPrivateKeyRequest"/>
      <wsdl:output message="tds:LoadCertificateWithPrivateKeyResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetCertificateInformation">
      <wsdl:input message="tds:GetCertificateInformationRequest"/>
      <wsdl:output message="tds:GetCertificateInformationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="LoadCACertificates">
      <wsdl:input message="tds:LoadCACertificatesRequest"/>
      <wsdl:output message="tds:LoadCACertificatesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="CreateDot1XConfiguration">
      <wsdl:input message="tds:CreateDot1XConfigurationRequest"/>
      <wsdl:output message="tds:CreateDot1XConfigurationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetDot1XConfiguration">
      <wsdl:input message="tds:SetDot1XConfigurationRequest"/>
      <wsdl:output message="tds:SetDot1XConfigurationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDot1XConfiguration">
      <wsdl:input message="tds:GetDot1XConfigurationRequest"/>
      <wsdl:output message="tds:GetDot1XConfigurationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDot1XConfigurations">
      <wsdl:input message="tds:GetDot1XConfigurationsRequest"/>
      <wsdl:output message="tds:GetDot1XConfigurationsResponse"/>
    </wsdl:operation>
    <wsdl:operation name="DeleteDot1XConfiguration">
      <wsdl:input message="tds:DeleteDot1XConfigurationRequest"/>
      <wsdl:output message="tds:DeleteDot1XConfigurationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDot11Capabilities">
      <wsdl:input message="tds:GetDot11CapabilitiesRequest"/>
      <wsdl:output message="tds:GetDot11CapabilitiesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetDot11Status">
      <wsdl:input message="tds:GetDot11StatusRequest"/>
      <wsdl:output message="tds:GetDot11StatusResponse"/>
    </wsdl:operation>
    <wsdl:operation name="ScanAvailableDot11Networks">
      <wsdl:input message="tds:ScanAvailableDot11NetworksRequest"/>
      <wsdl:output message="tds:ScanAvailableDot11NetworksResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetSystemUris">
      <wsdl:input message="tds:GetSystemUrisRequest"/>
      <wsdl:output message="tds:GetSystemUrisResponse"/>
    </wsdl:operation>
    <wsdl:operation name="StartFirmwareUpgrade">
```

```
    <wsdl:input message="tds:StartFirmwareUpgradeRequest"/>
    <wsdl:output message="tds:StartFirmwareUpgradeResponse"/>
  </wsdl:operation>
  <wsdl:operation name="StartSystemRestore">
    <wsdl:input message="tds:StartSystemRestoreRequest"/>
    <wsdl:output message="tds:StartSystemRestoreResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="DeviceBinding" type="tds:Device">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GetServices">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetServices"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetServiceCapabilities">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetServiceCapabilities"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDeviceInformation">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDeviceInformation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetSystemDateAndTime">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetSystemDateAndTime"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetSystemDateAndTime">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetSystemDateAndTime"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetSystemFactoryDefault">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetSystemFactoryDefault"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
```

```
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="UpgradeSystemFirmware">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/UpgradeSystemFirmware"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="SystemReboot">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SystemReboot"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="RestoreSystem">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/RestoreSystem"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="GetSystemBackup">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetSystemBackup"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="GetSystemLog">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetSystemLog"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="GetSystemSupportInformation">
    <soap:operation
soapAction="http://www.onvif.org/ver10/device/wsdl/GetSystemSupportInformation"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal"/>
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="GetScopes">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetScopes"/>
    <wsdl:input>
     <soap:body use="literal"/>
    </wsdl:input>
```

```
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="SetScopes">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetScopes"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="AddScopes">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/AddScopes"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="RemoveScopes">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/RemoveScopes"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetDiscoveryMode">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDiscoveryMode"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="SetDiscoveryMode">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetDiscoveryMode"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetEndpointReference">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetEndpointReference"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetRemoteUser">
  <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetRemoteUser"/>
  <wsdl:input>
    <soap:body use="literal"/>
```

```
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetRemoteUser">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetRemoteUser"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetUsers">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetUsers"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="CreateUsers">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/CreateUsers"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="DeleteUsers">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/DeleteUsers"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetUser">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetUser"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetWsdlUrl">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetWsdlUrl"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetHostname">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetHostname"/>
    <wsdl:input>
```

```
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetHostname">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetHostname"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetHostnameFromDHCP">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetHostnameFromDHCP"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDNS">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDNS"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetDNS">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetDNS"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetNTP">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetNTP"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetNTP">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetNTP"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDynamicDNS">
   <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDynamicDNS"/>
```

```
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetDynamicDNS">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetDynamicDNS"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetNetworkInterfaces">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetNetworkInterfaces"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetNetworkInterfaces">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetNetworkInterfaces"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetNetworkProtocols">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetNetworkProtocols"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetNetworkProtocols">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetNetworkProtocols"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetNetworkDefaultGateway">
    <soap:operation
soapAction="http://www.onvif.org/ver10/device/wsdl/GetNetworkDefaultGateway"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
```

```
    <wsdl:operation name="SetNetworkDefaultGateway">
      <soap:operation
soapAction="http://www.onvif.org/ver10/device/wsdl/SetNetworkDefaultGateway"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetZeroConfiguration">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetZeroConfiguration"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetZeroConfiguration">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetZeroConfiguration"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetIPAddressFilter">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetIPAddressFilter"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetIPAddressFilter">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetIPAddressFilter"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AddIPAddressFilter">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/AddIPAddressFilter"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="RemoveIPAddressFilter">
      <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/RemoveIPAddressFilter"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
```

```
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetAccessPolicy">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetAccessPolicy"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetAccessPolicy">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetAccessPolicy"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="CreateCertificate">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/CreateCertificate"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetCertificates">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetCertificates"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetCertificatesStatus">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetCertificatesStatus"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetCertificatesStatus">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetCertificatesStatus"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="DeleteCertificates">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/DeleteCertificates"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
```

```
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetPkcs10Request">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetPkcs10Request"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="LoadCertificates">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/LoadCertificates"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetClientCertificateMode">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetClientCertificateMode"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetClientCertificateMode">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetClientCertificateMode"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SendAuxiliaryCommand">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SendAuxiliaryCommand"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetCACertificates">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetCACertificates"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="LoadCertificateWithPrivateKey">
    <soap:operation
soapAction="http://www.onvif.org/ver10/device/wsdl/LoadCertificateWithPrivateKey"/>
    <wsdl:input>
      <soap:body use="literal"/>
```

```
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetCertificateInformation">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetCertificateInformation"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="LoadCACertificates">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/LoadCACertificates"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="CreateDot1XConfiguration">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/CreateDot1XConfiguration"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetDot1XConfiguration">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/SetDot1XConfiguration"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDot1XConfiguration">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDot1XConfiguration"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDot1XConfigurations">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDot1XConfigurations"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="DeleteDot1XConfiguration">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/DeleteDot1XConfiguration"/>
    <wsdl:input>
```

```
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDot11Capabilities">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDot11Capabilities"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDot11Status">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetDot11Status"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ScanAvailableDot11Networks">
    <soap:operation
soapAction="http://www.onvif.org/ver10/device/wsdl/ScanAvailableDot11Networks"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetSystemUris">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/GetSystemUris"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="StartFirmwareUpgrade">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/StartFirmwareUpgrade"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="StartSystemRestore">
    <soap:operation soapAction="http://www.onvif.org/ver10/device/wsdl/StartSystemRestore"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

```
</wsdl:definitions>
```

B.2 Device IO service WSDL

The wsdl:definitions provided at the commencement of the following XML schema should be confirmed at the time of implementation to ensure correct operation

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
xmlns:tmd="http://www.onvif.org/ver10/deviceIO/wsdl"
targetNamespace="http://www.onvif.org/ver10/deviceIO/wsdl">
  <wsdl:import namespace="http://www.onvif.org/ver10/device/wsdl"
location="../ver10/device/wsdl/devicemgmt.wsdl"/>
  <wsdl:types>
    <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/deviceIO/wsdl" elementFormDefault="qualified"
version="2.4.1">
      <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../ver10/schema/onvif.xsd"/>
      <!--================================-->
      <xs:element name="GetServiceCapabilities">
        <xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServiceCapabilitiesResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Capabilities" type="tmd:Capabilities"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!--================================-->
      <xs:complexType name="Capabilities">
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="RelayOutputs" type="xs:int" default="0"/>
        <xs:attribute name="SerialPorts" type="xs:int" default="0"/>
        <xs:attribute name="DigitalInputs" type="xs:int" default="0"/>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:element name="Capabilities" type="tmd:Capabilities"/>
      <!--================================-->
      <xs:element name="GetRelayOutputOptions">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="RelayOutputToken" type="tt:ReferenceToken" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetRelayOutputOptionsResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="RelayOutputOptions" type="tmd:RelayOutputOptions" minOccurs="0"
maxOccurs="unbounded"/>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="RelayOutputOptions">
    <xs:sequence>
      <xs:element name="Mode" type="tt:RelayMode" maxOccurs="unbounded"/>
      <xs:element name="DelayTimes" type="tmd:DelayTimes" minOccurs="0"/>
      <xs:element name="Discrete" type="xs:boolean" minOccurs="0"/>
      <xs:element name="Extension" type="tmd:RelayOutputOptionsExtension" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="token" type="tt:ReferenceToken" use="required"/>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="RelayOutputOptionsExtension">
    <xs:sequence>
      <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="DelayTimes">
    <xs:list itemType="xs:float"/>
  </xs:simpleType>
  <!--================================-->
  <xs:element name="SetRelayOutputSettings">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RelayOutput" type="tt:RelayOutput"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetRelayOutputSettingsResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetDigitalInputs">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetDigitalInputsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DigitalInputs" type="tt:DigitalInput" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="GetSerialPorts">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetSerialPortsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SerialPort" type="tmd:SerialPort" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
```

```
    </xs:element>
    <!--================================-->
    <xs:element name="GetSerialPortConfiguration">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialPortToken" type="tt:ReferenceToken"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="GetSerialPortConfigurationResponse">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialPortConfiguration" type="tmd:SerialPortConfiguration"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SetSerialPortConfiguration">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialPortConfiguration" type="tmd:SerialPortConfiguration"/>
       <xs:element name="ForcePersistance" type="xs:boolean"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="SetSerialPortConfigurationResponse">
     <xs:complexType>
      <xs:sequence/>
     </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetSerialPortConfigurationOptions">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialPortToken" type="tt:ReferenceToken"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="GetSerialPortConfigurationOptionsResponse">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialPortOptions" type="tmd:SerialPortConfigurationOptions"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="SendReceiveSerialCommand">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialData" type="tmd:SerialData" minOccurs="0"/>
       <xs:element name="TimeOut" type="xs:duration" minOccurs="0"/>
       <xs:element name="DataLength" type="xs:integer" minOccurs="0"/>
       <xs:element name="Delimiter" type="xs:string" minOccurs="0"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="SendReceiveSerialCommandResponse">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="SerialData" type="tmd:SerialData" minOccurs="0"/>
      </xs:sequence>
     </xs:complexType>
```

```
    </xs:element>
    <!--================================-->
    <xs:complexType name="SerialData">
      <xs:choice>
        <xs:element name="Binary" type="xs:base64Binary"/>
        <xs:element name="String" type="xs:string"/>
      </xs:choice>
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
    <xs:complexType name="SerialPort">
      <xs:complexContent>
        <xs:extension base="tt:DeviceEntity">
          <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <!--================================-->
    <xs:simpleType name="SerialPortType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="RS232"/>
        <xs:enumeration value="RS422HalfDuplex"/>
        <xs:enumeration value="RS422FullDuplex"/>
        <xs:enumeration value="RS485HalfDuplex"/>
        <xs:enumeration value="RS485FullDuplex"/>
        <xs:enumeration value="Generic"/>
      </xs:restriction>
    </xs:simpleType>
    <!--================================-->
    <xs:complexType name="SerialPortConfiguration">
      <xs:sequence>
        <xs:element name="BaudRate" type="xs:int"/>
        <xs:element name="ParityBit" type="tmd:ParityBit"/>
        <xs:element name="CharacterLength" type="xs:int"/>
        <xs:element name="StopBit" type="xs:float"/>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="token" type="tt:ReferenceToken" use="required"/>
      <xs:attribute name="type" type="tmd:SerialPortType" use="required"/>
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
    <xs:simpleType name="ParityBit">
      <xs:restriction base="xs:string">
        <xs:enumeration value="None"/>
        <xs:enumeration value="Even"/>
        <xs:enumeration value="Odd"/>
        <xs:enumeration value="Mark"/>
        <xs:enumeration value="Space"/>
        <xs:enumeration value="Extended"/>
      </xs:restriction>
    </xs:simpleType>
    <!--================================-->
    <xs:complexType name="SerialPortConfigurationOptions">
      <xs:sequence>
        <xs:element name="BaudRateList" type="tt:IntList"/>
        <xs:element name="ParityBitList" type="tmd:ParityBitList"/>
```

```
        <xs:element name="CharacterLengthList" type="tt:IntList"/>
        <xs:element name="StopBitList" type="tt:FloatList"/>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="token" type="tt:ReferenceToken" use="required"/>
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
    <xs:complexType name="ParityBitList">
      <xs:sequence>
        <xs:element name="Items" type="tmd:ParityBit" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
 </wsdl:types>
 <wsdl:message name="GetServiceCapabilitiesRequest">
  <wsdl:part name="parameters" element="tmd:GetServiceCapabilities"/>
 </wsdl:message>
 <wsdl:message name="GetServiceCapabilitiesResponse">
  <wsdl:part name="parameters" element="tmd:GetServiceCapabilitiesResponse"/>
 </wsdl:message>
 <wsdl:message name="GetRelayOutputOptionsRequest">
  <wsdl:part name="parameters" element="tmd:GetRelayOutputOptions"/>
 </wsdl:message>
 <wsdl:message name="GetRelayOutputOptionsResponse">
  <wsdl:part name="parameters" element="tmd:GetRelayOutputOptionsResponse"/>
 </wsdl:message>
 <wsdl:message name="GetRelayOutputsRequest">
  <wsdl:part name="parameters" element="tds:GetRelayOutputs"/>
 </wsdl:message>
 <wsdl:message name="GetRelayOutputsResponse">
  <wsdl:part name="parameters" element="tds:GetRelayOutputsResponse"/>
 </wsdl:message>
 <wsdl:message name="SetRelayOutputSettingsRequest">
  <wsdl:part name="parameters" element="tmd:SetRelayOutputSettings"/>
 </wsdl:message>
 <wsdl:message name="SetRelayOutputSettingsResponse">
  <wsdl:part name="parameters" element="tmd:SetRelayOutputSettingsResponse"/>
 </wsdl:message>
 <wsdl:message name="SetRelayOutputStateRequest">
  <wsdl:part name="parameters" element="tds:SetRelayOutputState"/>
 </wsdl:message>
 <wsdl:message name="SetRelayOutputStateResponse">
  <wsdl:part name="parameters" element="tds:SetRelayOutputStateResponse"/>
 </wsdl:message>
 <wsdl:message name="GetDigitalInputsRequest">
  <wsdl:part name="parameters" element="tmd:GetDigitalInputs"/>
 </wsdl:message>
 <wsdl:message name="GetDigitalInputsResponse">
  <wsdl:part name="parameters" element="tmd:GetDigitalInputsResponse"/>
 </wsdl:message>
 <wsdl:message name="GetSerialPortsRequest">
  <wsdl:part name="parameters" element="tmd:GetSerialPorts"/>
 </wsdl:message>
 <wsdl:message name="GetSerialPortsResponse">
  <wsdl:part name="parameters" element="tmd:GetSerialPortsResponse"/>
 </wsdl:message>
 <wsdl:message name="GetSerialPortConfigurationRequest">
  <wsdl:part name="parameters" element="tmd:GetSerialPortConfiguration"/>
 </wsdl:message>
 <wsdl:message name="GetSerialPortConfigurationResponse">
```

```
  <wsdl:part name="parameters" element="tmd:GetSerialPortConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="SetSerialPortConfigurationRequest">
  <wsdl:part name="parameters" element="tmd:SetSerialPortConfiguration"/>
</wsdl:message>
<wsdl:message name="SetSerialPortConfigurationResponse">
  <wsdl:part name="parameters" element="tmd:SetSerialPortConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetSerialPortConfigurationOptionsRequest">
  <wsdl:part name="parameters" element="tmd:GetSerialPortConfigurationOptions"/>
</wsdl:message>
<wsdl:message name="GetSerialPortConfigurationOptionsResponse">
  <wsdl:part name="parameters" element="tmd:GetSerialPortConfigurationOptionsResponse"/>
</wsdl:message>
<wsdl:message name="SendReceiveSerialCommandRequest">
  <wsdl:part name="parameters" element="tmd:SendReceiveSerialCommand"/>
</wsdl:message>
<wsdl:message name="SendReceiveSerialCommandResponse">
  <wsdl:part name="parameters" element="tmd:SendReceiveSerialCommandResponse"/>
</wsdl:message>
<wsdl:portType name="DeviceIOPort">
  <wsdl:operation name="GetServiceCapabilities">
    <wsdl:input message="tmd:GetServiceCapabilitiesRequest"/>
    <wsdl:output message="tmd:GetServiceCapabilitiesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetRelayOutputOptions">
    <wsdl:input message="tmd:GetRelayOutputOptionsRequest"/>
    <wsdl:output message="tmd:GetRelayOutputOptionsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetRelayOutputs">
    <wsdl:input message="tmd:GetRelayOutputsRequest"/>
    <wsdl:output message="tmd:GetRelayOutputsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SetRelayOutputSettings">
    <wsdl:input message="tmd:SetRelayOutputSettingsRequest"/>
    <wsdl:output message="tmd:SetRelayOutputSettingsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SetRelayOutputState">
    <wsdl:input message="tmd:SetRelayOutputStateRequest"/>
    <wsdl:output message="tmd:SetRelayOutputStateResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetDigitalInputs">
    <wsdl:input message="tmd:GetDigitalInputsRequest"/>
    <wsdl:output message="tmd:GetDigitalInputsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSerialPorts">
    <wsdl:input message="tmd:GetSerialPortsRequest"/>
    <wsdl:output message="tmd:GetSerialPortsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSerialPortConfiguration">
    <wsdl:input message="tmd:GetSerialPortConfigurationRequest"/>
    <wsdl:output message="tmd:GetSerialPortConfigurationResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SetSerialPortConfiguration">
    <wsdl:input message="tmd:SetSerialPortConfigurationRequest"/>
    <wsdl:output message="tmd:SetSerialPortConfigurationResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetSerialPortConfigurationOptions">
    <wsdl:input message="tmd:GetSerialPortConfigurationOptionsRequest"/>
    <wsdl:output message="tmd:GetSerialPortConfigurationOptionsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="SendReceiveSerialCommand">
```

```
    <wsdl:input message="tmd:SendReceiveSerialCommandRequest"/>
    <wsdl:output message="tmd:SendReceiveSerialCommandResponse"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="DeviceIOBinding" type="tmd:DeviceIOPort">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GetServiceCapabilities">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetServiceCapabilities"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetRelayOutputOptions">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetRelayOutputOptions"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetRelayOutputs">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetRelayOutputs"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetRelayOutputSettings">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/SetRelayOutputSettings"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetRelayOutputState">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/SetRelayOutputState"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetDigitalInputs">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetDigitalInputs"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetSerialPorts">
   <soap:operation soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetSerialPorts"/>
```

```
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSerialPortConfiguration">
      <soap:operation
soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetSerialPortConfigurations"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetSerialPortConfiguration">
      <soap:operation
soapAction="http://www.onvif.org/ver10/deviceio/wsdl/SetSerialPortConfiguration"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetSerialPortConfigurationOptions">
      <soap:operation
soapAction="http://www.onvif.org/ver10/deviceio/wsdl/GetSerialPortConfigurationOptions"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SendReceiveSerialCommand">
      <soap:operation
soapAction="http://www.onvif.org/ver10/deviceio/wsdl/SendReceiveSerialCommand"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```


B.3 Event service WSDL

The wsdl:definitions provided at the commencement of the following XML schema should be confirmed at the time of implementation to ensure correct operation

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
```

```
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1" xmlns:wsntw="http://docs.oasis-open.org/wsn/bw-2"
xmlns:tev="http://www.onvif.org/ver10/events/wsdl" xmlns:wsrf-rw="http://docs.oasis-open.org/wsrf/rw-
2" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
targetNamespace="http://www.onvif.org/ver10/events/wsdl">
  <wsdl:import namespace="http://docs.oasis-open.org/wsn/bw-2" location="http://docs.oasis-
open.org/wsn/bw-2.wsdl"/>
  <wsdl:types>
    <xs:schema targetNamespace="http://www.onvif.org/ver10/events/wsdl"
elementFormDefault="qualified" version="2.4">
      <xs:import namespace="http://www.w3.org/2005/08/addressing"
schemaLocation="http://www.w3.org/2005/08/addressing/ws-addr.xsd"/>
      <xs:import namespace="http://docs.oasis-open.org/wsn/t-1" schemaLocation="http://docs.oasis-
open.org/wsn/t-1.xsd"/>
      <xs:import namespace="http://docs.oasis-open.org/wsn/b-2" schemaLocation="http://docs.oasis-
open.org/wsn/b-2.xsd"/>
      <!--  Message Request/Responses elements  -->
      <!--================================-->
      <xs:element name="GetServiceCapabilities">
        <xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServiceCapabilitiesResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Capabilities" type="tev:Capabilities"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!--================================-->
      <xs:complexType name="Capabilities">
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="WSSubscriptionPolicySupport" type="xs:boolean"/>
        <xs:attribute name="WSPullPointSupport" type="xs:boolean"/>
        <xs:attribute name="WSPausableSubscriptionManagerInterfaceSupport" type="xs:boolean"/>
        <xs:attribute name="MaxNotificationProducers" type="xs:int"/>
        <xs:attribute name="MaxPullPoints" type="xs:int"/>
        <xs:attribute name="PersistentNotificationStorage" type="xs:boolean"/>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:element name="Capabilities" type="tev:Capabilities"/>
      <!--================================-->
      <xs:element name="CreatePullPointSubscription">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Filter" type="wsnt:FilterType" minOccurs="0"/>
            <xs:element name="InitialTerminationTime" type="wsnt:AbsoluteOrRelativeTimeType"
nillable="true" minOccurs="0"/>
            <xs:element name="SubscriptionPolicy" minOccurs="0">
              <xs:complexType>
                <xs:sequence>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="CreatePullPointSubscriptionResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SubscriptionReference" type="wsa:EndpointReferenceType"/>
        <xs:element ref="wsnt:CurrentTime"/>
        <xs:element ref="wsnt:TerminationTime"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="PullMessages">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Timeout" type="xs:duration"/>
        <xs:element name="MessageLimit" type="xs:int"/>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="PullMessagesResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CurrentTime" type="xs:dateTime"/>
        <xs:element name="TerminationTime" type="xs:dateTime"/>
        <xs:element ref="wsnt:NotificationMessage" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="PullMessagesFaultResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MaxTimeout" type="xs:duration"/>
        <xs:element name="MaxMessageLimit" type="xs:int"/>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="Seek">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UtcTime" type="xs:dateTime"/>
        <xs:element name="Reverse" type="xs:boolean" minOccurs="0"/>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SeekResponse">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
  <!--================================-->
  <xs:element name="SetSynchronizationPoint">
    <xs:complexType>
      <xs:sequence/>
    </xs:complexType>
  </xs:element>
```

```
    <xs:element name="SetSynchronizationPointResponse">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:element name="GetEventProperties">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="GetEventPropertiesResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="TopicNamespaceLocation" type="xs:anyURI" maxOccurs="unbounded"/>
          <xs:element ref="wsnt:FixedTopicSet"/>
          <xs:element ref="wstop:TopicSet"/>
          <xs:element ref="wsnt:TopicExpressionDialect" maxOccurs="unbounded"/>
          <xs:element name="MessageContentFilterDialect" type="xs:anyURI"
maxOccurs="unbounded"/>
          <xs:element name="ProducerPropertiesFilterDialect" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="MessageContentSchemaLocation" type="xs:anyURI"
maxOccurs="unbounded"/>
          <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <!--================================-->
    <xs:complexType name="SubscriptionPolicy">
      <xs:sequence>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ChangedOnly" type="xs:boolean"/>
      <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <!--================================-->
  </xs:schema>
 </wsdl:types>
 <wsdl:message name="GetServiceCapabilitiesRequest">
  <wsdl:part name="parameters" element="tev:GetServiceCapabilities"/>
 </wsdl:message>
 <wsdl:message name="GetServiceCapabilitiesResponse">
  <wsdl:part name="parameters" element="tev:GetServiceCapabilitiesResponse"/>
 </wsdl:message>
 <wsdl:message name="CreatePullPointSubscriptionRequest">
  <wsdl:part name="parameters" element="tev:CreatePullPointSubscription"/>
 </wsdl:message>
 <wsdl:message name="CreatePullPointSubscriptionResponse">
  <wsdl:part name="parameters" element="tev:CreatePullPointSubscriptionResponse"/>
 </wsdl:message>
 <wsdl:message name="PullMessagesRequest">
  <wsdl:part name="parameters" element="tev:PullMessages"/>
 </wsdl:message>
 <wsdl:message name="PullMessagesResponse">
  <wsdl:part name="parameters" element="tev:PullMessagesResponse"/>
 </wsdl:message>
 <wsdl:message name="PullMessagesFaultResponse">
  <wsdl:part name="parameters" element="tev:PullMessagesFaultResponse"/>
 </wsdl:message>
 <wsdl:message name="SeekRequest">
```

```
  <wsdl:part name="parameters" element="tev:Seek"/>
 </wsdl:message>
 <wsdl:message name="SeekResponse">
  <wsdl:part name="parameters" element="tev:SeekResponse"/>
 </wsdl:message>
 <wsdl:message name="SetSynchronizationPointRequest">
  <wsdl:part name="parameters" element="tev:SetSynchronizationPoint"/>
 </wsdl:message>
 <wsdl:message name="SetSynchronizationPointResponse">
  <wsdl:part name="parameters" element="tev:SetSynchronizationPointResponse"/>
 </wsdl:message>
 <wsdl:message name="GetEventPropertiesRequest">
  <wsdl:part name="parameters" element="tev:GetEventProperties"/>
 </wsdl:message>
 <wsdl:message name="GetEventPropertiesResponse">
  <wsdl:part name="parameters" element="tev:GetEventPropertiesResponse"/>
 </wsdl:message>
 <wsdl:portType name="EventPortType">
  <wsdl:operation name="GetServiceCapabilities">
   <wsdl:input message="tev:GetServiceCapabilitiesRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilitiesRequest"/
>
   <wsdl:output message="tev:GetServiceCapabilitiesResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilitiesResponse
"/>
  </wsdl:operation>
  <wsdl:operation name="CreatePullPointSubscription">
   <wsdl:input message="tev:CreatePullPointSubscriptionRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionReq
uest"/>
   <wsdl:output message="tev:CreatePullPointSubscriptionResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionRes
ponse"/>
   <wsdl:fault name="ResourceUnknownFault" message="wsrf-rw:ResourceUnknownFault"/>
   <wsdl:fault name="InvalidFilterFault" message="wsntw:InvalidFilterFault"/>
   <wsdl:fault name="TopicExpressionDialectUnknownFault"
message="wsntw:TopicExpressionDialectUnknownFault"/>
   <wsdl:fault name="InvalidTopicExpressionFault" message="wsntw:InvalidTopicExpressionFault"/>
   <wsdl:fault name="TopicNotSupportedFault" message="wsntw:TopicNotSupportedFault"/>
   <wsdl:fault name="InvalidProducerPropertiesExpressionFault"
message="wsntw:InvalidProducerPropertiesExpressionFault"/>
   <wsdl:fault name="InvalidMessageContentExpressionFault"
message="wsntw:InvalidMessageContentExpressionFault"/>
   <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
message="wsntw:UnacceptableInitialTerminationTimeFault"/>
   <wsdl:fault name="UnrecognizedPolicyRequestFault"
message="wsntw:UnrecognizedPolicyRequestFault"/>
   <wsdl:fault name="UnsupportedPolicyRequestFault"
message="wsntw:UnsupportedPolicyRequestFault"/>
   <wsdl:fault name="NotifyMessageNotSupportedFault"
message="wsntw:NotifyMessageNotSupportedFault"/>
   <wsdl:fault name="SubscribeCreationFailedFault"
message="wsntw:SubscribeCreationFailedFault"/>
  </wsdl:operation>
  <wsdl:operation name="GetEventProperties">
   <wsdl:input message="tev:GetEventPropertiesRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest"/>
   <wsdl:output message="tev:GetEventPropertiesResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesResponse"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:portType name="PullPointSubscription">
```

```
    <wsdl:operation name="PullMessages">
      <wsdl:input message="tev:PullMessagesRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest"/>
      <wsdl:output message="tev:PullMessagesResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesResponse"/
>
      <wsdl:fault name="PullMessagesFaultResponse" message="tev:PullMessagesFaultResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessages/Fault/PullM
essagesFaultResponse"/>
    </wsdl:operation>
    <wsdl:operation name="Seek">
      <wsdl:input message="tev:SeekRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekRequest"/>
      <wsdl:output message="tev:SeekResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekResponse"/>
    </wsdl:operation>
    <wsdl:operation name="SetSynchronizationPoint">
      <wsdl:input message="tev:SetSynchronizationPointRequest"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointR
equest"/>
      <wsdl:output message="tev:SetSynchronizationPointResponse"
wsaw:Action="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointR
esponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="PullPointSubscriptionBinding" type="tev:PullPointSubscription">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="PullMessages">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="PullMessagesFaultResponse">
        <soap:fault name="PullMessagesFaultResponse" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Seek">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekRequest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetSynchronizationPoint">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointRe
quest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

```
  <wsdl:binding name="EventBinding" type="tev:EventPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetServiceCapabilities">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilities"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="CreatePullPointSubscription">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionRequ
est"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidFilterFault">
        <soap:fault name="InvalidFilterFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TopicExpressionDialectUnknownFault">
        <soap:fault name="TopicExpressionDialectUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidTopicExpressionFault">
        <soap:fault name="InvalidTopicExpressionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TopicNotSupportedFault">
        <soap:fault name="TopicNotSupportedFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidProducerPropertiesExpressionFault">
        <soap:fault name="InvalidProducerPropertiesExpressionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidMessageContentExpressionFault">
        <soap:fault name="InvalidMessageContentExpressionFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnacceptableInitialTerminationTimeFault">
        <soap:fault name="UnacceptableInitialTerminationTimeFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnrecognizedPolicyRequestFault">
        <soap:fault name="UnrecognizedPolicyRequestFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnsupportedPolicyRequestFault">
        <soap:fault name="UnsupportedPolicyRequestFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="NotifyMessageNotSupportedFault">
        <soap:fault name="NotifyMessageNotSupportedFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="SubscribeCreationFailedFault">
        <soap:fault name="SubscribeCreationFailedFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="GetEventProperties">
      <soap:operation
soapAction="http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest"/>
```

```
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="SubscriptionManagerBinding" type="wsntw:SubscriptionManager">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Renew">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/RenewRequest"/>
      <wsdl:input name="RenewRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="RenewResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnacceptableTerminationTimeFault">
        <soap:fault name="UnacceptableTerminationTimeFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Unsubscribe">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/UnsubscribeRequest"/>
      <wsdl:input name="UnsubscribeRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="UnsubscribeResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnableToDestroySubscriptionFault">
        <soap:fault name="UnableToDestroySubscriptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="NotificationProducerBinding" type="wsntw:NotificationProducer">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Subscribe">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="InvalidFilterFault">
        <soap:fault name="InvalidFilterFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TopicExpressionDialectUnknownFault">
        <soap:fault name="TopicExpressionDialectUnknownFault" use="literal"/>
```

```
    </wsdl:fault>
    <wsdl:fault name="InvalidTopicExpressionFault">
      <soap:fault name="InvalidTopicExpressionFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TopicNotSupportedFault">
      <soap:fault name="TopicNotSupportedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InvalidProducerPropertiesExpressionFault">
      <soap:fault name="InvalidProducerPropertiesExpressionFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InvalidMessageContentExpressionFault">
      <soap:fault name="InvalidMessageContentExpressionFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnacceptableInitialTerminationTimeFault">
      <soap:fault name="UnacceptableInitialTerminationTimeFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnrecognizedPolicyRequestFault">
      <soap:fault name="UnrecognizedPolicyRequestFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnsupportedPolicyRequestFault">
      <soap:fault name="UnsupportedPolicyRequestFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NotifyMessageNotSupportedFault">
      <soap:fault name="NotifyMessageNotSupportedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="SubscribeCreationFailedFault">
      <soap:fault name="SubscribeCreationFailedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="GetCurrentMessage">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/GetCurrentMessageRequest"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ResourceUnknownFault">
      <soap:fault name="ResourceUnknownFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TopicExpressionDialectUnknownFault">
      <soap:fault name="TopicExpressionDialectUnknownFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InvalidTopicExpressionFault">
      <soap:fault name="InvalidTopicExpressionFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TopicNotSupportedFault">
      <soap:fault name="TopicNotSupportedFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NoCurrentMessageOnTopicFault">
      <soap:fault name="NoCurrentMessageOnTopicFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="MultipleTopicsSpecifiedFault">
      <soap:fault name="MultipleTopicsSpecifiedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:binding name="NotificationConsumerBinding" type="wsntw:NotificationConsumer">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="Notify">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify"/>
```

```
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="PullPointBinding" type="wsntw:PullPoint">
   <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="GetMessages">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PullPoint/GetMessagesRequest"/>
    <wsdl:input name="GetMessagesRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="GetMessagesResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ResourceUnknownFault">
      <soap:fault name="ResourceUnknownFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnableToGetMessagesFault">
      <soap:fault name="UnableToGetMessagesFault" use="literal"/>
    </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="DestroyPullPoint">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PullPoint/DestroyPullPointRequest"/>
    <wsdl:input name="DestroyPullPointRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="DestroyPullPointResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ResourceUnknownFault">
      <soap:fault name="ResourceUnknownFault" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnableToDestroyPullPointFault">
      <soap:fault name="UnableToDestroyPullPointFault" use="literal"/>
    </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="Notify">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-2/PullPoint/Notify"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="CreatePullPointBinding" type="wsntw:CreatePullPoint">
   <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="CreatePullPoint">
    <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-2/CreatePullPoint
/CreatePullPointRequest"/>
    <wsdl:input name="CreatePullPointRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="CreatePullPointResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnableToCreatePullPointFault">
      <soap:fault name="UnableToCreatePullPointFault" use="literal"/>
    </wsdl:fault>
   </wsdl:operation>
  </wsdl:binding>
```

```
  <wsdl:binding name="PausableSubscriptionManagerBinding"
type="wsntw:PausableSubscriptionManager">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Renew">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PausableSubscriptionManager/RenewRequest"/>
      <wsdl:input name="RenewRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="RenewResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnacceptableTerminationTimeFault">
        <soap:fault name="UnacceptableTerminationTimeFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="Unsubscribe">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PausableSubscriptionManager/UnsubscribeRequest"/>
      <wsdl:input name="UnsubscribeRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="UnsubscribeResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnableToDestroySubscriptionFault">
        <soap:fault name="UnableToDestroySubscriptionFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="PauseSubscription">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PausableSubscriptionManager/PauseSubscriptionRequest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="PauseFailedFault">
        <soap:fault name="PauseFailedFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ResumeSubscription">
      <soap:operation soapAction="http://docs.oasis-open.org/wsn/bw-
2/PausableSubscriptionManager/ResumeSubscriptionRequest"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal"/>
```

```
    </wsdl:fault>
    <wsdl:fault name="ResumeFailedFault">
      <soap:fault name="ResumeFailedFault" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>
</wsdl:definitions>
```

B.4 Common schema

```
 <?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
targetNamespace="http://www.onvif.org/ver10/schema" elementFormDefault="qualified"
version="2.4.1">
  <xs:import namespace="http://www.w3.org/2005/05/xmlmime"
schemaLocation="http://www.w3.org/2005/05/xmlmime"/>
  <xs:import namespace="http://www.w3.org/2004/08/xop/include"
schemaLocation="http://www.w3.org/2004/08/xop/include"/>
  <!--================================-->
  <xs:complexType name="DeviceEntity">
    <xs:attribute name="token" type="tt:ReferenceToken" use="required"/>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="ReferenceToken">
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="Name">
    <xs:restriction base="xs:string">
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="IntList">
    <xs:sequence>
      <xs:element name="Items" type="xs:int" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="FloatList">
    <xs:sequence>
      <xs:element name="Items" type="xs:float" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="StringAttrList">
    <xs:list itemType="xs:string"/>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="ScopeDefinition">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Fixed"/>
      <xs:enumeration value="Configurable"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="Scope">
    <xs:sequence>
```

```
    <xs:element name="ScopeDef" type="tt:ScopeDefinition"/>
    <xs:element name="ScopeItem" type="xs:anyURI"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="DiscoveryMode">
   <xs:restriction base="xs:string">
    <xs:enumeration value="Discoverable"/>
    <xs:enumeration value="NonDiscoverable"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="NetworkInterface">
   <xs:complexContent>
    <xs:extension base="tt:DeviceEntity">
     <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:element name="Info" type="tt:NetworkInterfaceInfo" minOccurs="0"/>
      <xs:element name="Link" type="tt:NetworkInterfaceLink" minOccurs="0"/>
      <xs:element name="IPv4" type="tt:IPv4NetworkInterface" minOccurs="0"/>
      <xs:element name="IPv6" type="tt:IPv6NetworkInterface" minOccurs="0"/>
      <xs:element name="Extension" type="tt:NetworkInterfaceExtension" minOccurs="0"/>
     </xs:sequence>
     <xs:anyAttribute processContents="lax"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceExtension">
   <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="InterfaceType" type="tt:IANA-IfTypes"/>
    <xs:element name="Dot3" type="tt:Dot3Configuration" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Dot11" type="tt:Dot11Configuration" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="Extension" type="tt:NetworkInterfaceExtension2" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="NetworkInterfaceConfigPriority">
   <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="31"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="Dot3Configuration">
   <xs:sequence>
    <!-- Placeholder for 802.3 configuration -->
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceExtension2">
   <xs:sequence>
    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceLink">
```

```
    <xs:sequence>
      <xs:element name="AdminSettings" type="tt:NetworkInterfaceConnectionSetting"/>
      <xs:element name="OperSettings" type="tt:NetworkInterfaceConnectionSetting"/>
      <xs:element name="InterfaceType" type="tt:IANA-IfTypes"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceConnectionSetting">
    <xs:sequence>
      <xs:element name="AutoNegotiation" type="xs:boolean"/>
      <xs:element name="Speed" type="xs:int"/>
      <xs:element name="Duplex" type="tt:Duplex"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="Duplex">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Full"/>
      <xs:enumeration value="Half"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="IANA-IfTypes">
    <xs:restriction base="xs:int"/>
  </xs:simpleType>
  <xs:complexType name="NetworkInterfaceInfo">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="HwAddress" type="tt:HwAddress"/>
      <xs:element name="MTU" type="xs:int" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv6NetworkInterface">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:element name="Config" type="tt:IPv6Configuration" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv4NetworkInterface">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:element name="Config" type="tt:IPv4Configuration"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv4Configuration">
    <xs:sequence>
      <xs:element name="Manual" type="tt:PrefixedIPv4Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="LinkLocal" type="tt:PrefixedIPv4Address" minOccurs="0"/>
      <xs:element name="FromDHCP" type="tt:PrefixedIPv4Address" minOccurs="0"/>
      <xs:element name="DHCP" type="xs:boolean"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv6Configuration">
    <xs:sequence>
      <xs:element name="AcceptRouterAdvert" type="xs:boolean" minOccurs="0"/>
```

```
    <xs:element name="DHCP" type="tt:IPv6DHCPConfiguration"/>
    <xs:element name="Manual" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="LinkLocal" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="FromDHCP" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="FromRA" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="Extension" type="tt:IPv6ConfigurationExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="IPv6ConfigurationExtension">
  <xs:sequence>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:simpleType name="IPv6DHCPConfiguration">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Auto"/>
   <xs:enumeration value="Stateful"/>
   <xs:enumeration value="Stateless"/>
   <xs:enumeration value="Off"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="NetworkProtocol">
  <xs:sequence>
   <xs:element name="Name" type="tt:NetworkProtocolType"/>
   <xs:element name="Enabled" type="xs:boolean"/>
   <xs:element name="Port" type="xs:int" maxOccurs="unbounded"/>
   <xs:element name="Extension" type="tt:NetworkProtocolExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="NetworkProtocolExtension">
  <xs:sequence>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:simpleType name="NetworkProtocolType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="HTTP"/>
   <xs:enumeration value="HTTPS"/>
   <xs:enumeration value="RTSP"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:simpleType name="NetworkHostType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="IPv4"/>
   <xs:enumeration value="IPv6"/>
   <xs:enumeration value="DNS"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="NetworkHost">
  <xs:sequence>
```

```
    <xs:element name="Type" type="tt:NetworkHostType"/>
    <xs:element name="IPv4Address" type="tt:IPv4Address" minOccurs="0"/>
    <xs:element name="IPv6Address" type="tt:IPv6Address" minOccurs="0"/>
    <xs:element name="DNSname" type="tt:DNSName" minOccurs="0"/>
    <xs:element name="Extension" type="tt:NetworkHostExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="NetworkHostExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="IPAddress">
  <xs:sequence>
    <xs:element name="Type" type="tt:IPType"/>
    <xs:element name="IPv4Address" type="tt:IPv4Address" minOccurs="0"/>
    <xs:element name="IPv6Address" type="tt:IPv6Address" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="PrefixedIPv4Address">
  <xs:sequence>
    <xs:element name="Address" type="tt:IPv4Address"/>
    <xs:element name="PrefixLength" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:simpleType name="IPv4Address">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
<!--================================-->
<xs:complexType name="PrefixedIPv6Address">
  <xs:sequence>
    <xs:element name="Address" type="tt:IPv6Address"/>
    <xs:element name="PrefixLength" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:simpleType name="IPv6Address">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="HwAddress">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="IPType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="IPv4"/>
    <xs:enumeration value="IPv6"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="DNSName">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
<!--================================-->
<xs:complexType name="HostnameInformation">
  <xs:sequence>
```

```
    <xs:element name="FromDHCP" type="xs:boolean"/>
    <xs:element name="Name" type="xs:token" minOccurs="0"/>
    <xs:element name="Extension" type="tt:HostnameInformationExtension" minOccurs="0"/>
   </xs:sequence>
   <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="HostnameInformationExtension">
   <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DNSInformation">
   <xs:sequence>
    <xs:element name="FromDHCP" type="xs:boolean"/>
    <xs:element name="SearchDomain" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="DNSFromDHCP" type="tt:IPAddress" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="DNSManual" type="tt:IPAddress" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Extension" type="tt:DNSInformationExtension" minOccurs="0"/>
   </xs:sequence>
   <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DNSInformationExtension">
   <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NTPInformation">
   <xs:sequence>
    <xs:element name="FromDHCP" type="xs:boolean"/>
    <xs:element name="NTPFromDHCP" type="tt:NetworkHost" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="NTPManual" type="tt:NetworkHost" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="Extension" type="tt:NTPInformationExtension" minOccurs="0"/>
   </xs:sequence>
   <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NTPInformationExtension">
   <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="IPAddressFilterType">
   <xs:restriction base="xs:string">
    <xs:enumeration value="Allow"/>
    <xs:enumeration value="Deny"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="DynamicDNSInformation">
   <xs:sequence>
    <xs:element name="Type" type="tt:DynamicDNSType"/>
    <xs:element name="Name" type="tt:DNSName" minOccurs="0"/>
    <xs:element name="TTL" type="xs:duration" minOccurs="0"/>
    <xs:element name="Extension" type="tt:DynamicDNSInformationExtension" minOccurs="0"/>
```

```
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DynamicDNSInformationExtension">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="DynamicDNSType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="NoUpdate"/>
      <xs:enumeration value="ClientUpdates"/>
      <xs:enumeration value="ServerUpdates"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceSetConfiguration">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean" minOccurs="0"/>
      <xs:element name="Link" type="tt:NetworkInterfaceConnectionSetting" minOccurs="0"/>
      <xs:element name="MTU" type="xs:int" minOccurs="0"/>
      <xs:element name="IPv4" type="tt:IPv4NetworkInterfaceSetConfiguration" minOccurs="0"/>
      <xs:element name="IPv6" type="tt:IPv6NetworkInterfaceSetConfiguration" minOccurs="0"/>
      <xs:element name="Extension" type="tt:NetworkInterfaceSetConfigurationExtension"
minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkInterfaceSetConfigurationExtension">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Dot3" type="tt:Dot3Configuration" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Dot11" type="tt:Dot11Configuration" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Extension" type="tt:NetworkInterfaceSetConfigurationExtension2"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv6NetworkInterfaceSetConfiguration">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean" minOccurs="0"/>
      <xs:element name="AcceptRouterAdvert" type="xs:boolean" minOccurs="0"/>
      <xs:element name="Manual" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="DHCP" type="tt:IPv6DHCPConfiguration" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPv4NetworkInterfaceSetConfiguration">
    <xs:sequence>
      <xs:element name="Enabled" type="xs:boolean" minOccurs="0"/>
      <xs:element name="Manual" type="tt:PrefixedIPv4Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="DHCP" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkGateway">
```

```
    <xs:sequence>
      <xs:element name="IPv4Address" type="tt:IPv4Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="IPv6Address" type="tt:IPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkZeroConfiguration">
    <xs:sequence>
      <xs:element name="InterfaceToken" type="tt:ReferenceToken"/>
      <xs:element name="Enabled" type="xs:boolean"/>
      <xs:element name="Addresses" type="tt:IPv4Address" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Extension" type="tt:NetworkZeroConfigurationExtension" minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkZeroConfigurationExtension">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Additional" type="tt:NetworkZeroConfiguration" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Extension" type="tt:NetworkZeroConfigurationExtension2" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="NetworkZeroConfigurationExtension2">
    <xs:sequence>
      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPAddressFilter">
    <xs:sequence>
      <xs:element name="Type" type="tt:IPAddressFilterType"/>
      <xs:element name="IPv4Address" type="tt:PrefixedIPv4Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="IPv6Address" type="tt:PrefixedIPv6Address" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Extension" type="tt:IPAddressFilterExtension" minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="IPAddressFilterExtension">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="Dot11Configuration">
    <xs:sequence>
      <xs:element name="SSID" type="tt:Dot11SSIDType"/>
      <xs:element name="Mode" type="tt:Dot11StationMode"/>
      <xs:element name="Alias" type="tt:Name"/>
      <xs:element name="Priority" type="tt:NetworkInterfaceConfigPriority"/>
      <xs:element name="Security" type="tt:Dot11SecurityConfiguration"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
```

```
</xs:complexType>
<!--================================-->
<xs:simpleType name="Dot11SSIDType">
  <xs:restriction base="xs:hexBinary">
    <xs:minLength value="1"/>
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="Dot11StationMode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Ad-hoc"/>
    <xs:enumeration value="Infrastructure"/>
    <xs:enumeration value="Extended"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:complexType name="Dot11SecurityConfiguration">
  <xs:sequence>
    <xs:element name="Mode" type="tt:Dot11SecurityMode"/>
    <xs:element name="Algorithm" type="tt:Dot11Cipher" minOccurs="0"/>
    <xs:element name="PSK" type="tt:Dot11PSKSet" minOccurs="0"/>
    <xs:element name="Dot1X" type="tt:ReferenceToken" minOccurs="0"/>
    <xs:element name="Extension" type="tt:Dot11SecurityConfigurationExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="Dot11SecurityConfigurationExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:simpleType name="Dot11SecurityMode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="WEP"/>
    <xs:enumeration value="PSK"/>
    <xs:enumeration value="Dot1X"/>
    <xs:enumeration value="Extended"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="Dot11Cipher">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CCMP"/>
    <xs:enumeration value="TKIP"/>
    <xs:enumeration value="Any"/>
    <xs:enumeration value="Extended"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="Dot11PSK">
  <xs:restriction base="xs:hexBinary">
    <xs:length value="32"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="Dot11PSKPassphrase">
  <xs:restriction base="xs:string">
```

```
    <xs:pattern value="[ -~]{8,63}"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="Dot11PSKSet">
  <xs:sequence>
   <xs:element name="Key" type="tt:Dot11PSK" minOccurs="0"/>
   <xs:element name="Passphrase" type="tt:Dot11PSKPassphrase" minOccurs="0"/>
   <xs:element name="Extension" type="tt:Dot11PSKSetExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="Dot11PSKSetExtension">
  <xs:sequence>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="NetworkInterfaceSetConfigurationExtension2">
  <xs:sequence>
   <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="Dot11Capabilities">
  <xs:sequence>
   <xs:element name="TKIP" type="xs:boolean"/>
   <xs:element name="ScanAvailableNetworks" type="xs:boolean"/>
   <xs:element name="MultipleConfiguration" type="xs:boolean"/>
   <xs:element name="AdHocStationMode" type="xs:boolean"/>
   <xs:element name="WEP" type="xs:boolean"/>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:simpleType name="Dot11SignalStrength">
  <xs:restriction base="xs:string">
   <xs:enumeration value="None"/>
   <xs:enumeration value="Very Bad"/>
   <xs:enumeration value="Bad"/>
   <xs:enumeration value="Good"/>
   <xs:enumeration value="Very Good"/>
   <xs:enumeration value="Extended"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="Dot11Status">
  <xs:sequence>
   <xs:element name="SSID" type="tt:Dot11SSIDType"/>
   <xs:element name="BSSID" type="xs:string" minOccurs="0"/>
   <xs:element name="PairCipher" type="tt:Dot11Cipher" minOccurs="0"/>
   <xs:element name="GroupCipher" type="tt:Dot11Cipher" minOccurs="0"/>
   <xs:element name="SignalStrength" type="tt:Dot11SignalStrength" minOccurs="0"/>
   <xs:element name="ActiveConfigAlias" type="tt:ReferenceToken"/>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
```

```
  <xs:simpleType name="Dot11AuthAndMangementSuite">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None"/>
      <xs:enumeration value="Dot1X"/>
      <xs:enumeration value="PSK"/>
      <xs:enumeration value="Extended"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="Dot11AvailableNetworks">
    <xs:sequence>
      <xs:element name="SSID" type="tt:Dot11SSIDType"/>
      <xs:element name="BSSID" type="xs:string" minOccurs="0"/>
      <xs:element name="AuthAndMangementSuite" type="tt:Dot11AuthAndMangementSuite"
minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="PairCipher" type="tt:Dot11Cipher" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="GroupCipher" type="tt:Dot11Cipher" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="SignalStrength" type="tt:Dot11SignalStrength" minOccurs="0"/>
      <xs:element name="Extension" type="tt:Dot11AvailableNetworksExtension" minOccurs="0"/>
    </xs:sequence>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="Dot11AvailableNetworksExtension">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="OnvifVersion">
    <xs:sequence>
      <xs:element name="Major" type="xs:int"/>
      <xs:element name="Minor" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="SystemLogType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="System"/>
      <xs:enumeration value="Access"/>
    </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="SystemLog">
    <xs:sequence>
      <xs:element name="Binary" type="tt:AttachmentData" minOccurs="0"/>
      <xs:element name="String" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="SupportInformation">
    <xs:sequence>
      <xs:element name="Binary" type="tt:AttachmentData" minOccurs="0"/>
      <xs:element name="String" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="BinaryData">
    <xs:sequence>
      <xs:element name="Data" type="xs:base64Binary" nillable="false"/>
    </xs:sequence>
```

```
  <xs:attribute ref="xmime:contentType" use="optional"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="AttachmentData">
  <xs:sequence>
   <xs:element ref="xop:Include"/>
  </xs:sequence>
  <xs:attribute ref="xmime:contentType" use="optional"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="BackupFile">
  <xs:sequence>
   <xs:element name="Name" type="xs:string"/>
   <xs:element name="Data" type="tt:AttachmentData"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="SystemLogUriList">
  <xs:sequence>
   <xs:element name="SystemLog" type="tt:SystemLogUri" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="SystemLogUri">
  <xs:sequence>
   <xs:element name="Type" type="tt:SystemLogType"/>
   <xs:element name="Uri" type="xs:anyURI"/>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:simpleType name="FactoryDefaultType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Hard"/>
   <xs:enumeration value="Soft"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:simpleType name="SetDateTimeType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Manual"/>
   <xs:enumeration value="NTP"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="SystemDateTime">
  <xs:sequence>
   <xs:element name="DateTimeType" type="tt:SetDateTimeType"/>
   <xs:element name="DaylightSavings" type="xs:boolean"/>
   <xs:element name="TimeZone" type="tt:TimeZone" minOccurs="0"/>
   <xs:element name="UTCDateTime" type="tt:DateTime" minOccurs="0"/>
   <xs:element name="LocalDateTime" type="tt:DateTime" minOccurs="0"/>
   <xs:element name="Extension" type="tt:SystemDateTimeExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="SystemDateTimeExtension">
  <xs:sequence>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
```

```
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="DateTime">
  <xs:sequence>
   <xs:element name="Time" type="tt:Time"/>
   <xs:element name="Date" type="tt:Date"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="Date">
  <xs:sequence>
   <xs:element name="Year" type="xs:int"/>
   <xs:element name="Month" type="xs:int"/>
   <xs:element name="Day" type="xs:int"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="Time">
  <xs:sequence>
   <xs:element name="Hour" type="xs:int"/>
   <xs:element name="Minute" type="xs:int"/>
   <xs:element name="Second" type="xs:int"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="TimeZone">
  <xs:sequence>
   <xs:element name="TZ" type="xs:token"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="RemoteUser">
  <xs:sequence>
   <xs:element name="Username" type="xs:string"/>
   <xs:element name="Password" type="xs:string" minOccurs="0"/>
   <xs:element name="UseDerivedPassword" type="xs:boolean"/>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:simpleType name="UserLevel">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Administrator"/>
   <xs:enumeration value="Operator"/>
   <xs:enumeration value="User"/>
   <xs:enumeration value="Anonymous"/>
   <xs:enumeration value="Extended"/>
  </xs:restriction>
 </xs:simpleType>
 <!--================================-->
 <xs:complexType name="User">
  <xs:sequence>
   <xs:element name="Username" type="xs:string"/>
   <xs:element name="Password" type="xs:string" minOccurs="0"/>
   <xs:element name="UserLevel" type="tt:UserLevel"/>
   <xs:element name="Extension" type="tt:UserExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
```

```
<xs:complexType name="UserExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="Certificate">
  <xs:sequence>
    <xs:element name="CertificateID" type="xs:token"/>
    <xs:element name="Certificate" type="tt:BinaryData"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="CertificateStatus">
  <xs:sequence>
    <xs:element name="CertificateID" type="xs:token"/>
    <xs:element name="Status" type="xs:boolean"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="CertificateWithPrivateKey">
  <xs:sequence>
    <xs:element name="CertificateID" type="xs:token" minOccurs="0"/>
    <xs:element name="Certificate" type="tt:BinaryData"/>
    <xs:element name="PrivateKey" type="tt:BinaryData"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="CertificateInformation">
  <xs:sequence>
    <xs:element name="CertificateID" type="xs:token"/>
    <xs:element name="IssuerDN" type="xs:string" minOccurs="0"/>
    <xs:element name="SubjectDN" type="xs:string" minOccurs="0"/>
    <xs:element name="KeyUsage" type="tt:CertificateUsage" minOccurs="0"/>
    <xs:element name="ExtendedKeyUsage" type="tt:CertificateUsage" minOccurs="0"/>
    <xs:element name="KeyLength" type="xs:int" minOccurs="0"/>
    <xs:element name="Version" type="xs:string" minOccurs="0"/>
    <xs:element name="SerialNum" type="xs:string" minOccurs="0"/>
    <xs:element name="SignatureAlgorithm" type="xs:string" minOccurs="0"/>
    <xs:element name="Validity" type="tt:DateTimeRange" minOccurs="0"/>
    <xs:element name="Extension" type="tt:CertificateInformationExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="CertificateUsage">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Critical" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<!--================================-->
<xs:complexType name="CertificateInformationExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

```
<!--================================-->
<xs:complexType name="Dot1XConfiguration">
  <xs:sequence>
    <xs:element name="Dot1XConfigurationToken" type="tt:ReferenceToken"/>
    <xs:element name="Identity" type="xs:string"/>
    <xs:element name="AnonymousID" type="xs:string" minOccurs="0"/>
    <xs:element name="EAPMethod" type="xs:int"/>
    <xs:element name="CACertificateID" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="EAPMethodConfiguration" type="tt:EAPMethodConfiguration"
minOccurs="0"/>
    <xs:element name="Extension" type="tt:Dot1XConfigurationExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="Dot1XConfigurationExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="EAPMethodConfiguration">
  <xs:sequence>
    <xs:element name="TLSConfiguration" type="tt:TLSConfiguration" minOccurs="0"/>
    <xs:element name="Password" type="xs:string" minOccurs="0"/>
    <xs:element name="Extension" type="tt:EapMethodExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="EapMethodExtension">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="TLSConfiguration">
  <xs:sequence>
    <xs:element name="CertificateID" type="xs:token"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:simpleType name="RelayLogicalState">
  <xs:restriction base="xs:string">
    <xs:enumeration value="active"/>
    <xs:enumeration value="inactive"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:simpleType name="RelayIdleState">
  <xs:restriction base="xs:string">
    <xs:enumeration value="closed"/>
    <xs:enumeration value="open"/>
  </xs:restriction>
</xs:simpleType>
<!--================================-->
<xs:complexType name="RelayOutputSettings">
  <xs:sequence>
    <xs:element name="Mode" type="tt:RelayMode"/>
    <xs:element name="DelayTime" type="xs:duration"/>
```

```
    <xs:element name="IdleState" type="tt:RelayIdleState"/>
   </xs:sequence>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="RelayMode">
   <xs:restriction base="xs:string">
    <xs:enumeration value="Monostable"/>
    <xs:enumeration value="Bistable"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:complexType name="RelayOutput">
   <xs:complexContent>
    <xs:extension base="tt:DeviceEntity">
     <xs:sequence>
      <xs:element name="Properties" type="tt:RelayOutputSettings"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:anyAttribute processContents="lax"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:complexType name="DigitalInput">
   <xs:complexContent>
    <xs:extension base="tt:DeviceEntity">
     <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:anyAttribute processContents="lax"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <!--================================-->
  <xs:simpleType name="AuxiliaryData">
   <xs:restriction base="xs:string">
    <xs:maxLength value="128"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:simpleType name="PropertyOperation">
   <xs:restriction base="xs:string">
    <xs:enumeration value="Initialized"/>
    <xs:enumeration value="Deleted"/>
    <xs:enumeration value="Changed"/>
   </xs:restriction>
  </xs:simpleType>
  <!--================================-->
  <xs:element name="Message">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Source" type="tt:ItemList" minOccurs="0"/>
     <xs:element name="Key" type="tt:ItemList" minOccurs="0"/>
     <xs:element name="Data" type="tt:ItemList" minOccurs="0"/>
     <xs:element name="Extension" type="tt:MessageExtension" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="UtcTime" type="xs:dateTime" use="required"/>
    <xs:attribute name="PropertyOperation" type="tt:PropertyOperation"/>
    <xs:anyAttribute processContents="lax"/>
   </xs:complexType>
```

```
</xs:element>
<!--================================-->
<xs:complexType name="MessageExtension">
 <xs:sequence>
  <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="ItemList">
 <xs:sequence>
  <xs:element name="SimpleItem" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="Value" type="xs:anySimpleType" use="required"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="ElementItem" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:any namespace="##any"/>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:string" use="required"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Extension" type="tt:ItemListExtension" minOccurs="0"/>
 </xs:sequence>
 <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="ItemListExtension">
 <xs:sequence>
  <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="MessageDescription">
 <xs:sequence>
  <xs:element name="Source" type="tt:ItemListDescription" minOccurs="0"/>
  <xs:element name="Key" type="tt:ItemListDescription" minOccurs="0"/>
  <xs:element name="Data" type="tt:ItemListDescription" minOccurs="0"/>
  <xs:element name="Extension" type="tt:MessageDescriptionExtension" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="IsProperty" type="xs:boolean"/>
 <xs:anyAttribute processContents="lax"/>
</xs:complexType>
<!--================================-->
<xs:complexType name="MessageDescriptionExtension">
 <xs:sequence>
  <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<!--================================-->
<xs:complexType name="ItemListDescription">
 <xs:sequence>
  <xs:element name="SimpleItemDescription" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="Type" type="xs:QName" use="required"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="ElementItemDescription" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
```

```
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:QName" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="Extension" type="tt:ItemListDescriptionExtension" minOccurs="0"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="ItemListDescriptionExtension">
  <xs:sequence>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <!--================================-->
 <xs:complexType name="DateTimeRange">
  <xs:sequence>
   <xs:element name="From" type="xs:dateTime"/>
   <xs:element name="Until" type="xs:dateTime"/>
   <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
 </xs:complexType>
</xs:schema>
```

Bibliography

EAP-Registry, *Extensible Authentication Protocol (EAP) Registry*
<http://www.iana.org/assignments/eap-numbers/eap-numbers.xml>

IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee et al., August 1998*
<http://www.ietf.org/rfc/rfc2396.txt>

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*
<http://www.ietf.org/rfc/rfc2616.txt>

IETF RFC 4346, *The Transport Layer Security (TLS) Protocol Version 1.1*
<http://www.ietf.org/rfc/rfc4346.txt>

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*
<http://www.ietf.org/rfc/rfc5246.txt>

ONVIF Core specification, *ONVIF Core Specification Version 2.4 August, 2013*
<http://www.onvif.org/specs/core/ONVIF-Core-Specification-v240.pdf>

UDDI API ver2, *UDDI Version 2.04 API Specification UDDI Committee Specification, 19 July 2002, OASIS standard, 19 July 2002*
<http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf>

UDDI Data Structure ver2, *UDDI Version 2.03 Data Structure Reference UDDI Committee Specification, OASIS standard, 19 July 2002*
<http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.pdf>

W3C WSDL11, *Web Services Description Language (WSDL) 1.1*
<http://www.w3.org/TR/wsdl>

W3C XML-Schema 1, *W3C XML Schema Part 1: Structures Second Edition*
<http://www.w3.org/TR/xmlschema-1/>

W3C XML-Schema 2, *XML Schema Part 2: Datatypes Second Edition*
<http://www.w3.org/TR/xmlschema-2/>

W3C XOP, *XML-binary Optimized Packaging*
<http://www.w3.org/TR/2005/REC-xop10-20050125/>

W3C Xpath, *XML Path Language (XPath) Version 1.0*
<http://www.w3.org/TR/xpath/>

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*
<http://www.ietf.org/rfc/rfc3986.txt>

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

SYSTÈMES D'ALARME ET DE SÉCURITÉ ÉLECTRONIQUES –

Partie 11-31: Systèmes de contrôle d'accès électronique – Protocole de base d'interopérabilité en fonction des services Web

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – 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 l'IEC"). 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 l'IEC, participent également aux travaux. L'IEC 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 l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC 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 l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC 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 l'IEC. L'IEC 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 l'IEC, 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 l'IEC, 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 l'IEC ou de toute autre Publication de l'IEC, 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 l'IEC peuvent faire l'objet de droits de brevet. L'IEC 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 IEC 60839-11-31 a été établie par le comité d'études 79 de l'IEC: Systèmes d'alarme et de sécurité électroniques.

Le texte de cette norme est issu des documents suivants:

CDV	Rapport de vote
79/522/CDV	79/546/RVC

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/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 60839, publiées sous le titre général *Systèmes d'alarme et de sécurité électroniques*, peut être consultée sur le site web de l'IEC.

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 l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. À cette date, la publication sera

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

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

Le présent document a pour objet de fournir une base commune pour la mise en œuvre d'un réseau entièrement interopérable constitué de produits provenant de différents fournisseurs en réseau. Le présent document décrit le modèle de réseau, les interfaces, les types de données et les schémas d'échange de données. Le présent document réutilise les normes existantes pertinentes lorsqu'elles sont disponibles et présente de nouvelles spécifications, uniquement lorsque cela est nécessaire.

Le présent document a été élaboré sur la base des travaux réalisés par le Forum industriel ouvert ONVIF (Open Network Video Interface Forum). La spécification principale de l'ONVIF est compatible avec le présent document.

Le présent document s'accompagne d'un ensemble de définitions d'interfaces informatiques:

- WSDL de service de dispositif, voir l'Article B.1;
- WSDL de service d'E-S de dispositif, voir l'Article B.2;
- WSDL de service d'événement, voir l'Article B.3;
- Schéma commun, voir l'Article B.4.

Le présent document comporte les articles suivants:

Présentation du document: Donne une vue générale des différentes parties de la norme et leurs relations les unes aux autres.

Cadre d'application des services Web: Offre une brève présentation des services Web et de la base de services Web pour le présent document.

Configuration IP: Définit les exigences de configuration IP du réseau.

Découverte de dispositif: Décrit comment les dispositifs sont découverts dans les réseaux locaux et à distance.

Gestion de dispositif: Définit la configuration de notions élémentaires, comme les paramètres réseau et de sécurité.

Dispositif ES: Définit le traitement des ports d'entrée et de sortie sur un dispositif.

Traitement des événements: Définit comment s'abonner et recevoir des notifications (événements) à partir d'un dispositif.

Sécurité: Définit les exigences de sécurité au niveau du transport et du message.

SYSTÈMES D'ALARME ET DE SÉCURITÉ ÉLECTRONIQUES –

Partie 11-31: Systèmes de contrôle d'accès électronique – Protocole de base d'interopérabilité en fonction des services Web

1 Domaine d'application

La présente partie de l'IEC 60839 définit les procédures de communication entre les clients en réseau et les dispositifs. Cette série de normes relatives à l'interopérabilité permet de concevoir un système d'alarme et de sécurité électronique avec des clients et des dispositifs de différents fabricants utilisant des interfaces communes et bien définies. Les fonctions définies dans le présent document couvrent la découverte, la gestion de dispositif et le cadre des événements. Des services spécialisés supplémentaires sont définis dans des documents séparés.

Les interfaces de gestion et de contrôle définies dans le présent document sont décrites sous forme de services Web. Le présent document comprend également les définitions complètes du schéma XML et du langage de description de services Web (WSDL).

Afin d'offrir une interopérabilité complète et prête à l'emploi, le présent document définit les procédures de découverte de dispositif. Les mécanismes de découverte de dispositif du présent document reposent sur la spécification WS-Discovery, avec des extensions.

Le présent document n'empêche en aucune façon un fabricant d'ajouter un autre protocole ou d'étendre le protocole défini ici et de plus amples informations sur les règles d'ajout ou d'extension sont également données dans le présent document.

2 Références normatives

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences 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).

IEEE 1003.1, *The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition*
<http://pubs.opengroup.org/onlinepubs/009695399/>

IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*
<http://standards.ieee.org/getieee802/download/802.11-2007.pdf>

IEEE 802.1X, *Port-Based Network Access Control*
<http://standards.ieee.org/getieee802/download/802.1X-2004.pdf>

IETF RFC 952, Internet Host Table Specification
<https://tools.ietf.org/html/rfc952

IETF RFC 1123:1989, Requirements for Internet Hosts - Application and Support
<https://tools.ietf.org/html/rfc1123>

IETF RFC 2131, *Dynamic Host Configuration Protocol*
<http://www.ietf.org/rfc/rfc2131.txt>

IETF RFC 2136, *Dynamic Updates in the Domain Name System (DNS UPDATE)*
<http://www.ietf.org/rfc/rfc2136.txt>

IETF RFC 2246, *The TLS Protocol Version 1.0*
<http://www.ietf.org/rfc/rfc2246.txt>

IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*
<http://www.ietf.org/rfc/rfc2617.txt>

IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*
<http://www.ietf.org/rfc/rfc2986.txt>

IETF RFC 3268, *Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS)*
<http://www.ietf.org/rfc/rfc3268.txt>

IETF RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
<http://www.ietf.org/rfc/rfc3315.txt>

IETF RFC 3927, *Dynamic Configuration of IPv4 Link-Local Addresses*
<http://www.ietf.org/rfc/rfc3927.txt>

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*
<http://www.ietf.org/rfc/rfc4122.txt>

IETF RFC 4514, *Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names*
<http://www.ietf.org/rfc/rfc4514.txt>

IETF RFC 4702, *The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option*
<http://www.ietf.org/rfc/rfc4702.txt>

IETF RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*
<http://www.ietf.org/rfc/rfc4861.txt>

IETF RFC 4862, *IPv6 Stateless Address Auto configuration*
<http://www.ietf.org/rfc/rfc4862.txt>

ISO/IEC 8824-2, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): Spécification des objets informationnels*

ISO/IEC 8824-3, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): Spécification des contraintes*

ISO/IEC 8824-4, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): Paramétrage des spécifications de la notation de syntaxe abstraite numéro un*

ISO/IEC 8825-1, T*echnologies de l'information – Règles de codage ASN.1: Spécification des règles de codage de base (BER), des règles de codage canoniques (CER) et des règles de codage distinctives (DER)*

OASIS WS-BaseNotification, *Web Services Base Notification 1.3 (WS-BaseNotification)*
<http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf>

OASIS WS-Topics, *Web Services Topics 1.3 (WS-Topics)*
<http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf>

W3C SOAP-MTOM, *SOAP Message Transmission Optimization Mechanism*
<http://www.w3.org/TR/soap12-mtom/>

W3C SOAP12-PART1, *SOAP 1.2 Part 1, Messaging Framework*
<http://www.w3.org/TR/soap12-part1/>

W3C WS-Addressing, *Web Services Addressing 1.0 – Core*
<http://www.w3.org/TR/ws-addr-core/>

WS-I BP 2.0, *Basic Profile Version 2.0*
<http://www.ws-i.org/Profiles/BasicProfile-2.0-2010-11-09.html>

XMLSOAP WS-Discovery, *Web Services Dynamic Discovery (WS-Discovery)", J. Beatty et al., April 2005* .
<http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf>

3 Termes, définitions et termes abrégés

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
réseau ad hoc
ensemble de services de base indépendants

Note 1 à l'article: "Réseau ad hoc" est un terme vernaculaire.

3.1.2
ensemble de services de base
ensemble de stations IEEE 802.11 qui participent à un réseau commun

3.1.3
fonctionnalité
commande permettant à un client de demander les services offerts par un dispositif

3.1.4
normes de cryptographie à clé publique
PKCS
groupe de normes inventé et publié par RSA Security

Note 1 à l'article: Le terme abrégé «PKCS» est dérivé du terme anglais développé correspondant «public key cryptography standards».

3.1.5
clé prépartagée
clé statique distribuée au dispositif

3.1.6
pull point
point d'extraction
ressources permettant d'extraire des messages

Note 1 à l'article: Lors de l'extraction des messages, les notifications ne sont pas bloquées par les pare-feu.

3.1.7
identifiant d'ensemble de services
identité d'un réseau sans fil IEEE 802.11

[SOURCE: IEEE 802.11-2007]

3.1.8
accès protégé Wi-Fi
WPA
programme de certification créé par la Wi-Fi Alliance afin d'indiquer la conformité au protocole de sécurité couvert par le programme

Note 1 à l'article: Le terme abrégé «WPA» est dérivé du terme anglais développé correspondant «wi-fi protected access».

3.2 Termes abrégés

API	Application Programming Interface (Interface de programmation d'application)
ASCII	American Standard Code for Information Interchange (Code normalisé américain pour l'échange d'informations)
ASN	Abstract Syntax Notation (Notation de syntaxe abstraite)
BSSID	Basic Service Set Identification (Identifiant d'ensemble de services de base)
CA	Certificate Authority (Autorité de certification)
CBC	Cipher-Block Chaining (Chaînage de bloc de chiffrement)
CCMP	Counter mode with Cipher-block chaining Message authentication code Protocol (Protocole CCMP)
DER	Distinguished Encoding Rules (Règles de codage distinctives)
DHCP	Dynamic Host Configuration Protocol (Protocole DHCP)
DM	Device Management (Gestion de dispositif)
DNS	Domain Name Server (Serveur de noms de domaine)
DP	Discovery Proxy (Proxy de découverte)
EAP	Extensible Authentication Protocol (Protocole EAP)
GW	Gateway (Passerelle)
HMAC	Hash-based Message Authentication Code (Code d'authentification de message par hachage)
HTTP	Hypertext Transfer Protocol (Protocole HTTP)
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer (Protocole HTTPS)
IANA	Internet Assigned Numbers Authority (autorité chargée de la gestion de l'adressage sur Internet)
ES, E/S	Entrée/Sortie
IP	Internet Protocol (Protocole Internet)
IPv4	Internet Protocol Version 4 (Protocole Internet, Version 4)
IPv6	Internet Protocol Version 6 (Protocole Internet, Version 6)
LAN	Local Area Network (Réseau local)

MTOM	Message Transmission Optimization Mechanism (Mécanisme d'optimisation de la transmission de message)
NAT	Network Address Translation (Traduction d'adresse de réseau)
NFC	Near Field Communication (Communication en champ proche)
NTP	Network Time Protocol (Protocole NTP)
OASIS	Organization for the Advancement of Structured Information Standards (Organisme de normalisation Groupe OASIS)
POSIX	Portable Operating System Interface (Interface de système d'exploitation portable)
PKCS	Public Key Cryptography Standards (Normes de cryptographie à clé publique)
PSK	Pre Shared Key (Clé prépartagée)
REL	Rights Expression Language (Langage REL)
RSA	Rivest ,Shamir and Adleman (Rivest, Shamir et Adleman)
SAML	Security Assertion Markup Language (Langage SAML)
SOAP	Simple Object Access Protocol (Protocole SOAP)
SSID	Service Set ID (Identifiant d'ensemble de services)
TCP	Transmission Control Protocol (Protocole TCP)
TLS	Transport Layer Security (Sécurité de couche transport)
TKIP	Temporal Key Integrity Protocol (Protocole TKIP)
TTL	Time To Live (Durée de vie)
UDDI	Universal Description, Discovery and Integration (Description, découverte et intégration universelles)
UDP	User Datagram Protocol (Protocole UDP)
URI	Uniform Resource Identifier (Identifiant unique de ressource)
URN	Uniform Resource Name (Nom URN)
USB	Universal Serial Bus (Bus série universel)
UTC	Coordinated Universal Time (Temps universel coordonné)
UTF	Unicode Transformation Format (Format de transformation Unicode)
UUID	Universally Unique Identifier (Identifiant unique universel)
WDR	Wide Dynamic Range (Plage dynamique large)
WS	Web Services (Services Web)
WSDL	Web Services Description Language (Langage de description de services Web)
WS-I	Web Services Interoperability (Interopérabilité des services Web)
XML	eXtensible Markup Language (Langage XML)
XPath	XML Path Language (Langage de chemin XML)

4 Présentation

4.1 Généralités

Le présent document définit un ensemble de fonctions d'interface pour la configuration et le fonctionnement de dispositifs en réseau en définissant leurs interfaces côté serveur. Le présent document aborde la découverte de dispositif, la configuration de dispositif ainsi que le cadre des événements.

Tous les services partagent un schéma XML commun, voir l'Article B.4. Les différents services sont définis dans les sections et documents WSDL de service respectifs.

4.2 Services Web

Le terme Services Web est le nom d'une méthode normalisée d'intégration d'applications s'appuyant sur des normes de services Web ouvertes et indépendantes des plateformes telles que XML, SOAP 1.2 [W3C SOAP12-PART1] et WSDL1.1 [W3C WSDL11] sur un réseau IP. XML est utilisé comme syntaxe de description des données, SOAP est utilisé pour le transfert de messages et WSDL est utilisé pour décrire les services.

Ce cadre d'application repose sur les normes des services Web. Tous les services de configuration définis dans le présent document sont exprimés sous forme d'opérations de services Web et sont définis en WSDL, avec HTTP [RFC 2616] comme mécanisme de transport sous-jacent.

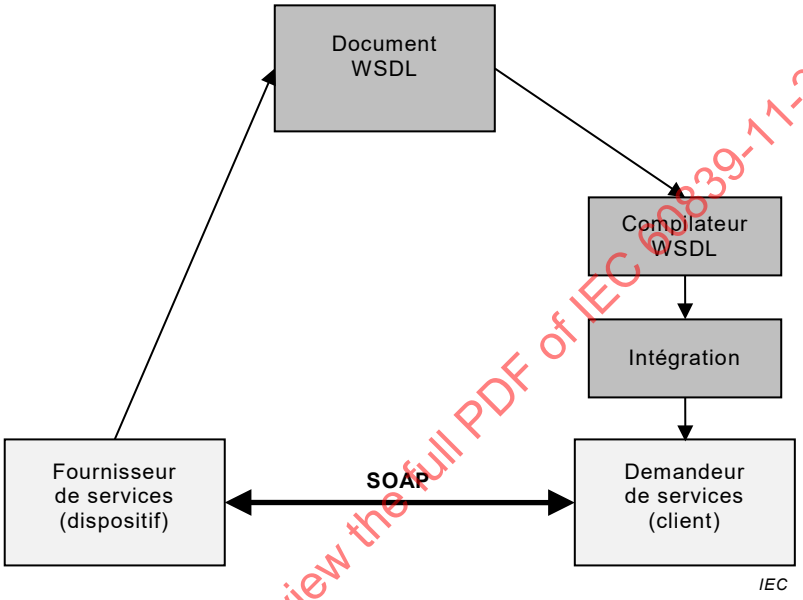


Figure 1 – Principes de développement des services Web

La Figure 1 présente les principes de base du développement basé sur les services Web. Le fournisseur de services (dispositif) met en œuvre le ou les services. Le service est décrit au moyen du WSDL basé sur XML. Ensuite, le WSDL est utilisé comme base de mise en œuvre/d'intégration du demandeur de services (client). L'intégration côté client est simplifiée par l'utilisation d'outils de compilation WSDL qui génèrent un code spécifique à la plateforme pouvant être utilisé par le développeur côté client pour intégrer le service Web dans une application.

Le fournisseur et le demandeur de services Web communiquent à l'aide du protocole d'échange de messages SOAP. SOAP est un protocole de messagerie léger basé sur XML utilisé pour coder les informations dans une demande de service Web et dans un message de réponse avant de les envoyer sur un réseau. Les messages SOAP sont indépendants de tout système d'exploitation ou protocole et peuvent être transportés au moyen de divers protocoles Internet. Le présent document définit les protocoles de transport conformes des messages SOAP pour les services Web décrits.

Le paragraphe relatif aux services Web expose la structure générale des services, la syntaxe de définition des commandes utilisée dans le présent document, les principes de traitement des erreurs et les mécanismes de sécurité adoptés pour les services Web.

Afin de garantir l'interopérabilité, tous les services suivent les recommandations de profil de base 2.0 WS-I et utilisent le schéma littéral/enveloppant.

4.3 Configuration IP

Le paragraphe Configuration IP définit les exigences et recommandations en matière de conformité de la configuration IP. La configuration IP comprend:

- la capacité de communication du réseau IP,
- la configuration IP statique,
- la configuration IP dynamique.

4.4 Découverte de dispositif

Les interfaces de configuration définies dans le présent document sont des interfaces de services Web reposant sur la norme WS-Discovery. L'utilisation du présent document permet de réutiliser un cadre d'application de découverte de services Web adapté existant plutôt que d'avoir à définir un service ou un adressage de service complètement nouveau.

Le présent document présente un comportement de découverte spécifique adapté aux systèmes d'alarme et de sécurité électroniques. Par exemple, une découverte pleinement interopérable exige une définition de service et des critères de recherche de service bien définis. Dans ce but, le présent document couvre le type de dispositif et les définitions des domaines d'application.

Une découverte réussie fournit l'adresse de service du dispositif. Dès que le client obtient l'adresse de service du dispositif, il peut recevoir des informations détaillées sur le dispositif par l'intermédiaire du service de dispositif, voir 4.5 ci-dessous.

4.5 Gestion de dispositif

4.5.1 Généralités

Les fonctions de gestion de dispositif sont traitées par l'intermédiaire du service de dispositif. Le service de dispositif est le point d'entrée vers tous les autres services proposés par un dispositif. Le WSDL pour le service de dispositif est fourni dans le fichier WSDL de gestion de dispositif. Les interfaces de gestion de dispositif sont constituées des sous-catégories suivantes:

- fonctionnalités,
- réseau,
- système,
- sécurité.

4.5.2 Fonctionnalités

Les commandes de fonctionnalités permettent à un client de demander les services offerts par un dispositif et de déterminer les services généraux et spécifiques au fournisseur offerts par le dispositif. Les fonctionnalités sont structurées par service. Le présent document définit l'échange de fonctionnalité pour le dispositif, le dispositif ES et le service d'événement. Pour les autres services, se reporter à la spécification de service respective:

- dispositif,
  - réseau,
  - système,
  - sécurité,
- dispositif ES,
- evénement.

Les fonctionnalités correspondant aux différentes catégories indiquent les commandes et les paramètres disponibles pour le service particulier ou la sous-catégorie de service particulière.

4.5.3 Réseau

L'ensemble suivant de commandes réseau permet une gestion normalisée des fonctions:

- Obtenir et définir le nom d'hôte.
- Obtenir et définir les configurations DNS.
- Obtenir et définir les configurations NTP.
- Obtenir et définir le DNS dynamique.
- Obtenir et définir les configurations d'interface de réseau.
- Activer/désactiver et énumérer les protocoles de réseau.
- Obtenir et définir la passerelle par défaut.
- Obtenir et définir la configuration zéro.
- Obtenir, définir, ajouter et supprimer un filtre d'adresses IP.
- Configuration de l'interface réseau sans fil.

4.5.4 Système

Les commandes système sont utilisées pour gérer les paramètres système suivants du dispositif:

- Obtenir des informations sur le dispositif.
- Effectuer des sauvegardes du système.
- Obtenir et régler la date et l'heure du système.
- Réinitialiser les réglages par défaut d'usine.
- Mettre à niveau le micrologiciel.
- Obtenir le journal système.
- Obtenir des données de diagnostic sur le dispositif (information d'assistance).
- Redémarrer.
- Obtenir et définir les paramètres de découverte du dispositif.

4.5.5 Extraction des informations sur le système

Les informations sur le système, telles que les journaux système, les informations de prise en charge spécifiques au fournisseur et les images de sauvegarde de la configuration, peuvent être extraites à l'aide de MTOM ou HTTP.

La méthode MTOM est prise en charge par les commandes GetSystemLog, GetSystemSupportInformation et GetSystemBackup. La méthode HTTP est prise en charge par la commande GetSystemUris; cela permet d'extraire les URI à partir desquels les fichiers peuvent être téléchargés à l'aide d'une opération HTTP GET.

4.5.6 Mise à niveau du micrologiciel

Deux mécanismes permettent de mettre à niveau le micrologiciel sur un dispositif. Le premier utilise la commande UpgradeSystemFirmware pour envoyer la nouvelle image de micrologiciel à l'aide de MTOM.

Le second est un processus en deux étapes: en premier lieu, le client envoie la commande StartFirmwareUpgrade pour demander au dispositif de se préparer à la mise à niveau, puis envoie l'image du micrologiciel à l'aide de HTTP POST.

La méthode HTTP est conçue pour les dispositifs dont les ressources sont limitées et qui ne peuvent pas recevoir l'image du nouveau micrologiciel dans leur état de fonctionnement normal.

4.5.7 Restauration du système

La fonctionnalité de restauration du système permet de restaurer la configuration d'un dispositif à partir d'une image de sauvegarde. Deux mécanismes sont également proposés. Le premier utilise la commande RestoreSystem pour envoyer l'image de sauvegarde à l'aide de MTOM. Le second utilise la commande StartSystemRestore suivie d'une opération HTTP POST pour envoyer l'image de sauvegarde.

4.5.8 Sécurité

Les opérations de sécurité suivantes sont utilisées pour gérer les configurations de sécurité du dispositif:

- Obtenir et définir la politique de sécurité d'accès.
- Traiter les identifiants et les paramètres utilisateur.
- Traiter les certificats de serveur HTTPS.
- Activer/désactiver l'authentification client HTTPS.
- Fonctions de génération de clé et de téléchargement de certificats.
- Traiter les certificats de supplication IEEE 802.1X.
- Traiter les certificats de CA IEEE 802.1X.
- Configuration IEEE 802.1X.

4.6 DeviceIO (Dispositif ES)

Le service DeviceIO propose des commandes permettant d'extraire et de configurer les paramètres d'entrées et de sorties physiques d'un dispositif.

Le service DeviceIO prend en charge la configuration des interfaces de dispositif suivantes:

- RelayOutputs,
- DigitalInputs,
- Envoyer et/ou recevoir des données en série.

4.7 Traitement des événements

Le traitement des événements repose sur les spécifications OASIS WS-BaseNotification et WS-Topics. Ces spécifications permettent de réutiliser un cadre de notification riche sans qu'il soit nécessaire de redéfinir les principes du traitement des événements, ses formats de base et ses schémas de communication.

Selon WS-BaseNotification, la traversée de pare-feu est traitée par l'intermédiaire d'un schéma de notification PullPoint. Ce schéma ne permet toutefois pas la notification en temps réel. Le présent document définit donc en variante un schéma de communication PullPoint et une interface de service. Le schéma PullPoint permet à un client résidant derrière un pare-feu de recevoir des notifications en temps réel tout en utilisant le cadre d'application WS-BaseNotification.

Le traitement d'un événement totalement normalisé exige des notifications normalisées. Toutefois, les rubriques de notification dépendent dans une large mesure des besoins de l'application. Le présent document définit un ensemble de rubriques de notification de base.

Le WSDL du service d'événement comprenant les extensions est fourni dans le fichier WSDL d'événement.

4.8 Sécurité

Le Paragraphe 4.8 décrit les exigences de sécurité du réseau. Le présent document définit le mécanisme de sécurité à deux niveaux de communication différents:

- sécurité au niveau transport,
- sécurité au niveau du message.

Le présent document définit également la sécurité réseau basée sur le port de la manière suivante.

- IEEE 802.1X

Les exigences générales de sécurité, les définitions et les exigences de sécurité du transport sont spécifiées en 11.3. Les exigences de sécurité au niveau du message sont spécifiées en 5.13. Les exigences IEEE 802.1X sont spécifiées en 8.4.8. La gestion de la sécurité est traitée grâce au service de gestion de dispositif tel qu'indiqué en 4.5.8.

5 Cadre d'application des services Web

5.1 Généralités

Toutes les commandes de gestion et de configuration reposent sur les services Web.

Pour les besoins du présent document:

- le dispositif est un fournisseur de services;
- le client est un demandeur de services.

Un système réseau classique dispose de plusieurs clients qui traitent les opérations de configuration et de gestion de dispositif pour de nombreux dispositifs. Un dispositif fournissant des services peut également agir comme un client.

Les services Web exigent également une manière commune de découvrir les fournisseurs de services. Cette découverte est obtenue au moyen des spécifications de Description, découverte et intégration universelles (UDDI) [UDDI API ver2], [UDDI Data Structure ver2]. Les spécifications UDDI utilisent des courtiers de services pour la découverte de service. Le présent document cible les dispositifs, alors que le modèle UDDI n'est pas orienté dispositif. Par conséquent, l'UDDI et les courtiers de services ne relèvent pas du domaine d'application du présent document.

Selon le présent document, les dispositifs (fournisseurs de services) sont découverts à l'aide de techniques basées sur WS-Discovery [XMLSOAP WS-Discovery]. Les principes de découverte de service sont décrits à l'Article 7.

Les services Web donnent aux développeurs la liberté de définir des échanges de services et de messages, ce qui peut poser des problèmes d'interopérabilité. L'organisme d'interopérabilité des services Web (WS-I) développe des profils normalisés et des lignes directrices permettant de créer des services Web interopérables. Les dispositifs et les clients doivent suivre les lignes directrices du profil de base WS-I 2.0 [WS-I BP 2.0]. Les descriptions de services du présent document suivent les recommandations du profil de base WS-I 2.0.

5.2 Présentation des services

5.2.1 Généralités

Un dispositif doit prendre en charge un certain nombre de services Web définis dans le présent document et dans les spécifications associées.

Le service de gestion de dispositif est non seulement le point d'entrée de tous les autres services du dispositif, mais aussi le service cible du comportement WS-Discovery défini par le présent document, voir l'Article 7.

Le point d'entrée pour le service de gestion de dispositif est fixé à:

```
http://onvif_host/onvif/device_service
```

5.2.2 Exigences des services

Un dispositif doit offrir des services de gestion de dispositif et des services d'événement.

Si un dispositif prend en charge un certain service, il doit répondre à toutes les commandes définies dans le WSDL du service correspondant. Si la commande spécifique n'est pas exigée pour ce service et que le dispositif ne prend pas en charge la commande, il convient que le dispositif réponde à une demande par les codes d'erreur:

env:Receiver,

ter:ActionNotSupported,

Voir 5.12.3 pour les définitions des codes d'erreur.

5.3 Présentation de WSDL

"WSDL est un format XML de description des services réseau, se présentant sous la forme d'un ensemble de points terminaux opérant sur des messages contenant des informations orientées document ou procédure. Les opérations et les messages sont décrits de manière abstraite, puis associés à un protocole de réseau et un format de message concrets afin de définir un point terminal. Les points terminaux concrets associés sont combinés en points terminaux abstraits (services). WSDL est extensible, afin de permettre la description de points terminaux et de leurs messages indépendamment des formats de message ou des protocoles de réseau utilisés pour communiquer" [SOURCE: W3C WSDL11].

Le présent document suit la spécification WSDL 1.1, voir [W3C WSDL11], et utilise le schéma littéral/enveloppant.

Un document WSDL est composé des sections suivantes:

- types – Définition des types de données à l'aide de définitions de schéma XML.
- message – Définition du contenu des messages d'entrée et de sortie.
- opération – Définition de la manière dont les messages d'entrée et de sortie sont associés à une opération logique.
- type de port – Regroupe un ensemble d'opérations.
- liaison – Spécification des protocoles utilisés pour l'échange de message pour un type de port particulier.
- port – Spécifie une adresse pour une liaison.
- service – Utilisé pour regrouper un ensemble de ports associés.

5.4 Espaces de nommage

À noter que les URI d'espace de nommage, énumérés dans les Tableaux 1, 2 et 3, sont principalement utilisés pour créer un identifiant unique. Ils n'ont pas besoin d'être utiles pour extraire un document correspondant, par exemple un schéma XML ou un fichier WSDL.

Les préfixes et les espaces de nommage utilisés dans le présent document figurent dans le Tableau 1. Ces préfixes ne font pas partie de la norme, une mise en œuvre pouvant utiliser le préfixe de son choix.

Tableau 1 – Espaces de nommage définis dans le présent document

Préfixe	URI d'espace de nommage	Description
tt	http://www.onvif.org/ver10/schema	Descriptions de schéma XML du présent document.
tds	http://www.onvif.org/ver10/device/wsdl	Espace de nommage du service de dispositif WSDL.
tmd	http://www.onvif.org/ver10/deviceIO/wsdl	Espace de nommage du service de dispositif ES WSDL.
trt	http://www.onvif.org/ver10/media/wsdl	Espace de nommage du service multimédia WSDL.
tev	http://www.onvif.org/ver10/events/wsdl	Espace de nommage du service d'événement WSDL.
ter	http://www.onvif.org/ver10/error	Espace de nommage des défauts définis par le présent document.
tns1	http://www.onvif.org/ver10/topics	Espace de nommage des rubriques définies par le présent document.

Les espaces de nommage figurant dans le Tableau 2 sont référencés par le présent document.

Tableau 2 – Espaces de nommage référencés (avec préfixe)

Préfixe	URI d'espace de nommage	Description
wsdl	http://schemas.xmlsoap.org/wsdl/	Espace de nommage WSDL pour le cadre d'application WSDL.
wsoap12	http://schemas.xmlsoap.org/wsdl/soap12/	Espace de nommage WSDL pour la liaison WSDL SOAP 1.2.
http	http://schemas.xmlsoap.org/wsdl/http/	Espace de nommage WSDL pour la liaison WSDL GET & POST HTTP.
soapenv	http://www.w3.org/2003/05/soap-envelope	Espace de nommage d'enveloppe défini par SOAP 1.2 [W3C SOAP12-PART11]
xs	http://www.w3.org/2001/XMLSchema	Espace de nommage d'instance défini par XS [W3C XML-Schema-1] et [W3C XML-Schema- 2]
xsi	http://www.w3.org/2001/XMLSchema-instance	Espace de nommage d'instance de schéma XML.
d	http://schemas.xmlsoap.org/ws/2005/04/discovery	Espace de nommage de découverte de dispositif défini par [XMLSOAP WS-Discovery].
wsadis	http://schemas.xmlsoap.org/ws/2004/08/addressing	Espace de nommage d'adressage de dispositif référencé dans [XMLSOAP WS-Discovery].
wsa	http://www.w3.org/2005/08/addressing	Espace de nommage d'adressage de dispositif défini par [W3C WS-Addressing].
wstop	http://docs.oasis-open.org/wsn/t-1	Espace de nommage de schéma de la spécification [OASIS WS-Topics].
wsnt	http://docs.oasis-open.org/wsn/b-2	Espace de nommage de schéma de la spécification [OASIS WS-BaseNotification].
xop	http://www.w3.org/2004/08/xop/include	Espace de nommage d'encapsulation optimisée binaire XML défini par [W3C XOP]

De plus, le présent document se réfère aux espaces de nommage sans préfixe figurant dans le Tableau 3.

Tableau 3 – Espaces de nommage référencés (sans préfixe)

URI d'espace de nommage	Description
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete	Dialecte d'expression de rubrique défini pour les expressions de rubrique.
http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet	Dialecte d'expression de rubrique défini par le présent document.
http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter	Dialecte de filtrage, utilisé pour le filtrage de contenu de message, défini par le présent document.

5.5 Types

Les types de données sont définis à l'aide de descriptions de schéma XML Partie 1 et Partie 2. Tous les types de données définis dans le présent document sont inclus dans le schéma commun, voir Article B.4.

5.6 Messages

Selon WSDL 1.1, les opérations sont décrites au moyen de messages d'entrée et de sortie au format XML. La section de message contient le contenu du message.

Dans le présent document, un message contient deux éléments principaux:

- nom de message,
- parties du message.

Le nom de message spécifie le nom de l'élément. Il est utilisé dans la définition d'opération du document WSDL. Le nom de message définit le nom du message.

L'élément parties du message WSDL permet de définir le format réel du message. Bien qu'un message WSDL puisse être composé de plusieurs parties, le présent document suit le profil de base WS-I [WS-I BP 2.0] et n'autorise pas plus d'un élément de partie dans un message. Le même nom est donc toujours utilisé ("paramètres") pour le nom de partie du message.

La notation WSDL qui suit est utilisée pour le présent document:

```
<message name="'Operation_Name'Request">
      <part name="parameters" element="'prefix':'Operation_Name'"/>
</message>
```

respectivement,

```
<message name="'Operation_Name'Response">
      <part name="parameters" element="'prefix':'Operation_Name'Response"/>
</message>
```

où "prefix" est le préfixe de l'espace de nommage dans lequel est défini le message.

Le présent document utilise des types spécifiques de messages qui encapsulent de multiples parties pour autoriser de multiples arguments (ou données) dans les messages.

5.7 Opérations

5.7.1 Généralités

Les opérations sont définies dans la déclaration WSDL portType (type de port). Une opération peut être de l'un des deux types suivants:

- Unidirectionnel – Le fournisseur de services reçoit un message.
- Demande-réponse – Le fournisseur de services reçoit un message et envoie un message correspondant.

Selon l'opération, différents types de ports peuvent être utilisés.

Le nom d'opération définit le nom de l'opération.

Les opérations de la spécification sont définies à l'aide du format de table suivant présenté dans le Tableau 4.

Tableau 4 – Description des opérations utilisée dans le présent document

Operation_Name	Access_Class_Name
Nom de message	**Description**
'Operation_Name'Request	*Description du message de demande.* *Type*$_{r1}$ **Nom**$_{r1}$ [a_{r1}][b_{r1}] *Type*$_{r2}$ **Nom**$_{r2}$ [a_{r2}][b_{r2}] : *Type*$_{rn}$ **Nom**$_{rn}$ [a_{rn}][b_{rn}]
'Operation_Name'Response	*Description du message de réponse.* *Type*$_{s1}$ **Nom**$_{s1}$ [a_{s1}][b_{s2}] *Type*$_{s2}$ **Nom**$_{s2}$ [a_{s2}][b_{s2}] : *Type*$_{sn}$ **Nom**$_{sn}$ [a_{sn}][b_{sn}]
'FaultMessage_Name'	*Si des défauts spécifiques à cette opération sont définis, ce champ décrit la structure du message de défaut défini.*
Codes de défaut	**Description**
Code *Sous-code* *Sous-code*	*Description du défaut spécifique à l'opération.*

La colonne de description comprend une liste des éléments (le cas échéant) inclus respectivement dans les messages de demande et de réponse. La valeur entre crochets définit les limites supérieure et inférieure du nombre d'occurrences qui peuvent être prévues pour l'élément du type spécifié. Par exemple, Nom$_{s2}$ du tableau ci-dessus se produit au moins a_{s2} fois et au plus b_{s2} fois.

La plupart des commandes ne définissent aucun message de défaut spécifique. Si un message est défini, il figure dans le tableau immédiatement après le message de réponse.

Les codes de défaut figurant dans les tableaux sont les codes de défaut spécifiques qui peuvent être attendus de la commande, voir 5.12.3.3. Chaque commande peut générer un défaut générique, voir 5.12.3.3.

Le Access_Class_Name définit la classe d'accès de l'opération. La classe d'accès caractérise l'impact de l'opération, voir 5.13.2.3.

5.7.2 Type d'opération unidirectionnelle

Un type d'opération unidirectionnelle est utilisé lorsque le fournisseur de services reçoit un message de contrôle et n'envoie aucun message explicite d'accusé de réception ni de

confirmation. Le présent document utilise les opérations unidirectionnelles pour la découverte et les événements uniquement.

Ce type d'opération est défini par un seul message d'entrée.

Utiliser le format de table suivant pour décrire les opérations unidirectionnelles:

Operation_Name	Unidirectionnel
Nom de message	**Description**
'Operation_Name'Request	*Description du message de demande.* *Type*$_1$ **Nom**$_1$ [a_1][b_1] *Type*$_2$ **Nom**$_2$ [a_2][b_2] : *Type*$_n$ **Nom**$_n$ [a_n][b_n]

Ce tableau correspond à la notation WSDL suivante dans les spécifications de services:

```
<operation name="'Operation_Name'">
     <input message="'prefix':'Operation_Name'"/>
</operation>
```

5.7.3 Type d'opération demande-réponse

Un type d'opération demande-réponse est utilisé lorsque le fournisseur de services reçoit un message et répond par un message correspondant.

Ce type d'opération est défini par un message d'entrée, un message de sortie et plusieurs messages de défaut.

Utiliser le format de table suivant pour décrire les opérations demande-réponse:

Operation_Name	Demande-réponse
Nom de message	**Description**
'Operation_Name'Request	*Description du message de demande.* *Type*$_{r1}$ **Nom**$_{r1}$ [a_{r1}][b_{r1}] *Type*$_{r2}$ **Nom**$_{r2}$ [a_{r2}][b_{r2}] : *Type*$_{rn}$ **Nom**$_{rn}$ [a_{rn}][b_{rn}]
'Operation_Name'Response	*Description du message de réponse.* *Type*$_{s1}$ **Nom**$_{s1}$ [a_{s1}][b_{s2}] *Type*$_{s2}$ **Nom**$_{s2}$ [a_{s2}][b_{s2}] : *Type*$_{sn}$ **Nom**$_{sn}$ [a_{sn}][b_{sn}]
'"FaultMessage_Name'	*Si des défauts spécifiques à cette opération sont définis, ce champ décrit la structure du message de défaut défini.*
Codes de défaut	**Description**
Code *Sous-code* *Sous-code*	*Description du défaut spécifique à l'opération.*

Ce tableau correspond à la notation WSDL suivante:

```
<operation name="'Operation_Name'">
     <input message="'prefix':'Operation_Name'"/>
       <output message="'prefix':'Operation_Name'Response"/>
       <fault name> = "Fault" message = "'prefix':'FaultMessage_Name'">
</operation>
```

5.8 Types de ports

Un type de port est un ensemble nommé d'opérations abstraites et de messages abstraits impliqués. Un seul type de port est un ensemble de plusieurs opérations différentes.

Tous les noms d'opérations du présent document sont classés en catégories. Chaque catégorie d'opérations contient une ou plusieurs opérations. Chaque catégorie contient un seul type d'opération et est regroupée en un seul type de port. Une opération unidirectionnelle et une opération demande-réponse ne peuvent jamais exister pour le même type de port.

5.9 Liaison

Une liaison définit une spécification concrète de format de données de transport et de protocole pour un type de port particulier. Il peut exister n'importe quel nombre de liaisons pour un type de port donné.

“Port_type” est un type défini au préalable et “Liaison” est une chaîne de caractères commençant par une lettre majuscule qui définit le nom de la liaison.

Les définitions de liaison pour un dispositif conforme doivent suivre les exigences contenues dans [WS-I BP 2.0]. Cela implique que les liaisons WSDL SOAP 1.2 doivent être utilisées.

La liaison SOAP peut avoir différents styles. Un dispositif conforme doit utiliser le style "document" spécifié au niveau opérationnel.

Les liaisons sont définies dans les spécifications WSDL pour les services respectifs.

5.10 Ports

Le point terminal individuel est spécifié par une adresse unique correspondant à une liaison. Un nom unique doit être attribué à chaque port. Une définition de port contient un nom et un attribut de liaison.

Le présent document n'impose aucun principe de nomination des ports.

5.11 Services

Un service est un ensemble de ports associés. Le présent document n'impose aucun principe de nomination des services.

5.12 Traitement des erreurs

5.12.1 Généralités

Comme pour tout autre protocole, des erreurs peuvent se produire lors des communications ou du traitement des messages ou du protocole.

La spécification classe le traitement des erreurs selon les catégories suivantes:

- erreurs de protocole,
- erreurs SOAP,
- erreurs d'application.

5.12.2 Erreurs de protocole

Les erreurs de protocole sont le résultat d'un message de protocole formé de manière incorrecte, qui peut contenir des valeurs d'en-tête illégales, être reçu de manière intempestive ou faire l'objet d'un retard du connecteur. Pour indiquer et interpréter les erreurs de protocole,

les protocoles HTTP et RTSP ont défini un ensemble de codes de statut normalisés (par exemple, 1xx, 2xx, 3xx, 4xx, 5xx). Conformément au présent document, le dispositif et le client doivent utiliser les codes de statut appropriés définis par les protocoles RTSP et HTTP pour les rapports d'erreur, et les gérer correctement lorsqu'ils en reçoivent.

5.12.3 Erreurs SOAP

5.12.3.1 Généralités

Les erreurs SOAP sont générées par suite d'erreurs de fonctionnement des services Web ou lors du traitement d'un message SOAP. Toutes ces erreurs SOAP doivent être signalées et traitées par l'intermédiaire de messages de défaut SOAP. La spécification SOAP donne un cadre d'application commun bien défini pour traiter les erreurs par l'intermédiaire de SOAP.

Un message de défaut SOAP est un message SOAP normal dont le corps contient un seul élément bien connu (soapenv:Fault). Pour mieux comprendre l'erreur, SOAP a défini une structure de message de défaut SOAP contenant divers composants.

- Code de défaut
- Sous-code
- Raison
- Nœud et rôle
- Détails de défaut

Les éléments d'information **Subcode (Sous-code)** et **Fault Detail (Détail de défaut)** sont destinés à transporter les informations d'erreur spécifiques à l'application.

Le présent document utilise un espace de nommage séparé pour les défauts spécifiques, voir 5.12.3.3:

ter = “http://www.onvif.org/ver10/error”

Les messages de défaut SOAP des différents services Web sont définis dans les définitions des différents services Web. Le serveur et le client doivent utiliser le traitement de message de défaut SOAP 1.2 [W3C SOAP12-PART1], comme spécifié dans le présent document, et doivent suivre les recommandations de traitement des défauts du profil de base WS-I 2.0.

L'exemple qui suit est un message d'erreur (message de défaut SOAP 1.2 via HTTP). Les valeurs en italique sont des valeurs fictives remplaçant les valeurs réelles.

```
HTTP/1.1 500 Internal Server Error
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: application/soap+xml; charset=”utf-8”
DATE: when response was generated
<?xml version=”1.0” ?>
<soapenv:Envelope        xmlns:soapenv="http://www.w3.org/2003/05/soap-
envelope"
   xmlns:ter="http://www.onvif.org/ver10/error"
   xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
<soapenv:Body>
 <soapenv:Fault>
   <soapenv:Code>
     <soapenv:Value>fault code </soapenv:Value>
     <soapenv:Subcode>
       <soapenv:Value>ter:fault subcode</soapenv:Value>
         <soapenv:Subcode>
           <soapenv:Value>ter:fault subcode</soapenv:Value>
         </soapenv:Subcode>
     </soapenv:Subcode>
```

```
    </soapenv:Code>
    <soapenv:Reason>
      <soapenv:Text xml:lang="en">fault reason</soapenv:Text>
    </soapenv:Reason>
    <soapenv:Node>http://www.w3.org/2003/05/soap-
envelope/node/ultimateReceiver</soapenv:Node>
    <soapenv:Role>http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver</soapenv:Role>
    <soapenv:Detail>
      <soapenv:Text>fault detail</soapenv:Text>
    </soapenv:Detail>
  </soapenv:Fault>
</soapenv:Body>
</soapenv:Envelope>
```

Le Tableau 5 suivant résume les codes de défaut SOAP généraux (les codes de défaut sont définis dans SOAP version 1.2 Part 1: Messaging Framework). Le serveur et le client peuvent définir des sous-codes de défaut supplémentaires pour les applications.

La distinction est faite entre les défauts génériques et les défauts spécifiques. Toute commande peut générer un défaut générique. Les défauts spécifiques sont associés à une commande ou un jeu de commandes spécifique. Les défauts spécifiques qui s'appliquent à une commande particulière sont définis dans les tableaux de définition des commandes.

Dans le Tableau 5, le code de défaut, le sous-code et la raison de défaut sont définis comme des valeurs normatives. La colonne de description est ajoutée à titre d'information.

5.12.3.2 Défauts génériques

Le Tableau 5 répertorie les codes de défauts génériques et, le cas échéant, les sous-codes. Toutes les mises en œuvre de serveur et de client doivent traiter tous les défauts énumérés ci-dessous. Une commande de service Web peut retourner un ou plusieurs des défauts génériques.

Les défauts indiqués sans sous-code n'ont aucune valeur sous-code.

Tableau 5 – Défauts génériques

Code de défaut	Sous-code	Raison de défaut	Description
env:VersionMismatch		Défaut de correspondance de version SOAP	Le dispositif a trouvé un élément d'information non valide à la place de l'élément d'information attendu Envelope.
env:MustUnderstand		Blocs d'en-tête SOAP incompris	Un ou plusieurs blocs d'en-tête SOAP obligatoires n'ont pas été compris.
env:DataEncodingUnknown		Codage de données SOAP non pris en charge	Un bloc d'en-tête SOAP ou un élément d'information enfant dans le corps SOAP a un codage de données non pris en charge par le dispositif.
env:Sender	ter:WellFormed	Erreur bien formée	Une violation d'erreur bien formée XML s'est produite.
env:Sender	ter:TagMismatch	Défaut de correspondance de balise	Il y a un défaut de correspondance de nom de balise ou d'espace de nommage.
env:Sender	ter:Tag	Aucune balise	Il manque une balise XML.
env:Sender	ter:Namespace	Erreur d'espace de nommage	Une erreur d'espace de nommage SOAP s'est produite.
env:Sender	ter:MissingAttr	Attribut exigé absent	Il manque un attribut exigé.

Code de défaut	Sous-code	Raison de défaut	Description
env:Sender	ter:ProhibAttr	Attribut interdit	Un attribut interdit est présent.
env:Sender	ter:InvalidArgs	Arguments non valides	Une erreur due à l'un des cas suivants: – argument manquant – arguments trop nombreux – arguments de type de données incorrect.
env:Sender	ter:InvalidArgVal	Valeur argument invalide	La valeur de l'argument est invalide.
env:Sender	ter:UnknownAction	Action inconnue	Une action inconnue est spécifiée.
env:Sender	ter:OperationProhibited	Opération non autorisée	L'opération demandée n'est pas autorisée par le dispositif.
env:Sender	ter:NotAuthorized	Expéditeur non autorisé	L'action demandée exige une autorisation et l'expéditeur n'est pas autorisé.
env:Receiver	ter:ActionNotSupported	Action facultative non mise en œuvre	L'action demandée est facultative et n'est pas mise en œuvre par le dispositif.
env:Receiver	ter:Action	Échec action	L'action SOAP demandée a échoué.
env:Receiver	ter:OutofMemory	Hors mémoire	Le dispositif n'a pas de mémoire suffisante pour terminer l'action.
env:Receiver	ter:CriticalError	Erreur critique	Le dispositif a rencontré une condition d'erreur qu'il ne peut pas régler lui-même et a besoin d'une réinitialisation ou d'une réalimentation.

5.12.3.3 Défauts spécifiques

Les défauts spécifiques s'appliquent uniquement à une commande ou un ensemble de commandes spécifique. Les défauts spécifiques sont déclarés comme faisant partie intégrante des définitions de service.

5.12.3.4 Erreurs HTTP

Si le serveur attend le début du message entrant et qu'aucun message SOAP n'est reçu, il ne doit pas générer un défaut SOAP et envoie à la place une réponse d'erreur HTTP (voir le Tableau 6).

Tableau 6 – Erreurs HTTP

Erreur HTTP	Code d'erreur HTTP	Raison HTTP
Demande mal formulée	400	Demande incorrecte
Exige une autorisation	401	Non autorisé
La méthode HTTP n'est pas POST ou GET	405	Méthode non autorisée
Méthode d'encapsulation de message non prise en charge	415	Supports non pris en charge

Il convient qu'un serveur évite de signaler les erreurs internes, car cela peut révéler les failles de sécurité qui peuvent faire l'objet d'un mauvais usage.

5.13 Sécurité

5.13.1 Authentification

Un dispositif doit prendre en charge l'authentification Digest conformément à [RFC 2617] pour l'authentification des demandes de services Web.

L'authentification Digest ne confère qu'un niveau rudimentaire de sécurité. Dans un système dans lequel la sécurité est importante, il est recommandé de toujours configurer le dispositif pour un accès TLS (voir 11.2). L'authentification Digest, combinée avec la sécurité de niveau transport protégé par TLS, avec authentification de client et de serveur, produit un niveau acceptable de sécurité dans un grand nombre de systèmes.

Il convient qu'un dispositif authentifie une demande RTSP au niveau RTSP. Si HTTP est utilisé pour formuler la demande RTSP, le dispositif ne doit pas authentifier à ce niveau.

Un dispositif doit, lors de l'authentification des méthodes RTSP et HTTP, utiliser un utilisateur/des identifiants issus du même ensemble d'utilisateurs/d'identifiants qui sont utilisés pour la partie WS. Par ailleurs, l'authentification Digest [RFC 2617] doit être utilisée pour RTSP et HTTP.

5.13.2 Contrôle d'accès sur la base d'utilisateurs

5.13.2.1 Généralités

Le cadre d'autorisation décrit en 5.13 permet l'authentification des demandes de services. Une fois la demande de service authentifiée, le dispositif doit déterminer, en fonction de sa politique d'accès, si le demandeur est autorisé à recevoir le service.

Le présent document définit un ensemble de commandes pour gérer les identifiants d'utilisateur, voir 8.4. Ces commandes permettent d'associer des utilisateurs aux différents niveaux d'utilisateur définis en 5.13.2.2.

Un dispositif peut prendre en charge la définition d'une politique d'accès personnalisée par l'utilisateur du dispositif par l'intermédiaire des opérations définies en 8.4 visant à obtenir et à définir la politique d'accès.

5.13.2.2 Niveaux d'utilisateur

Chaque utilisateur est associé à un seul des niveaux d'utilisateur suivants:

1) Administrateur
2) Opérateur
3) Utilisateur
4) Anonyme

Les utilisateurs non authentifiés sont placés dans la catégorie anonyme, et un dispositif ne doit pas ajouter des utilisateurs à la catégorie de niveau d'utilisateur anonyme.

5.13.2.3 Classes d'accès pour les demandes de services

Les demandes de services sont répertoriées dans des classes d'accès en fonction de leur impact. Les classes d'accès suivantes sont définies:

- PRE_AUTH

  Le service ne doit pas exiger l'authentification de l'utilisateur.

  Exemple: GetEndpointReference
- READ_SYSTEM

Le service lit les informations de configuration du système à partir du dispositif.

Exemple: GetNetworkInterfaces

- READ_SYSTEM_SENSITIVE

  Le service lit des informations sensibles (mais pas vraiment confidentielles) de configuration du système à partir du dispositif.

- READ_SYSTEM_SECRET

  Le service lit des informations confidentielles de configuration du système à partir du dispositif.

  Exemple: GetSystemLog

- WRITE_SYSTEM

  Le service modifie la configuration du système du dispositif.

  Exemple: SetNetworkDefaultGateway

- UNRECOVERABLE

  Le service modifie de façon irréversible la configuration du système du dispositif.

  Exemple: SetSystemFactoryDefault

- READ_MEDIA

  Le service lit des données relatives aux supports enregistrés.

  Exemple: GetRecordings

- ACTUATE

  Le service affecte le comportement d'exécution du système.

  Exemple: CreateRecordingJob

Le Tableau 7 définit, pour chaque classe d'accès, les niveaux d'utilisateur accessibles. Un utilisateur du niveau c doit pouvoir accéder à une demande de service associée à la classe d'accès r si et seulement si un "X" est présent dans la cellule au niveau de la colonne c et de la ligne r.

Tableau 7 – Classe d'accès au mapping du niveau d'utilisateur

	Administrateur	Opérateur	Utilisateur	Anonyme
PRE_AUTH	X	X	X	X
READ_SYSTEM	X	X	X	
READ_SYSTEM_SENSITIVE	X	X		
READ_SYSTEM_SECRET	X			
WRITE_SYSTEM	X			
UNRECOVERABLE	X			
READ_MEDIA	X	X	X	
ACTUATE	X	X		

5.13.2.4 Politique d'accès par défaut

Par défaut, il convient que le dispositif applique la politique d'accès par défaut définie par le présent document et par d'autres spécifications basées sur le présent document, qui offre un niveau acceptable de sécurité dans de nombreux systèmes.

La politique d'accès par défaut est définie pour chaque demande de service. Cela s'effectue en identifiant la classe d'accès qu'il convient d'associer à une demande de service spécifique. Voir 5.7.1 pour plus d'informations sur cette opération.

5.14 Représentation sous forme de chaînes

5.14.1 Jeu de caractères

Un dispositif doit prendre en charge le jeu de caractères UTF-8 et peut prendre en charge d'autres jeux de caractères. Si un client envoie une demande avec le jeu de caractères UTF-8, le dispositif doit toujours répondre en UTF-8.

5.14.2 Caractères autorisés en chaînes

Un dispositif ne doit appliquer aucune restriction concernant les caractères légalement autorisés en chaînes qui ne figurent pas explicitement dans le présent document ou dans une autre spécification basée sur le présent document.

5.15 Extensions de propriété

Les extensions de propriété applicables au présent document doivent être traitées de la façon suivante:

1) Il est fortement recommandé d'ajouter des extensions de propriété en définissant de nouvelles commandes de services Web, dans un espace de nommage différent de l'espace de nommage cible, et non en étendant les éléments de schéma existants. Cette option permet d'instaurer une interopérabilité complète en amont et en aval entre des extensions de différents fournisseurs.
2) Si la définition de nouvelles commandes n'est pas la meilleure façon d'ajouter des extensions de propriété, il est recommandé, dans la mesure du possible, d'exprimer les extensions d'éléments de propriété sous forme de déclarations d'attributs au lieu d'ajouter de nouveaux éléments de schéma aux types existants. Les nouveaux attributs de propriété doivent être déclarés dans un espace de nommage différent de l'espace de nommage cible. Cette option permet d'instaurer une interopérabilité complète en amont et en aval.
3) S'il est impossible de déclarer les extensions de propriété à l'aide de nouvelles commandes ou de nouveaux attributs, les extensions de propriété doivent être déclarées en ajoutant un élément d'extension de type unique et obligatoire devant le point d'extension, puis en ajoutant tous les nouveaux éléments de propriété dans cet élément d'extension. Les nouveaux éléments doivent être déclarés dans un espace de nommage différent de l'espace de nommage cible. Chaque élément d'extension de propriété doit avoir minOccurs="0" et les déclarations des éléments d'extension de propriété dans le schéma doivent être suivies par un xs:toute déclaration d'élément dans l'espace de nommage cible avec minOccurs="0" et maxOccurs="unbounded".

6 Configuration IP

Le dispositif et le client communiquent sur un réseau IP ouvert ou fermé. Le présent document n'impose aucune restriction ou exigence générale sur le type de réseau. Cependant, des liaisons de communication doivent pouvoir être établies entre les entités conformément au cadre architectural spécifié à l'Article 4. La configuration IP du dispositif comprend des paramètres tels que des adresses IP et une passerelle par défaut.

Un dispositif doit avoir au moins une interface réseau qui lui assure une connectivité de réseau IP. De manière similaire, le client doit avoir au moins une interface réseau qui lui assure une connectivité IP et permet la communication de données entre le dispositif et le client.

Le dispositif et le client doivent prendre en charge la communication réseau basée sur IPv4. Il convient que le dispositif et le client prennent en charge la communication réseau basée sur IPv6.

Une configuration IP statique doit pouvoir être effectuée sur le dispositif en utilisant une interface de configuration réseau ou locale.

Il convient que le dispositif prenne en charge la configuration IP dynamique d'adresses locales-liaison conformément à la norme [RFC 3927]. Un dispositif qui prend en charge IPv6 doit prendre en charge la configuration IP sans état conformément à la norme [RFC 4862] et la découverte de voisin conformément à la norme [RFC 4861].

Le dispositif doit prendre en charge la configuration IP dynamique conformément à la norme [RFC 2131]. Un dispositif qui prend en charge IPv6 doit prendre en charge la configuration IP avec état via DHCPv6 conformément à la norme [RFC 3315] si elle est signalée par l'intermédiaire de la fonctionnalité correspondante.

Le dispositif peut prendre en charge un mécanisme de configuration IP supplémentaire.

La configuration réseau d'un dispositif doit être fournie par l'intermédiaire du service de gestion de dispositif tel que spécifié en 8.3 et peut en outre être fournie par des interfaces locales. La configuration réseau par l'intermédiaire d'une interface locale ne relève pas du domaine d'application du présent document.

Dans la configuration de dispositif par défaut, la configuration d'adresse DHCP et liaison-locale dynamique (sans état) doit être activée. Même si le dispositif est configuré par l'intermédiaire d'une configuration d'adresse statique, il convient que l'adresse de liaison-locale soit activée par défaut.

Lorsqu'un dispositif est connecté à un réseau IPv4, il convient que les priorités d'attribution d'adresse (adresse liaison-locale par rapport à l'adresse acheminable) soient définies comme recommandé dans la norme [RFC 3927].

Les détails supplémentaires quant à la manière d'obtenir la connectivité IP ne relèvent pas du domaine d'application du présent document.

7 Découverte de dispositif

7.1 Généralités

Un client recherche les dispositifs disponibles en utilisant le protocole de découverte de services Web dynamique [XMLSOAP WS-Discovery].

Un dispositif satisfaisant au présent document doit mettre en œuvre le rôle Target Service tel que spécifié dans [XMLSOAP WS-Discovery].

Si nécessaire, un client satisfaisant au présent document doit mettre en œuvre le rôle Client tel que spécifié dans [XMLSOAP WS-Discovery].

[XMLSOAP WS-Discovery] décrit l'identifiant UUID (Universally Unique Identifier): une recommandation de format URI pour des références de point terminal, mais le présent document supplante cette recommandation. Au lieu de cela, le format Uniform Resource Name: Universally Unique Identifier (URN:UUID) est utilisé [RFC 4122] (voir 7.3.1).

7.2 Modes de fonctionnement

Le dispositif doit pouvoir fonctionner dans deux modes:

- découvrable,
- non découvrable.

Un dispositif en mode découvrable envoie des messages Hello en multidiffusion une fois qu'il est connecté au réseau ou envoie ses modifications de statut conformément à [XMLSOAP WS-Discovery]. De plus, il écoute toujours les messages Probe (sonde) et Resolve (Résoudre) et envoie des réponses en conséquence. Un dispositif en mode non découvrable ne doit pas écouter les messages [XMLSOAP WS-Discovery] ni envoyer de tels messages.

Le mode découvrable doit être le comportement par défaut d'un dispositif. Afin de contrer les attaques de refus de service, un dispositif doit pouvoir être configuré en mode non découvrable grâce à l'opération définie en 8.4.19.

7.3 Définitions de découverte

7.3.1 Référence de point terminal

Un dispositif ou un point terminal qui joue le rôle de client doit utiliser un URN:UUID [RFC 4122] comme propriété d'adresse de sa référence de point terminal.

Le dispositif ou un point terminal qui joue le rôle de client doit utiliser un identifiant stable, globalement unique et constant entre les interfaces réseau comme faisant partie de sa propriété de référence de point terminal. La combinaison d'un wsadis:Address et d'un wsadis:ReferenceProperties constitue un identifiant stable et globalement unique.

7.3.2 Hello

7.3.2.1 Types

Un dispositif doit inclure le type de port de service de gestion de dispositif, c'est-à-dire tds:Device, dans la déclaration `<d:Types>`.

L'exemple suivant indique comment le type est codé dans le corps SOAP Hello:

```
<d:Types>tds:Device</d:Types>.
```

Le message Hello peut comprendre des types additionnels.

7.3.2.2 Domaines d'application

7.3.2.2.1 Généralités

Un dispositif doit inclure l'attribut `<d:Scopes>` avec les domaines d'application du dispositif dans le message Hello.

Le domaine d'application du dispositif est défini à l'aide des URI de la norme [RFC 3986]. Le présent document définit les attributs de domaine d'application comme suit:

L'attribut de schéma: onvif

L'attribut d'autorité: www.onvif.org

Cela implique que tous les URI de domaine d'application définis par le présent document se présentent sous le format suivant:

```
onvif://www.onvif.org/<path>
```

Un dispositif peut avoir d'autres URI de domaine d'application. Ces URI ne sont pas limités par les attributs de domaine d'application définis dans le présent document.

Le Tableau 8 définit un ensemble de paramètres de domaine d'application. Hormis ces paramètres normalisés, un paramètre de domaine d'application tel que défini par le propriétaire du dispositif doit pouvoir être défini. Les paramètres de domaine d'application peuvent être énumérés et définis à l'aide des commandes définies en 8.4.

Tableau 8 – Paramètres de domaine d'application

Catégorie	Valeurs définies	Description
Emplacement	Une chaîne de caractères ou une valeur de chemin.	L'emplacement définit l'emplacement physique du dispositif. La valeur d'emplacement peut être une chaîne décrivant l'emplacement physique du dispositif.
Matériel	Une chaîne de caractères ou une valeur de chemin.	Chaîne ou valeur de chemin décrivant le matériel du dispositif. La liste de domaines d'application d'un dispositif doit comprendre au moins une entrée de matériel.
Nom	Une chaîne de caractères ou une valeur de chemin.	Nom ouvert à la recherche du dispositif. La liste de domaines d'application d'un dispositif doit comprendre au moins une entrée de nom.

La liste de domaines d'application d'un dispositif doit comprendre au moins une entrée fixe (définie par le fournisseur du dispositif) de matériel et des catégories de nom respectivement. La liste de domaines d'application d'un dispositif peut comprendre des attributs de domaine d'application supplémentaires.

La liste de domaines d'application d'un dispositif peut comprendre un nombre arbitraire de domaines d'application. Cela implique qu'une unité peut, par exemple, définir plusieurs domaines d'application d'emplacement différents. Une sonde est comparée à tous les domaines d'application de la liste.

7.3.2.2.2 Exemple

L'exemple suivant présente l'utilisation de la valeur de domaine d'application. Ceci n'est qu'un exemple, et en aucune façon une indication du type de paramètre de domaine d'application devant faire partie d'une configuration de dispositif. Dans cet exemple, le dispositif est par hypothèse configuré avec les domaines d'application suivants:

```
onvif://www.onvif.org/hardware/D1-566
onvif://www.onvif.org/location/country/china
onvif://www.onvif.org/location/city/bejing
onvif://www.onvif.org/location/building/headquarter
onvif://www.onvif.org/location/floor/R5
onvif://www.onvif.org/name/ARV-453
```

Un client qui sonde le dispositif avec le domaine d'application onvif://www.onvif.org obtient une correspondance. De manière similaire, une sonde pour le dispositif avec le domaine d'application:

onvif://www.onvif.org/location/country/china

génère une correspondance. Une sonde avec:

`onvif://www.onvif.org/hardware/D1`

ne génère aucune correspondance.

7.3.2.3 Adresses

Un dispositif doit comprendre l'élément <d:XAddrs> avec l'adresse ou les adresses du service de dispositif dans le message Hello. Un URI doit être fourni pour chaque protocole (http, https), ainsi qu'une adresse IP disponible en externe.

Il convient que le dispositif fournisse une entrée de service de dispositif de port 80 afin de permettre la traversée du pare-feu.

Les principes de configuration d'adressage IP d'un dispositif sont définis à l'Article 6.

7.3.3 Sonde et correspondance de sonde

Pour les définitions de types de correspondances, de domaines d'application et d'adresses de sonde de dispositif, voir 7.3.2 Hello.

Le dispositif doit au moins prendre en charge la règle suivante de mise en correspondance de domaine d'application:

`http://schemas.xmlsoap.org/ws/2005/04/discovery`

Ces définitions de mise en correspondance de domaine d'application diffèrent légèrement de la définition de [XMLSOAP WS-Discovery], la norme [RFC 2396] étant remplacée par la norme [RFC 3986].

Un dispositif doit inclure l'élément `<d:XAddrs>` avec les adresses du service de dispositif dans un message de correspondance de sonde de mise en correspondance. Dans la plupart des cas, l'élément `<d:XAddrs>` ne contient qu'une seule adresse vers le service de gestion de dispositif comme défini en 5.2.

7.3.4 Résolution et correspondance de résolution

Le présent document exige d'inclure des informations d'adresse de point terminal dans les messages Hello et Probe Match. Dans la plupart des cas, il n'est pas nécessaire d'échanger des résolutions et des correspondances de résolution. Cependant, pour être compatible avec la spécification [XMLSOAP WS-Discovery], il convient qu'un dispositif mette en œuvre la réponse de correspondance de résolution.

7.3.5 Bye

Il convient qu'un dispositif envoie un message Bye unilatéral lorsqu'il se prépare à quitter un réseau comme décrit dans [XMLSOAP WS-Discovery].

7.3.6 Messages de défaut SOAP

En cas d'erreur avec le paquet de multidiffusion, il convient que le dispositif et le client rejettent et ignorent la demande de façon silencieuse. Il n'est pas recommandé d'envoyer une réponse d'erreur en raison du risque de génération de grandes quantités de paquets si de nombreux dispositifs envoient une réponse d'erreur à la même demande. À des fins d'exhaustivité, le traitement des erreurs avec le paquet de monodiffusion est décrit ci-dessous.

Si un dispositif reçoit un message Probe à monodiffusion et qu'il ne prend pas en charge la règle de mise en correspondance, il peut choisir de ne pas envoyer un message Probe Match et de générer plutôt un défaut SOAP conforme à SOAP 1.2, comme suit:

[action] `http://schemas.xmlsoap.org/ws/2005/04/discovery/fault`

[Code] `s12:Sender`

[Sous-code] `d:MatchingRuleNotSupported`

[Raison] `Par exemple, la règle de mise en correspondance spécifiée n'est pas prise en charge`

[Détail] `<d: SupportedMatchingRules>`

`Liste de xs:anyURI`

`</d: SupportedMatchingRules>`

8 Gestion de dispositif

8.1 Généralités

Le service de dispositif est divisé en cinq catégories différentes: commandes de fonctionnalités, de réseau, de système, d'E/S et de sécurité. Cet ensemble de commandes peut être utilisé pour obtenir des informations sur les fonctionnalités et les configurations du dispositif ou pour définir des configurations de dispositif. Un dispositif doit prendre en charge le service de gestion de dispositif tel que spécifié à l'Article B.1. Un ensemble d'opérations de base est exigé pour le service de gestion de dispositif, la prise en charge d'autres opérations étant recommandée ou facultative. Les exigences détaillées sont présentées sous les descriptions de commandes.

8.2 Fonctionnalités

8.2.1 Get WSDL URL

Un point terminal peut demander une URL qui peut être utilisée pour extraire le schéma et les définitions WSDL complets d'un dispositif. La commande génère en retour un point d'entrée d'URL où toutes les définitions WSDL et de schéma spécifiques au produit nécessaires peuvent être extraites. Le dispositif doit fournir une URL pour le téléchargement WSDL et de schéma par l'intermédiaire de la commande GetWsdlUrl.

Tableau 9 – Commande GetWSDLUrl

GetWsdlUrl	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetWsdlUrlRequest	*Ceci est un message vide*
GetWsdlUrlResponse	*L'URL demandée* xs:anyURI **WsdlUrl** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.2.2 Échange de fonctionnalité

8.2.2.1 Généralités

Un point terminal peut demander les fonctionnalités d'un dispositif à l'aide de l'opération de réponse de demande d'échange de fonctionnalité. Un dispositif doit indiquer ses fonctionnalités à l'aide des commandes GetServices/GetServiceCapabilities.

La liste de fonctionnalités comprend des références aux adresses (XAddr) du service qui met en œuvre les opérations d'interface dans la catégorie.

8.2.2.2 GetServices

Retourne une partie des services de dispositifs et probablement leurs fonctionnalités disponibles. Le message de réponse des fonctionnalités retournées n'est pas typé de façon à pouvoir ajouter ultérieurement des services, des révisions de service et des fonctionnalités de service. Toutes les fonctionnalités de service retournées doivent être structurées par différents espaces de nommage qui sont pris en charge par un dispositif.

Le dispositif doit prendre en charge cette méthode. Pour des informations complémentaires sur la structure de la réponse GetServices avec les fonctionnalités, se référer à l'Annexe A.

Tableau 10 – Commande GetServices

GetServices	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetServicesRequest	*Le message contient une demande relative à l'ensemble des services du dispositif et probablement aux fonctionnalités de chaque service. Si la commande Boolean IncludeCapability est définie, la réponse doit alors inclure les fonctionnalités des services.* boolean **IncludeCapability**[1][1]
GetServicesResponse	*Le message de réponse de fonctionnalité contient les informations demandées à propos des services.* tt:ServiceList [1][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.2.2.3 GetServiceCapabilities

Cette commande retourne les fonctionnalités du service de dispositif. Le dispositif doit prendre en charge cette commande.

Le Tableau 11 et le Tableau 12 décrivent la manière d'interpréter les fonctionnalités indiquées.

Tableau 11 – Commande GetServiceCapabilities

GetServiceCapabilities	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetServiceCapabilitiesRequest	*Ceci est un message vide.*
GetServiceCapabilitiesResponse	*Le message de réponse de fonctionnalité contient les fonctionnalités de dispositif demandées en utilisant une structure de capacité XML hiérarchique.* tds:DeviceServiceCapabilities **Capabilities** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

Tableau 12 – Fonctionnalités dans la commande GetServiceCapabilities

Catégorie	Fonctionnalité	Description
Réseau	IPFilter	Indique si le dispositif prend en charge la commande de filtrage IP en utilisant les commandes de 8.3.18, 8.3.19, 8.3.20 et 8.3.21
	ZeroConfiguration	Indique si le dispositif prend en charge la configuration zéro conformément aux commandes de 8.3.16 et de 8.3.17.
	IPVersion6	Indique si le dispositif prend en charge IP version 6.
	DynDNS	Indique si le dispositif prend en charge la configuration DNS dynamique selon 8.3.8 et 8.3.9.
	Dot11Configuration	Indique si le dispositif prend en charge la configuration IEEE 802.11 comme spécifié en 8.3.22.
	HostnameFromDHCP	Indique si l'extraction du nom d'hôte via DHCP est prise en charge par le dispositif.
	NTP	Indique le nombre maximal de serveurs NTP pris en charge par la commande SetNTP des dispositifs.
	Dot1XConfigurations	Indique le nombre maximal de configurations Dot1X pris en charge par le dispositif, voir 8.5.8.
	DHCPv6	Indique la prise en charge du DHCP IPv6 avec état.
Système	DiscoveryResolve	Indique si le dispositif répond aux demandes de résolution comme décrit en 7.3.4.
	DiscoveryBye	Indique si le dispositif envoie des messages Bye comme décrit en 7.3.5
	SystemBackup	Indique si le dispositif prend en charge la sauvegarde et la restauration du système comme spécifié en 8.4.3 et en 8.4.5
	FirmwareUpgrade	Indique si le dispositif prend en charge la mise à niveau de micrologiciel comme spécifié en 8.4.10.
	SystemLogging	Indique si le dispositif prend en charge l'extraction de journal système comme spécifié en 8.4.11.
	HttpSystemBackup	Indique si le dispositif prend en charge la sauvegarde et la restauration du système à l'aide de HTTP GET et de POST.
	HttpFirmwareUpgrade	Indique si le dispositif prend en charge la mise à niveau de micrologiciel à l'aide de HTTP POST.
	HTTPSystemLogging	Indique si le dispositif prend en charge l'extraction de journal système à l'aide de HTTP Get, voir 8.4.2.
	HTTPSupportInformation	Indique si le dispositif prend en charge l'extraction d'informations de prise en charge à l'aide de HTTP Get, voir 8.4.2.
Sécurité	TLS1.0	Prise en charge de TLS 1.0.
	TLS1.1	Prise en charge de TLS 1.1.
	TLS1.2	Prise en charge de TLS 1.2.
	OnboardKeyGeneration	Indique si le dispositif prend en charge la génération de clé intégrée et la création de certificats autosignés comme spécifié en 8.5.9.
	AccessPolicyConfig	Indique si le dispositif prend en charge l'extraction et le chargement de politiques de contrôle d'accès de dispositif selon 8.5.2 et 8.5.3.
	DefaultAccessPolicy	Indique si le dispositif prend en charge les politiques d'accès par défaut tel que défini en 5.13.2.4.
	HttpDigest	Indique si le dispositif prend en charge l'authentification HTTP digest.
	Dot1X	Indique si le dispositif prend en charge l'authentification de réseau basée sur le port IEEE 802.1X
	SupportedEAPMethod	Liste des types de méthodes EAP pris en charge. Les numéros correspondent à l'IANA [EAP-Registry].
	RemoteUserHandling	Indique si le dispositif prend en charge le traitement utilisateur distant et les méthodes correspondantes définies en 8.5.22 et 8.5.23.
	MaxUsers	Le nombre maximal d'utilisateurs pris en charge par le dispositif

8.3 Réseau

8.3.1 Obtention de nom d'hôte

Cette opération est utilisée par un point terminal pour obtenir le nom d'hôte d'un dispositif. Le dispositif doit retourner ses configurations de nom d'hôte par l'intermédiaire de la commande GetHostname comme décrit dans le Tableau 13.

Tableau 13 – Commande GetHostname

GetHostname	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetHostnameRequest	*Ceci est un message vide.*
GetHostnameResponse	*Ce message contient:* • *"FromDHCP": True (Vrai) si le nom d'hôte est obtenu via DHCP* • *"Name": Nom de l'hôte. Dans le cas de DHCP, le nom d'hôte a été obtenu à partir du serveur DHCP.* xs:boolean **FromDHCP** [1][1] xs:token **Name** [0][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.2 Définition de nom d'hôte

Cette opération définit le nom d'hôte sur un dispositif. Les configurations de nom d'hôte du dispositif doivent pouvoir être définies grâce à la commande SetHostname comme décrit dans le Tableau 14. Attention: un appel adressé à SetDNS peut écraser un nom d'hôte déjà défini.

Un dispositif doit accepter les chaînes formatées selon 2.1 de la norme RFC 1123:1989, ou bien selon la norme RFC 952; les autres chaînes doivent être considérées comme non valides.

Un dispositif doit essayer d'extraire le nom via DHCP lorsque la fonctionnalité HostnameFromDHCP est définie et qu'une chaîne de nom vide est fournie.

Tableau 14 – Commande SetHostname

SetHostname	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetHostnameRequest	*Ce message contient:* • *"Name": Nom de l'hôte. Si le Nom est une chaîne vide, il convient d'extraire le nom d'hôte à partir de DHCP, sinon le Nom spécifié doit être utilisé.* xs:token **Name** [1][1]
SetHostnameResponse	*Ceci est un message vide*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidHostname	*Le dispositif ne peut pas accepter le nom d'hôte demandé.*

8.3.3 Définition de nom d'hôte à partir de DHCP

Cette opération, comme décrit dans le Tableau 15, détermine si le nom d'hôte doit être extrait du DHCP.

Un dispositif doit prendre en charge cette commande si la prise en charge est signalée via la fonctionnalité HostnameFromDHCP. En fonction de la mise en œuvre du dispositif, la modification peut ne prendre effet qu'après un redémarrage du dispositif.

Tableau 15 – Commande SetHostnameFromDHCP

SetHostnameFromDHCP	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetHostnameFromDHCPRequest	*Ce message contient:* • *"FromDHCP": True (Vrai) si le nom d'hôte doit être obtenu via DHCP.* xs:boolean **FromDHCP**[1][1]
SetHostnameFromDHCPResponse	*Indique si un redémarrage est nécessaire en cas de modification des paramètres du nom d'hôte.* xs:boolean **RebootNeeded** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.4 Obtention des paramètres DNS

Cette opération permet d'obtenir les paramètres DNS d'un dispositif comme décrit dans le Tableau 16. Le dispositif doit retourner ses configurations DNS grâce à la commande GetDNS.

Tableau 16 – Commande GetDNS

GetDNS	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDNSRequest	*Ceci est un message vide.*
GetDNSResponse	*Ce message contient:* • *"FromDHCP": True (Vrai) si les serveurs DNS sont obtenus via DHCP.* • *"SearchDomain": Domaine(s) à rechercher si le nom d'hôte n'est pas totalement qualifié.* • *"DNSFromDHCP": Liste de serveurs DNS obtenue via DHCP dans le cas où FromDHCP est égal à True (Vrai). Cela signifie que les adresses résolues dans le champ DNSFromDHCP proviennent de DHCP et décrivent le statut de configuration.* • *"DNSManual": Liste des serveurs DNS manuellement spécifiés* xs:boolean **FromDHCP** [1][1] xs:token **SearchDomain** [0][non limité] tt:IPAddress **DNSFromDHCP** [0][non limité] tt:IPAddress **DNSManual** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.5 Définition des paramètres DNS

Cette opération permet de définir les paramètres DNS d'un dispositif comme décrit dans le Tableau 17. Les configurations DNS du dispositif doivent pouvoir être définies grâce à la commande SetDNS.

Tableau 17 – Commande SetDNS

SetDNS	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetDNSRequest	*Ce message contient:* • *"FromDHCP": True (Vrai) si les serveurs DNS sont obtenus via DHCP* • *"SearchDomain": Domaine(s) à rechercher si le nom d'hôte n'est pas totalement qualifié.* • *"DNSManual": Liste des serveurs DNS manuellement spécifiés* xs:boolean **FromDHCP** [1][1] xs:token **SearchDomain** [0][non limité] tt:IPAddress **DNSManual** [0][non limité]
SetDNSResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*

8.3.6 Obtention des paramètres NTP

Cette opération permet d'obtenir les paramètres NTP d'un dispositif comme décrit dans le Tableau 18. Si le dispositif prend en charge NTP, les paramètres de serveur NTP doivent pouvoir être obtenus grâce à la commande GetNTP.

Tableau 18 – Commande GetNTP

GetNTP	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetNTPRequest	*Ceci est un message vide.*
GetNTPResponse	*Ce message contient:* • *"FromDHCP": True (Vrai) si les serveurs NTP sont obtenus via DHCP.* • *"NTPFromDHCP": Liste de serveurs NTP obtenue via DHCP dans le cas où FromDHCP est égal à True (Vrai). Cela signifie que les adresses de serveur NTP dans le champ NTPFromDHCP proviennent de DHCP et décrivent le statut de configuration actuel.* • *"NTPManual": Liste des serveurs NTP manuellement spécifiés* xs:boolean **FromDHCP** [1][1] tt:NetworkHost **NTPFromDHCP** [0][non limité] tt:NetworkHost **NTPManual** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.7 Définition des paramètres NTP

Cette opération permet de définir les paramètres NTP d'un dispositif comme décrit dans le Tableau 19. Si la prise en charge de NTP est signalée via la fonctionnalité NTP, les paramètres de serveur NTP doivent pouvoir être définis grâce à la commande SetNTP.

Un dispositif doit accepter les chaînes formatées selon 2.1 de la norme RFC 1123; les autres chaînes doivent être considérées comme non valides.

Les modifications apportées à la liste de serveurs NTP ne doivent pas influencer le mode horloge DateTimeType. Utiliser SetSystemDateAndTime pour activer l'opération NTP.

Tableau 19 – Commande SetNTP

SetNTP	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetNTPRequest	*Ce message contient:* • *"FromDHCP": True (Vrai) si les serveurs NTP sont obtenus via DHCP.* • *"NTPManual": Liste des serveurs NTP manuellement spécifiés lorsqu'ils ne sont pas obtenus via DHCP.* xs:boolean **FromDHCP** [1][1] tt:NetworkHost **NTPManual** [0][non limité]
SetNTPResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidDnsName	*Le nom de serveur NTP suggéré est non valide.*
env:Sender ter:InvalidArgVal ter:TimeSyncedToNtp	*Le DateTimeType actuel exige un serveur NTP.*

8.3.8 Obtention des paramètres DNS dynamiques

Cette opération permet d'obtenir les paramètres DNS dynamiques d'un dispositif comme décrit dans le Tableau 20. Si le dispositif prend en charge un DNS dynamique comme spécifié dans la norme [RFC 2136] et la norme [RFC 4702], les types, nom et TTL doivent pouvoir être obtenus grâce à la commande GetDynamicDNS.

Tableau 20 – Commande GetDynamicDNS

GetDynamicDNS	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDynamicDNSRequest	*Ceci est un message vide.*
GetDynamicDNSResponse	*Ce message contient:* • *"Type": Type de mise à jour. Il existe trois types possibles: le dispositif ne veut pas de mise à jour (NoUpdate), le dispositif souhaite que le serveur DHCP effectue une mise à jour (ServerUpdates) et le dispositif effectue lui-même la mise à jour (ClientUpdates).* • *"Name": Nom de DNS au cas où le dispositif effectue la mise à jour.* • *"TTL": Durée de vie.* *tt:DynamicDNSType **Type** [1][1]* *tt:DNSName **Name** [0][1]* *xs:duration **TTL** [0][1]*
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.9 Définition des paramètres DNS dynamiques

Cette opération permet de définir les paramètres DNS dynamiques d'un dispositif comme décrit dans le Tableau 21. Si le dispositif prend en charge le DNS dynamique comme spécifié dans la norme [RFC 2136] et la norme [RFC 4702], les types, nom et TTL doivent pouvoir être définis grâce à la commande SetDynamicDNS.

Tableau 21 – Commande SetDynamicDNS

SetDynamicDNS	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetDynamicDNSRequest	*Ce message contient:* • *"Type": Type de mise à jour. Il existe trois types possibles: le dispositif ne veut pas de mise à jour (NoUpdate), le dispositif souhaite que le serveur DHCP effectue une mise à jour (ServerUpdates) et le dispositif effectue lui-même la mise à jour (ClientUpdates).* • *"Name": Nom de DNS au cas où le dispositif effectue la mise à jour.* • *"TTL": Durée de vie.* *tt:DynamicDNSType* ***Type*** *[1][1]* *tt:DNSName* ***Name*** *[0][1]* *xs:duration* ***TTL*** *[0][1]*
SetDynamicDNSResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.10 Obtention de configuration d'interface réseau

Cette opération permet d'obtenir la configuration d'interface réseau d'un dispositif comme décrit dans le Tableau 22. Le dispositif doit prendre en charge le retour des paramètres de configuration d'interface réseau tel que défini par le type NetworkInterface grâce à la commande GetNetworkInterfaces.

Tableau 22 – Commande GetNetworkInterfaces

GetNetworkInterfaces	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetNetworkInterfacesRequest	*Ceci est un message vide.*
GetNetworkInterfacesResponse	*Ce message contient un ensemble d'interfaces réseau de dispositif.* tt:NetworkInterface **NetworkInterfaces** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.11 Définition de configuration d'interface réseau

Cette opération permet de définir la configuration d'interface réseau d'un dispositif comme décrit dans le Tableau 23. Le dispositif doit prendre en charge la configuration réseau d'interfaces réseau prises en charge grâce à la commande SetNetworkInterfaces.

Si un dispositif répond par un RebootNeeded défini sur False (faux), le dispositif peut être atteint via la nouvelle adresse IP sans action supplémentaire. Il convient que le client ait connaissance qu'un dispositif peut ne pas réagir pendant un court instant, jusqu'à ce qu'il signale la disponibilité à la nouvelle adresse via les messages de découverte Hello tel que défini en 7.3.2.

Si un dispositif répond par un RebootNeeded défini sur True (vrai), il devient disponible sous son ancienne adresse IP. Les paramètres ne sont activés que lorsque le dispositif redémarre via la commande SystemReboot.

Pour assurer l'interopérabilité avec un client n'ayant pas connaissance de l'extension IEEE 802.11, un dispositif doit conserver sa configuration IEEE 802.11 si l'élément de configuration IEEE 802.11 est absent de la demande.

Tableau 23 – Commande SetNetworkInterfaces

SetNetworkInterfaces	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetNetworkInterfaces-Request	*Ce message contient:* • *"InterfaceToken": Jeton de l'interface réseau cible.* • *"NetworkInterface": Interface réseau à configurer.* tt:ReferenceToken **InterfaceToken** [1][1] tt:NetworkInterfaceSetConfiguration **NetworkInterface** [1][1]
SetNetworkInterfaces-Response	*Ce message contient:* • *"RebootNeeded": Indique si un redémarrage est nécessaire en cas de modifications des paramètres réseau.* xs:boolean RebootNeeded [1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*Le jeton d'interface réseau fourni n'existe pas.*
env:Sender ter:InvalidArgVal ter:InvalidMtuValue	*La valeur MTU est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceSpeed	*La vitesse suggérée n'est pas prise en charge.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceType	*Le type d'interface réseau suggéré n'est pas pris en charge.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*La configuration IEEE 802.11 n'est pas prise en charge.*
env:Sender ter:InvalidArgVal ter:InvalidSecurityMode	*Le mode de sécurité sélectionné n'est pas pris en charge.*
env:Sender ter:InvalidArgVal ter:InvalidStationMode	*Le mode de station sélectionné n'est pas pris en charge.*
env:Sender ter:InvalidArgVal ter:MissingDot11	*La configuration de sécurité ne contient pas de valeur IEEE 802.11.*
env:Sender ter:InvalidArgVal ter:MissingPSK	*La configuration de sécurité ne contient pas de valeur PSK.*
env:Sender ter:InvalidArgVal ter:MissingDot1X	*La valeur IEEE 802.1X de la configuration de sécurité est absente ou inexistante.*
env:Sender ter:InvalidArgVal ter:IncompatibleDot1X	*La valeur IEEE 802.1X de la configuration de sécurité n'est pas compatible avec l'interface réseau.*
env:Receiver ter:ActionNotSupported ter:InvalidDHCPv6	*Le mode DHCPv6 avec état demandé n'est pas pris en charge.*

8.3.12 Obtention des protocoles réseau

Cette opération permet d'obtenir des protocoles réseau définis d'un dispositif comme décrit dans le Tableau 24. Le dispositif doit prendre en charge la commande GetNetworkProtocols retournant les protocoles réseau configurés.

Tableau 24 – Commande GetNetworkProtocols

GetNetworkProtocols	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetNetworkProtocolsRequest	*Ceci est un message vide.*
GetNetworkProtocolsResponse	*Ce message retourne un ensemble de protocoles définis pris en charge par le dispositif. Il existe trois protocoles définis, HTTP, HTTPS et RTSP. Les paramètres suivants peuvent être extraits pour chaque protocole:* • *Port* • *Activer/désactiver* tt:NetworkProtocol **NetworkProtocols** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.13 Définition des protocoles réseau

Cette opération permet de configurer des protocoles réseau définis d'un dispositif comme décrit dans le Tableau 25. Le dispositif doit prendre en charge la configuration de protocoles réseau définis grâce à la commande SetNetworkProtocols.

Tableau 25 – Commande SetNetworkProtocols

SetNetworkProtocols	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetNetworkProtocols-Request	*Ce message configure un ou plusieurs protocoles réseau définis pris en charge par le dispositif. Il existe actuellement trois protocoles définis, HTTP, HTTPS et RTSP. Les paramètres suivants peuvent être définis pour chaque protocole:* • *Port* • *Activer/désactiver* tt:NetworkProtocol **NetworkProtocols** [1][non limité]
SetNetworkProtocolsResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:ServiceNotSupported	*Le service réseau fourni n'est pas pris en charge.*
env:Sender ter:InvalidArgVal ter:PortAlreadyInUse	*Le port sélectionné est déjà utilisé.*
env:Receiver ter:ActionNotSupported ter:EnablingTlsFailed	*Le dispositif ne prend pas en charge la TLS ou la TLS n'est pas correctement configurée.*

8.3.14 Obtention de passerelle par défaut

Cette opération permet d'obtenir les paramètres de passerelle par défaut d'un dispositif comme décrit dans le Tableau 26. Le dispositif doit prendre en charge la commande GetNetworkDefaultGateway retournant la/les adresse(s) de passerelle par défaut configurée(s).

Tableau 26 – Commande GetNetworkDefaultGateway

GetNetworkDefaultGateway	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetNetworkDefaultGateway-Request	*Ceci est un message vide.*
GetNetworkDefaultGateway-Response	*Ce message contient:* • *"IPv4Address": Adresse(s) de passerelle IPv4 par défaut.* • *"IPv6Address": Adresse(s) de passerelle IPv6 par défaut.* tt:IPv4Address **IPv4Address** [0][non limité] tt:IPv6Address **IPv6Address** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.15 Définition de passerelle par défaut

Cette opération permet de définir les paramètres de passerelle par défaut d'un dispositif comme décrit dans le Tableau 27. Le dispositif doit prendre en charge la configuration de passerelle par défaut grâce à la commande SetNetworkDefaultGateway.

Tableau 27 – Commande SetNetworkDefaultGateway

SetNetworkDefaultGateway	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetNetworkDefaultGateway-Request	*Ce message contient:* • *"IPv4Address": Adresse(s) de passerelle IPv4 par défaut.* • *"IPv6Address": Adresse(s) de passerelle IPv6 par défaut.* tt:IPv4Address **IPv4Address** [0][non limité] tt:IPv6Address **IPv6Address** [0][non limité]
SetNetworkDefaultGateway-Response	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidGatewayAddress	*L'adresse de passerelle fournie était non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*

8.3.16 Obtention de configuration zéro

Cette opération permet d'obtenir la configuration zéro d'un dispositif comme décrit dans le Tableau 28. Si le dispositif prend en charge la configuration IP dynamique conformément à la norme [RFC 3927], il doit prendre en charge le retour de l'adresse et du statut de configuration zéro IPv4 grâce à la commande GetZeroConfiguration.

Tableau 28 – Commande GetZeroConfiguration

GetZeroConfiguration	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetZeroConfigurationRequest	*Ceci est un message vide.*
GetZeroConfigurationResponse	*Ce message contient:* • *"InterfaceToken": Jeton de l'interface réseau* • *"Enabled": Indique si la configuration zéro est activée ou non.* • *"Addresses": Adresse(s) de configuration zéro IPv4.* tt:ReferenceToken **InterfaceToken** [1][1] xs:boolean **Enabled** [1][1] tt:IPv4Addresses **Address** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.17 Définition de configuration zéro

Cette opération permet de définir la configuration zéro du dispositif comme décrit dans le Tableau 29. Si le dispositif prend en charge la configuration IP dynamique conformément à la norme [RFC 3927], il doit prendre en charge la configuration d'adresse et de statut de configuration zéro IPv4 grâce à la commande SetZeroConfiguration.

Tableau 29 – Commande SetZeroConfiguration

SetZeroConfiguration	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetZeroConfigurationRequest	*Ce message contient:* • *"InterfaceToken": Jeton de l'interface réseau cible.* • *"Enabled": Indique si la configuration zéro est activée ou non.* tt:ReferenceToken **InterfaceToken** [1][1] xs:boolean **Enabled** [1][1]
SetZeroConfigurationResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*Le jeton d'interface réseau fourni n'existe pas.*

8.3.18 Obtention de filtre d'adresse IP

Cette opération permet d'obtenir les paramètres de filtre d'adresse IP d'un dispositif comme décrit dans le Tableau 30. Si le dispositif prend en charge le contrôle d'accès de dispositif basé sur des règles de filtrage IP (plages d'adresses IP rejetées ou acceptées), il doit prendre en charge la commande GetIPAddressFilter.

Tableau 30 – Commande GetIPAddressFilter

GetIPAddressFilter	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetIPAddressFilterRequest	*Ceci est un message vide.*
GetIPAddressFilterResponse	*Ce message contient:* • *"Type": Définit s'il convient que le filtre refuse ou autorise l'accès.* • *"IPv4Address": Adresse(s) de filtre IPv4* • *"IPv6Address": Adresse(s) de filtre IPv6* tt:IPAddressFilterType **Type** [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][non limité] tt:PrefixedIPv6Address **IPv6Address** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.3.19 Définition de filtre d'adresse IP

Cette opération permet de définir les paramètres de filtre d'adresse IP d'un dispositif comme décrit dans le Tableau 31. Si le dispositif prend en charge le contrôle d'accès de dispositif basé sur des règles de filtrage IP (plages d'adresses IP rejetées ou acceptées), il doit prendre en charge la configuration de règles de filtrage IP grâce à la commande SetIPAddressFilter.

Tableau 31 – Commande SetIPAddressFilter

SetIPAddressFilter	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetIPAddressFilterRequest	*Ce message contient:* • *"Type": Définit s'il convient que le filtre refuse ou autorise l'accès.* • *"IPv4Address": Adresse(s) de filtre IPv4* • *"IPv6Address": Adresse(s) de filtre IPv6* tt:IPAddressFilterType **Type** [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][non limité] tt:PrefixedIPv6Address **IPv6Address** [0][non limité]
SetIPAddressFilterResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*

8.3.20 Ajout d'une adresse de filtre IP

Cette opération permet d'ajouter une adresse de filtre IP à un dispositif comme décrit dans le Tableau 32. Si le dispositif prend en charge le contrôle d'accès de dispositif basé sur des règles de filtrage IP (plages d'adresses IP rejetées ou acceptées), il doit prendre en charge l'ajout d'adresses de filtrage IP grâce à la commande AddIPAddressFilter.

Le dispositif doit ignorer la valeur du champ Type. Utiliser la commande SetIPAddressFilter pour définir le type.

Tableau 32 – Commande AddIPAddressFilter

AddIPAddressFilter	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
AddIPAddressFilterRequest	*Ce message contient:* • *"Type": Définit s'il convient que le filtre refuse ou autorise l'accès.* • *"IPv4Address": Adresse(s) de filtre IPv4* • *"IPv6Address": Adresse(s) de filtre IPv6* tt:IPAddressFilterType Type [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][non limité] tt:PrefixedIPv6Address **IPv6Address** [0][non limité]
AddIPAddressFilterResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:IPFilterListIsFull	*Aucun filtre IP supplémentaire ne peut être ajouté, car la liste de filtres IP est pleine.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*

8.3.21 Suppression d'une adresse de filtre IP

Cette opération permet de supprimer une adresse de filtre IP d'un dispositif comme décrit dans le Tableau 33. Si le dispositif prend en charge le contrôle d'accès de dispositif basé sur des règles de filtrage IP (plages d'adresses IP rejetées ou acceptées), il doit prendre en charge la suppression d'adresses de filtrage IP grâce à la commande RemoveIPAddressFilter.

Le dispositif doit ignorer la valeur du champ Type.

Tableau 33 – Commande RemoveIPAddressFilter

RemoveIPAddressFilter	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
RemoveIPAddressFilterRequest	*Ce message contient:* • *"Type": La valeur de ce champ est ignorée dans cette commande.* • *"IPv4Address": Adresse(s) de filtre IPv4* • *"IPv6Address": Adresse(s) de filtre IPv6* tt:IPAddressFilterType Type [1][1] tt:PrefixedIPv4Address **IPv4Address** [0][non limité] tt:PrefixedIPv6Address **IPv6Address** [0][non limité]
RemoveIPAddressFilterResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*L'adresse IPv6 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*L'adresse IPv4 suggérée est non valide.*
env:Sender ter:InvalidArgVal ter:NoIPv6Address	*L'adresse IPv6 à supprimer n'existe pas.*
env:Sender ter:InvalidArgVal ter:NoIPv4Address	*L'adresse IPv4 à supprimer n'existe pas.*

8.3.22 Configuration IEEE 802.11

8.3.22.1 Généralités

Les exigences de 8.3.22 concernent uniquement les dispositifs signalant une prise en charge IEEE 802.11 via leur fonctionnalité réseau Dot11Configuration. Dans 8.3.22, le terme "le dispositif" est utilisé pour indiquer un dispositif avec prise en charge IEEE 802.11.

Le dispositif doit prendre en charge la configuration IEEE 802.11 et doit, en réponse à la méthode GetNetworkInterfaces, renvoyer "ieee80211 (71)" comme IANA-IfTypes pour la/les interface(s) 802.11.

Un dispositif ne doit pas renvoyer d'élément de liaison dans la réponse GetNetworkInterfaces et doit ignorer les éléments Link de la demande SetNetworkInterfaces.

Il convient que le dispositif prenne en charge le fait que chaque interface réseau IEEE 802.11 puisse être associée à plusieurs autres configurations IEEE 802.11.

La configuration IEEE 802.11 est prise en charge grâce à un élément de configuration IEEE 802.11 facultatif de l'élément de configuration d'obtention et de définition de réseau. Les informations suivantes sont traitées:

- SSID,
- Mode de station,

- Configuration de réseau sans fil multiple,
- Configuration de sécurité.

Les opérations suivantes permettent de faciliter la gestion de la configuration sans fil:

- Obtention des fonctionnalités IEEE 802.11,
- Obtention du statut IEEE 802.11,
- Balayage de réseaux IEEE 802.11 disponibles.

8.3.22.2 SSID

Le dispositif doit prendre en charge la configuration du SSID.

8.3.22.3 Mode de station

Le dispositif doit prendre en charge le mode de station de l'infrastructure.

Le dispositif peut prendre en charge le mode de station de réseau ad hoc. La configuration réelle nécessaire du mode de station de réseau ad hoc, y compris la configuration manuelle du nombre de voies, ne relève pas du domaine d'application du présent document. Néanmoins, pour tenir compte des dispositifs qui prennent en charge les modes de station de réseau ad hoc, la spécification permet de sélectionner (et de signaler) ce mode.

8.3.22.4 Configuration de réseau sans fil multiple

Chaque configuration IEEE 802.11 doit être identifiée par un alias (identifiant). L'alias doit être unique dans une configuration d'interface réseau. Le client doit fournir l'alias dans la demande SetNetworkInterfaces. Si le client souhaite mettre à jour une configuration sans fil existante, le même alias doit être utilisé. Une configuration sans fil, incluant l'alias, doit uniquement exister s'il s'agit d'une partie d'une configuration d'interface réseau.

Pour que le dispositif soit en mesure de hiérarchiser les priorités entre plusieurs configurations IEEE 802.11 alternatives, une valeur de priorité facultative peut être utilisée, une valeur plus élevée indiquant une priorité plus importante. Si plusieurs configurations sans fil présentent la même valeur de priorité, l'ordre de ces configurations n'est pas défini.

L'algorithme utilisé par le dispositif pour activer un réseau IEEE 802.11 selon une liste hiérarchique de configurations IEEE 802.11 ne relève pas du domaine d'application du présent document.

8.3.22.5 Configuration de sécurité

8.3.22.5.1 Généralités

La configuration de sécurité contient le mode de sécurité choisi et la configuration nécessaire pour ce mode. Les modes de sécurité suivants sont pris en charge:

- Néant,
- PSK (Pre Shared Key – Clé prépartagée) (WPA- et WPA2-Personnel),
- IEEE 802.1X-2004 (WPA- et WPA2-Enterprise).

La configuration du mode de sécurité WEP ne relève pas du domaine d'application du présent document. Néanmoins, pour tenir compte des dispositifs qui prennent en charge le mode de sécurité WEP, le présent document permet de sélectionner (et de signaler) ce mode.

Pour assurer la confidentialité et l'intégrité des données, le dispositif doit, conformément à la spécification [IEEE 802.11-2007], prendre en charge l'algorithme CCMP et peut prendre en charge l'algorithme TKIP.

L'algorithme peut être sélectionné manuellement (CCMP, TKIP) ou automatiquement (Tous). En mode de sélection manuelle, le même algorithme doit être utilisé pour le chiffrement par paire et par groupe. Pour pouvoir prendre en charge d'autres algorithmes, une valeur "étendue" est disponible.

Le dispositif doit prendre en charge les modes de sélection manuelle et automatique.

8.3.22.5.2 Mode néant

Le dispositif doit prendre en charge le mode de sécurité "Néant".

8.3.22.5.3 Mode PSK

Le dispositif doit prendre en charge le mode de sécurité "PSK".

Pour réduire le plus possible les risques de compromettre la PSK, il convient que le dispositif ne transmette pas de PSK à un client. De plus, il ne doit pas renvoyer la PSK dans une réponse à un appel d'opération GetNetworkInterfaces.

Pour ajouter une configuration sans fil avec le mode de sécurité PSK, les règles suivantes s'appliquent:

- Un client doit inclure une valeur PSK dans la demande SetNetworkInterfaces.
- Le dispositif doit vérifier qu'une valeur PSK a été fournie. Si ce n'est pas le cas, il doit renvoyer une erreur.

Pour mettre à jour une configuration sans fil avec le mode de sécurité PSK, les règles suivantes s'appliquent:

- Si le client souhaite conserver la valeur PSK, il convient de ne pas l'inclure dans la demande SetNetworkInterfaces.
- Le dispositif qui reçoit une demande SetNetworkInterfaces sans valeur PSK doit conserver sa valeur PSK.

La norme [IEEE 802.11-2007] stipule qu'il convient de distribuer la PSK au STA selon une méthode hors bande.

8.3.22.5.4 Mode IEEE 802.1X-2004

Il convient que le dispositif prenne en charge le mode de sécurité IEEE 802.1X. Pour obtenir des exigences plus détaillées sur le mode de sécurité IEEE 802.1X-2004, voir [configuration IEEE 802.1X].

8.3.22.6 Obtention des fonctionnalités Dot11

Cette opération renvoie les fonctionnalités IEEE 802.11, voir le Tableau 34 et le Tableau 35. Le dispositif doit prendre en charge cette opération.

Tableau 34 – GetDot11Capabilities

GetDot11Capabilities	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDot11CapabilitiesRequest	*Ceci est un message vide*
GetDot11CapabilitesResponse	*Ce message contient:* *tt:Dot11Capabilities* ***Capabilities*** *[1][1]*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*La configuration IEEE 802.11 n'est pas prise en charge.*

Tableau 35 – Fonctionnalités IEEE 802.11

Fonctionnalité	Description
TKIP	Indique si le dispositif prend en charge l'algorithme TKIP.
ScanAvailableNetworks	Indique si le dispositif prend en charge le balayage de réseaux IEEE 802.11 disponibles.
MultipleConfiguration	Indique si le dispositif prend en charge plusieurs configurations IEEE 802.11 alternatives.
AdHocStationMode	Indique si le dispositif prend en charge le mode de station ad hoc.
WEP	Indique si le dispositif prend en charge le mode de sécurité WEP.

8.3.22.7 Obtention du statut IEEE 802.11

Cette opération renvoie le statut d'une interface réseau sans fil. Le dispositif doit prendre en charge cette commande comme décrit dans le Tableau 36. Les statuts suivants peuvent être renvoyés:

- SSID (obligatoire),
- BSSID (recommandé),
- Chiffrement par paire (recommandé),
- Chiffrement par groupe (recommandé),
- Force du signal (recommandé),
- Alias de la configuration sans fil active (obligatoire).

Tableau 36 – GetDot11Status

GetDot11Status	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDot11StatusRequest	*Ce message contient:* *tt:ReferenceToken* ***InterfaceToken*** *[1][1]*
GetDot11StatusResponse	*Ce message contient:* *tt:Dot11Status* ***Status*** *[1][1]*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*La configuration IEEE 802.11 n'est pas prise en charge.*
env:Sender ter:InvalidArgVal ter:NotDot11	*L'interface n'est pas une interface IEEE 802.11.*
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*Le jeton d'interface réseau fourni n'existe pas.*
env:Receiver ter:Action ter:NotConnectedDot11	*Le réseau IEEE 802.11 n'est pas connecté.*

8.3.22.8 Balayage de réseaux IEEE 802.11 disponibles

Cette opération renvoie une liste des réseaux sans fil dans la plage du dispositif. Il convient qu'un dispositif prenne en charge cette opération comme décrit dans le Tableau 37. Les statuts suivants peuvent être renvoyés pour chaque réseau:

- SSID (obligatoire),
- BSSID (recommandé),
- Authentification et suite(s) de gestion de clé (recommandé),
- Chiffrement(s) par paire (recommandé),
- Chiffrement(s) par groupe (recommandé),
- Force du signal (recommandé).

Tableau 37 – ScanAvailableDot11Networks

ScanAvailableDot11Networks	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
ScanAvailableDot11-NetworksRequest	*Ce message contient:* *tt:ReferenceToken* ***InterfaceToken*** *[1][1]*
ScanAvailableDot11-NetworksResponse	*Ce message contient:* *tt:Dot11AvailableNetworks* ***Networks*** *[0][non limité]*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*La configuration IEEE 802.11 n'est pas prise en charge.*
env:Sender ter:InvalidArgVal ter:NotDot11	*L'interface n'est pas une interface IEEE 802.11.*
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*Le jeton d'interface réseau fourni n'existe pas.*
env;Receiver ter:ActionNotSupported ter:NotScanAvailable	*ScanAvailableDot11Networks n'est pas pris en charge.*

8.4 Système

8.4.1 Informations de dispositif

Cette opération permet d'obtenir des informations de dispositif, telles que le fabricant, le modèle et la version de micrologiciel, à partir d'un dispositif. Le dispositif doit prendre en charge le retour d'informations de dispositif grâce à la commande GetDeviceInformation comme décrit dans le Tableau 38.

Tableau 38 – Commande GetDeviceInformation

GetDeviceInformation		Classe d'accès: READ_SYSTEM
Nom de message	**Description**	
GetDeviceInformationRequest	*Ceci est un message vide.*	
GetDeviceInformationResponse	*Ce message contient:* xs:string **Manufacturer** [1][1] xs:string **Model** [1][1] xs:string **FirmwareVersion** [1][1] xs:string **SerialNumber** [1][1] xs:string **HardwareId** [1][1]	
Codes de défaut	**Description**	
	Pas de défauts spécifiques à la commande!	

8.4.2 Obtention des URI du système

Cette opération permet d'extraire les URI à partir desquels les informations système peuvent être téléchargées à l'aide de HTTP. Les URI peuvent être renvoyés pour les informations système suivantes:

- Journaux système. Plusieurs journaux système de types différents peuvent être renvoyés. Le format exact des journaux système ne relève pas du domaine d'application du présent document.
- Informations de prise en charge. Il s'agit d'informations de diagnostic de dispositif arbitraires provenant d'un dispositif. Le format exact des informations de diagnostic ne relève pas du domaine d'application du présent document.
- Sauvegarde du système. Le fichier reçu est un fichier de sauvegarde qui peut être utilisé pour restaurer ultérieurement la configuration en cours du dispositif. Le format exact du fichier de configuration de sauvegarde ne relève pas du domaine d'application du présent document.

Si le dispositif permet d'extraire les journaux système, les informations de prise en charge ou les données de sauvegarde du système, il convient de les mettre à disposition via HTTP GET. Si c'est le cas, il doit prendre en charge la commande GetSystemUris comme décrit dans le Tableau 39.

Tableau 39 – Commande GetSystemUris

GetSystemUris		Classe d'accès: READ_SYSTEM
Nom de message	**Description**	
GetSystemUrisRequest	*Ceci est un message vide.*	
GetSystemUrisResponse	*Ce message contient les URI à partir desquels les différents composants d'information du système peuvent être téléchargés.* tt:SystemLogUriList **SystemLogUris [0][1]** xs:anyURI **SupportInfoUri [0][1]** xs:anyURI **SystemBackupUri [0][1]**	
Codes de défaut	**Description**	
	Pas de défauts spécifiques à la commande!	

8.4.3 Sauvegarde

Cette opération permet d'extraire le ou les fichiers de configuration de sauvegarde système d'un dispositif comme décrit dans le Tableau 40. Il convient que le dispositif prenne en charge le retour de fichier(s) de configuration de sauvegarde grâce à la commande GetSystemBackup. La sauvegarde est retournée avec une référence à un nom et un type mime conjointement avec des données binaires. Le format exact des fichiers de configuration de sauvegarde ne relève pas du domaine d'application du présent document.

Le ou les fichiers de configuration de sauvegarde doivent être transmis via MTOM [W3C SOAP-MTOM].

Tableau 40 – Commande GetSystemBackup

GetSystemBackup	Classe d'accès: READ_SYSTEM_SECRET
Nom de message	**Description**
GetSystemBackupRequest	*Ceci est un message vide.*
GetSystemBackupResponse	*Le message de réponse d'obtention de sauvegarde système contient le ou les fichiers de configuration de sauvegarde système.* tt:BackupFile **BackupFiles** [1][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.4.4 Restauration

Cette opération permet de restaurer le ou les fichiers de configuration de sauvegarde système qui ont été extraits d'un dispositif comme décrit dans le Tableau 41. Il convient que le dispositif prenne en charge la restauration du ou des fichiers de configuration de sauvegarde grâce à la commande RestoreSystem. Le format exact du ou des fichiers de configuration de sauvegarde ne relève pas du domaine d'application du présent document. Si la commande est prise en charge, elle doit accepter les fichiers de sauvegarde retournés par la commande GetSystemBackup.

Le ou les fichiers de configuration de sauvegarde doivent être transmis via MTOM [W3C SOAP-MTOM].

Tableau 41 – Commande RestoreSystem

RestoreSystem	Classe d'accès: UNRECOVERABLE
Nom de message	**Description**
RestoreSystemRequest	*Ce message contient le ou les fichiers de sauvegarde système.* tt:BackupFile **BackupFiles** [1][non limité]
RestoreSystemResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidBackupFile	*Le ou les fichiers de sauvegarde ne sont pas valides.*

8.4.5 Démarrage de la restauration du système

Cette opération permet de lancer une restauration du système à partir des données de configuration sauvegardées à l'aide du mécanisme HTTP POST. La réponse à la commande comprend une URL HTTP vers laquelle le fichier de sauvegarde peut être chargé. La restauration réelle a lieu à l'issue de l'opération HTTP POST. Il convient que les dispositifs prennent en charge la restauration du système grâce à la commande StartSystemRestore. Le format exact des données de configuration de sauvegarde ne relève pas du domaine d'application du présent document.

La restauration du système sur HTTP peut être assurée comme décrit dans le Tableau 42 selon la procédure suivante:

1) Le client appelle StartSystemRestore.
2) Le service de dispositif répond avec l'URI de chargement.
3) Le client transmet les données de configuration à l'URI de chargement à l'aide de HTTP POST.
4) Le serveur applique la configuration chargée, puis procède au redémarrage, si nécessaire.

Si la restauration du système n'aboutit pas car le fichier chargé n'est pas valide, la réponse HTTP POST doit être "415 Unsupported Media Type" (415 Type de support non pris en charge). Si la restauration du système n'aboutit pas du fait d'une erreur dans le dispositif, la réponse HTTP POST doit être "500 Internal Server Error" (500 Erreur du serveur interne).

La valeur de l'en-tête Content-Type de la demande HTTP POST doit être "application/octet-stream".

Tableau 42 – Commande StartSystemRestore

StartSystemRestore	Classe d'accès: UNRECOVERABLE
Nom de message	**Description**
StartSystemRestore-Request	*Ceci est un message vide*
StartSystemRestore-Response	*Ce message contient:* ▪ *Une URL vers laquelle le fichier de configuration du système peut être chargé.* ▪ *Une durée facultative indiquant combien de temps il est prévu que le dispositif soit indisponible à l'issue du chargement.* xs:anyURI **UploadUri [1][1]** xs:duration **ExpectedDownTime [0][1]**
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande.

8.4.6 Obtention des date et heure système

Cette opération permet d'obtenir la date et l'heure système du dispositif comme décrit dans le Tableau 43. Le dispositif doit prendre en charge le retour du réglage d'heure d'été et des date et heure système manuelles (le cas échéant) ou une indication de l'heure NTP (le cas échéant) grâce à la commande GetSystemDateAndTime.

Un dispositif doit fournir les informations UTCDateTime, même si l'élément est marqué comme étant facultatif, afin d'assurer la compatibilité en amont.

Tableau 43 – Commande GetSystemDateAndTime

GetSystemDateAndTime	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetSystemDateAndTime-Request	*Ceci est un message vide.*
GetSystemDateAndTime-Response	*Ce message contient les informations de date et d'heure du dispositif.* • *"DateTimeType": Indique si la date et l'heure système sont définies manuellement ou par NTP* • *"DaylightSavings": Heure d'été activée ou désactivée* • *"TimeZone": Fuseau horaire tel qu'il est défini dans POSIX 1003.1, section 8.3* • *"UTCDateTime": Date et heure en UTC.* • *"LocalDateTime": Date et heure locales du dispositif* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1] tt:DateTime **UTCDateTime** [0][1] tt:DateTime **LocalDateTime** [0][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.4.7 Définition des date et heure système

Cette opération permet de définir la date et l'heure système du dispositif comme décrit dans le Tableau 44. Le dispositif doit prendre en charge la configuration du réglage d'heure d'été et des date et heure système manuelles (le cas échéant) ou une indication de l'heure NTP (le cas échéant) grâce à la commande SetSystemDateAndTime. Le dispositif doit prendre en compte un fuseau horaire valide, conformément aux règles de la norme [IEEE 1003.1]:2004, Section 8.3.

Si les date et heure système sont définies manuellement, le client doit inclure UTCDateTime ou LocalDateTime dans la demande.

Il convient d'affecter la valeur "true" (vrai) au drapeau DayLightSavings afin d'activer les paramètres DST de la chaîne TimeZone. Supprimer le drapeau DayLightSavings s'il convient d'ignorer la partie DST des paramètres TimeZone.

Tableau 44 – Commande SetSystemDateAndTime

SetSystemDateAndTime	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetSystemDateAndTime-Request	*Ce message contient les informations de date et d'heure du dispositif.* • *"DateTimeType": Indique si la date et l'heure système sont définies manuellement ou par NTP* • *"DaylightSavings": Règle automatiquement l'heure d'été si définie dans TimeZone.* • *"TimeZone": Le fuseau horaire est défini dans POSIX 1003.1, section 8.3* • *"UTCDateTime": Date et heure en UTC. Si DateTimeType est NTP, UTCDateTime n'a aucune signification.* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1] tt:DateTime **UTCDateTime** [0][1]
SetSystemDateAndTime-Response	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidTimeZone	*Un fuseau horaire non valide a été spécifié.*
env:Sender ter:InvalidArgVal ter:InvalidDateTime	*Une date ou une heure non valide a été spécifiée.*
env:Sender ter:InvalidArgVal ter:NtpServerUndefined	*Impossible de remplacer DateTimeType par NTP, car aucun serveur NTP n'est défini.*

8.4.8 Réglages par défaut d'usine

Cette opération permet de réinitialiser les paramètres d'un dispositif à leurs valeurs par défaut d'usine comme décrit dans le Tableau 45. Le dispositif doit prendre en charge le réglage par défaut d'usine total et partiel grâce à la commande SetSystemFactoryDefault. La signification d'un réglage par défaut d'usine partiel est spécifique au produit et spécifique au fournisseur de dispositif. L'effet d'une opération de réglage par défaut d'usine partiel n'est pas totalement défini. Cependant, il doit être garanti qu'après une réinitialisation partielle, le dispositif soit accessible à la même adresse IP que celle utilisée avant la réinitialisation. Cela signifie que les paramètres de réseau de base, tels que les paramètres d'adresse IP, de sous-réseau et de passerelle ou DHCP, restent inchangés après la réinitialisation partielle.

Tableau 45 – Commande SetSystemFactoryDefault

SetSystemFactoryDefault	Classe d'accès: UNRECOVERABLE
Nom de message	**Description**
SetSystemFactoryDefault-Request	*Ce message contient les types de réglages par défaut d'usine à effectuer.* • *"Hard": Tous les paramètres sont définis à leurs valeurs par défaut d'usine* • *"Soft": Tous les paramètres, à l'exception des paramètres spécifiques au fournisseur de dispositif, sont définis à leurs valeurs par défaut d'usine* tt:FactoryDefaultType **FactoryDefault** [1][1]
SetSystemFactoryDefault-Response	*Ceci est un message vide.*
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.4.9 Mise à niveau du micrologiciel

Cette opération permet de mettre à niveau la version de micrologiciel d'un dispositif comme décrit dans le Tableau 46. Après une mise à niveau réussie, le message de réponse est envoyé avant le redémarrage du dispositif. Il convient que le dispositif prenne en charge la

mise à niveau de micrologiciel grâce à la commande UpgradeSystemFirmware. Le format exact des données de micrologiciel ne relève pas du domaine d'application du présent document.

Le micrologiciel doit être transmis via MTOM [W3C SOAP-MTOM].

Tableau 46 – Commande UpgradeSystemFirmware

UpgradeSystemFirmware	Classe d'accès: UNRECOVERABLE
Nom de message	**Description**
UpgradeSystemFirmware-Request	*Ce message contient le micrologiciel utilisé pour la mise à niveau. La mise à niveau de micrologiciel est "partielle", ce qui signifie que tous les paramètres conservent leur valeur actuelle.* tt:AttachmentData **Firmware** [1][1]
UpgradeSystemFirmware-Response	*Ce message contient une chaîne "Message" permettant au dispositif de renvoyer un message au client, par exemple "Mise à niveau réussie, redémarrage dans x secondes".* xs:string **Message** [1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgs ter:InvalidFirmware	*Le micrologiciel est non valide, c'est-à-dire qu'il n'est pas pris en charge par ce dispositif.*
env:Receiver ter:Action ter:FirmwareUpgrade-Refusé	*La mise à niveau de micrologiciel a échoué.*

8.4.10 Démarrage de la mise à niveau du micrologiciel

Cette opération permet de lancer une mise à niveau de micrologiciel à l'aide du mécanisme HTTP POST comme décrit dans le Tableau 47. La réponse à la commande comprend une URL HTTP vers laquelle le fichier de mise à niveau peut être chargé. La mise à niveau réelle a lieu à l'issue de l'opération HTTP POST. Il convient que le dispositif prenne en charge la mise à niveau de micrologiciel grâce à la commande StartFirmwareUpgrade. Le format exact des données de micrologiciel ne relève pas du domaine d'application du présent document.

La mise à niveau de micrologiciel sur HTTP peut être assurée selon la procédure suivante:

1) Le client appelle StartFirmwareUpgrade.
2) Le service de dispositif répond avec l'URI de chargement et une valeur de délai facultative.
3) Le client attend pendant le délai indiqué, s'il est spécifié par le serveur.
4) Le client transmet l'image de micrologiciel à l'URI de chargement à l'aide de HTTP POST.
5) Le serveur se reprogramme à l'aide de l'image chargée, puis redémarre.

Si la mise à niveau de micrologiciel n'aboutit pas, car le fichier de mise à niveau n'est pas valide, la réponse HTTP POST doit être "415 Unsupported Media Type" (415 Type de support non pris en charge). Si la mise à niveau de micrologiciel n'aboutit pas du fait d'une erreur dans le dispositif, la réponse HTTP POST doit être "500 Internal Server Error" (500 Erreur du serveur interne).

La valeur de l'en-tête Content-Type de la demande HTTP POST doit être "application/octet-stream".

Tableau 47 – Commande StartFirmwareUpgrade

StartFirmwareUpgrade	Classe d'accès: UNRECOVERABLE
Nom de message	**Description**
StartFirmwareUpgradeRequest	*Ceci est un message vide*
StartFirmwareUpgradeResponse	*Ce message contient:* ▪ *Une URL vers laquelle le fichier de micrologiciel peut être chargé.* ▪ *Un délai facultatif; le client doit attendre pendant la durée indiquée avant de procéder au chargement du micrologiciel.* ▪ *Une durée indiquant combien de temps il est prévu que le dispositif soit indisponible à l'issue du chargement du micrologiciel.* xs:anyURI **UploadUri [1][1]** xs:duration **UploadDelay [0][1]** xs:duration **ExpectedDownTime [0][1]**
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande.

8.4.11 Obtention des journaux système

Cette opération permet d'obtenir un journal système d'un dispositif comme décrit dans le Tableau 48. Il convient que le dispositif prenne en charge l'extraction d'informations de journal système grâce à la commande GetSystemLog. Le format exact des journaux système ne relève pas du domaine d'application du présent document.

Les informations de journal système doivent être transmises via MTOM [W3C SOAP-MTOM] ou sous la forme d'une chaîne.

Tableau 48 – Commande GetSystemLog

GetSystemLog	Classe d'accès: READ_SYSTEM_SECRET
Nom de message	**Description**
GetSystemLogRequest	*Ce message contient le type de journal système à extraire. Les informations de journal prises en charge sont définies dans deux types différents:* • *"System": Journal système* • *"Access": Journal d'accès client* tt:SystemLogType **LogType** [1][1]
GetSystemLogResponse	*Ce message contient les informations de journal système demandées. Le dispositif peut choisir de retourner les informations de journal système sous la forme de données binaires dans une pièce jointe ou sous la forme d'une chaîne commune.* tt:AttachmentData **Binary** [0][1] xs:string **String** [0][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgs ter:AccesslogUnavailable	*Aucune information de journal d'accès n'est disponible*
env:Sender ter:InvalidArgs ter:SystemlogUnavailable	*Aucune information de journal système n'est disponible*

8.4.12 Obtention d'informations de prise en charge

Cette opération permet d'obtenir des informations de diagnostic de dispositif arbitraires provenant d'un dispositif comme décrit dans le Tableau 49. Le dispositif peut prendre en charge l'extraction d'informations de diagnostic grâce à la commande GetSystemSupportInformation. Le format exact des informations de diagnostic ne relève pas du domaine d'application du présent document.

Les informations de diagnostic doivent être transmises sous la forme d'une pièce jointe via MTOM [W3C SOAP-MTOM] ou sous forme de chaîne.

Tableau 49 – Commande GetSystemSupportInformation

GetSystemSupportInformation	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetSystemSupport-InformationRequest	*Ceci est un message* vide.
GetSystemSupport-Information Response	*Le message contient les informations de prise en charge. Le dispositif peut choisir de retourner les informations de prise en charge sous la forme de données binaires ou d'une chaîne commune.* tt:AttachmentData **BinaryFormat** [0][1] xs:string **StringFormat** [0][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgs ter:SupportInformation-Unavailable	*Aucune information de prise en charge n'est disponible.*

8.4.13 Redémarrage

Cette opération permet de redémarrer un dispositif. Le message de réponse doit être envoyé avant le redémarrage du dispositif. Le dispositif doit prendre en charge le redémarrage grâce à la commande SystemReboot comme décrit dans le Tableau 50.

Tableau 50 – Commande SystemReboot

SystemReboot	Classe d'accès: ACTUATE
Nom de message	**Description**
SystemReboot	*Ceci est un message* vide.
SystemRebootResponse	*Ce message contient une chaîne "Message" permettant au dispositif de renvoyer un message au client, par exemple "Redémarrage dans x secondes".* Xs:string **Message** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.4.14 Obtention des paramètres de domaine d'application

Cette opération permet de demander les paramètres de domaine d'application d'un dispositif comme décrit dans le Tableau 51. Les paramètres de domaine d'application sont utilisés dans la découverte de dispositif pour mettre en correspondance un message de sonde, voir l'Article 7. Les paramètres de domaine d'application sont de deux types différents:

- fixes,
- configurables.

Les paramètres de domaine d'application fixes ne peuvent pas être modifiés par l'intermédiaire de l'interface de gestion de dispositif, mais ce sont des caractéristiques de dispositif permanentes faisant partie des configurations de micrologiciel. Le type de domaine d'application est indiqué dans la liste de domaines d'application retournée dans la réponse de demande de paramètres de domaine d'application. Les paramètres de domaine d'application configurables peuvent être définis par l'intermédiaire des opérations de définition et d'ajout de paramètres de domaine d'application, voir 8.4.15 et 8.4.16. Le dispositif doit prendre en charge l'extraction de paramètres de domaine d'application de découverte grâce à la commande GetScopes. Étant donné que certains paramètres de domaine d'application sont obligatoires, le client attend toujours une liste de domaines d'application dans la réponse.

Tableau 51 – Commande GetScopes

GetScopes	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetScopesRequest	Ceci est un message *vide*.
GetScopesResponse	*Le message de réponse de domaine d'application contient une liste d'URI définissant les domaines d'application du dispositif.* tt:Scope: **Scopes** [1][non limité]
Codes de défaut	**Description**
env:Receiver ter:Action ter:EmptyScope	*La liste de domaines d'application est vide.*

8.4.15 Définition des paramètres de domaine d'application

Cette opération permet de définir les paramètres de domaine d'application d'un dispositif comme décrit dans le Tableau 52. Les paramètres de domaine d'application sont utilisés dans la découverte de dispositif pour mettre en correspondance un message de sonde, voir l'Article 7.

Cette opération permet de remplacer tous les paramètres de domaine d'application configurables existants (paramètres non fixes). Si cela doit être évité, il convient d'utiliser plutôt la commande d'ajout de domaine d'application. Le dispositif doit prendre en charge la configuration de paramètres de domaine d'application de découverte grâce à la commande SetScopes.

Tableau 52 – Commande SetScopes

SetScopes	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetScopesRequest	*Le message de définition de domaine d'application contient une liste d'URI définissant le domaine d'application du dispositif.* Xs:anyURI: **Scopes** [1][non limité]
SetScopesResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:OperationProhibited ter:ScopeOverwrite	*Le paramètre de domaine d'application supplante les paramètres de domaine d'application de dispositif fixes. La commande est rejetée.*
env:Receiver ter:Action ter:TooManyScopes	*La liste de domaines d'application demandée dépasse le nombre de domaines d'application pris en charge.*

8.4.16 Ajout de paramètres de domaine d'application

Cette opération permet d'ajouter de nouveaux paramètres de domaine d'application configurables à un dispositif comme décrit dans le Tableau 53. Les paramètres de domaine d'application sont utilisés dans la découverte de dispositif pour mettre en correspondance un message de sonde, voir l'Article 7. Le dispositif doit prendre en charge l'ajout de paramètres de domaine d'application de découverte grâce à la commande AddScopes.

Tableau 53 – Commande AddScopes

AddScopes	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
AddScopesRequest	*Le message d'ajout de domaine d'application contient une liste d'URI à ajouter à la liste de domaines d'application configurables existants.* xs:anyURI:**ScopeItem** [1][non limité]
AddScopesResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Receiver ter:Action ter:TooManyScopes	*La liste de domaines d'application demandée dépasse le nombre de domaines d'application pris en charge.*

8.4.17 Suppression des paramètres de domaine d'application

Cette opération permet de supprimer des paramètres de domaine d'application configurables d'un dispositif comme décrit dans le Tableau 54. Les paramètres de domaine d'application sont utilisés dans la découverte de dispositif pour mettre en correspondance un message de sonde, voir l'Article 7. Le dispositif doit prendre en charge la suppression de paramètres de domaine d'application de découverte grâce à la commande RemoveScopes.

À noter que le message de réponse correspond toujours à la demande, sinon une erreur est renvoyée. Par conséquent, l'utilisation de la réponse est déconseillée.

Tableau 54 – Commande RemoveScopes

RemoveScopes	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
RemoveScopesRequest	*Le message de suppression de domaine d'application contient une liste d'URI qu'il convient de supprimer du domaine d'application du dispositif.* xs:anyURI: **ScopeItem** [1][non limité]
RemoveScopesResponse	*Le message de réponse de domaine d'application contient une liste d'URI qui ont été supprimés du domaine d'application du dispositif.* *xs:anyURI*: ScopeItem [0][non limité]
Codes de défaut	**Description**
env:Sender ter:OperationProhibited ter:FixedScope	*Tentative de suppression de paramètre de domaine d'application fixe. La commande est rejetée.*
env:Sender ter:InvalidArgVal ter:NoScope	*Tentative de suppression d'un domaine d'application qui n'existe pas.*

8.4.18 Obtention du mode de découverte

Cette opération permet d'obtenir le mode de découverte d'un dispositif comme décrit dans le Tableau 55. Voir 7.2 pour la définition des différents modes de découverte de dispositif. Le dispositif doit prendre en charge l'extraction des paramètres de mode de découverte grâce à la commande GetDiscoveryMode.

Tableau 55 – Commande GetDiscoveryMode

GetDiscoveryMode	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDiscoveryModeRequest	*Ceci est un message vide.*
GetDiscoveryModeResponse	*Ce message contient les paramètres de mode de découverte actuel, c'est-à-dire découvrable ou non découvrable.* tt:DiscoveryMode: **DiscoveryMode** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.4.19 Définition du mode de découverte

Cette opération permet de définir le mode de découverte d'un dispositif comme décrit dans le Tableau 56. Voir 7.2 pour la définition des différents modes de découverte de dispositif. Le dispositif doit prendre en charge la configuration des paramètres de mode de découverte grâce à la commande SetDiscoveryMode.

Tableau 56 – Commande SetDiscoveryMode

SetDiscoveryMode	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetDiscoveryModeRequest	*Ce message contient les paramètres de mode de découverte demandé, c'est-à-dire découvrable ou non découvrable.* tt:DiscoveryMode: **DiscoveryMode** [1][1]
SetDiscoveryMode-Response	*Ceci est un message vide.*
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5 Sécurité

8.5.1 Généralités

Le Paragraphe 8.5 contient un ensemble d'opérations de gestion de sécurité. Ces opérations sont sensibles aux attaques sur le réseau et doivent être protégées par un niveau d'autorisation approprié afin de ne pas mettre le dispositif en danger.

8.5.2 Obtention de la politique d'accès

Il convient que l'accès aux différents services et sous-ensembles de services soit soumis à un contrôle d'accès. Le Paragraphe 5.13 définit les conditions préalables pour l'authentification de point terminal. Les décisions d'autorisation peuvent alors être prises en utilisant une politique de sécurité d'accès. Le présent document ne spécifie aucun format de description de politique particulier ni aucune politique de sécurité particulière. C'est au fabricant de dispositif ou au fournisseur de système de choisir une politique et un format de description de politique. Cependant, une politique d'accès (dans un format arbitraire) peut être demandée à l'aide de cette commande. Si le dispositif prend en charge les paramètres de politique d'accès, il doit alors prendre en charge cette commande comme décrit dans le Tableau 57.

Tableau 57 – Commande GetAccessPolicy

GetAccessPolicy	Classe d'accès: READ_SYSTEM_SECRET
Nom de message	**Description**
GetAccessPolicyRequest	*Ceci est un message vide.*
GetAccessPolicyResponse	*Ce message contient le fichier de politique demandé.* tt:BinaryData **PolicyFile** [1][1]
Codes de défaut	**Description**
env:Receiver ter:Action ter:EmptyPolicy	*Le fichier de politique du dispositif n'existe pas ou est vide.*

8.5.3 Définition de la politique d'accès

Cette commande permet de définir la politique de sécurité d'accès du dispositif (pour plus de détails sur la politique de sécurité d'accès, voir la commande Get, 8.5.2). Si le dispositif prend en charge les paramètres de politique d'accès, il doit alors prendre en charge cette commande comme décrit dans le Tableau 58.

Tableau 58 – Commande SetAccessPolicy

SetAccessPolicy	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetAccessPolicyRequest	*Ce message contient le fichier de politique à définir.* tt:BinaryData **PolicyFile** [1][1]
SetAccessPolicyResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgs ter:PolicyFormat	*La politique demandée ne peut pas être définie en raison d'un format de politique inconnu.*

8.5.4 Obtention d'utilisateurs

Cette opération permet de répertorier les utilisateurs enregistrés ainsi que leurs niveaux d'utilisateur. Le dispositif doit prendre en charge l'extraction des utilisateurs de dispositif enregistrés grâce à la commande GetUsers comme décrit dans le Tableau 59.

De plus, un dispositif ne doit pas renvoyer les identifiants (mot de passe) dans la réponse.

Tableau 59 – Commande GetUsers

GetUsers	Classe d'accès: READ_SYSTEM_SECRET
Nom de message	**Description**
GetUsersRequest	*Ceci est un message vide.*
GetUsersResponse	*Ce message contient la liste des utilisateurs et leur niveau d'utilisateur correspondant. Chaque entrée comprend:* • *Le nom d'utilisateur* • *Le niveau d'utilisateur* *NOTE Le mot de passe n'est pas inclus dans la réponse, même s'il est présent dans le type tt:User.* tt:User: **User** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.5 Création d'utilisateurs

Cette opération permet de créer de nouveaux utilisateurs de dispositif avec leurs identifiants correspondants sur un dispositif à des fins d'authentification, voir 5.13 pour plus de détails. Le dispositif doit prendre en charge la création d'utilisateurs de dispositif et leurs identifiants à des fins d'authentification grâce à la commande CreateUsers, comme décrit dans le Tableau 60, tant que le nombre d'utilisateurs existants ne dépasse pas la valeur de fonctionnalité MaxUsers. Soit la création de tous les utilisateurs aboutit, soit un message de défaut doit être retourné sans créer d'utilisateur.

Il est recommandé que les dispositifs prennent en charge une longueur de mot de passe d'au moins 28 octets.

Tableau 60 – Commande CreateUsers

CreateUsers	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
CreateUsersRequest	*Ce message contient un élément de paramètres d'utilisateur pour un nouvel utilisateur. Chaque entrée d'utilisateur comprend:* • *Le nom d'utilisateur* • *Le mot de passe* • *Le niveau d'utilisateur* tt:User: **User** [1][non limité]
CreateUsersResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:OperationProhibited ter:UsernameClash	*Le nom d'utilisateur existe déjà*
env:Sender ter:OperationProhibited ter:PasswordTooLong	*Le mot de passe est trop long*
env:Sender ter:OperationProhibited ter:UsernameTooLong	*Le nom d'utilisateur est trop long*
env:Sender ter:OperationProhibited ter:Password	*Mot de passe trop faible.*
env:Receiver ter:Action ter:TooManyUsers	*Nombre maximal d'utilisateurs pris en charge dépassé.*
env:Sender ter:OperationProhibited ter:AnonymousNotAllowed	*Le niveau d'utilisateur anonyme n'est pas autorisé.*
env:Sender ter:OperationProhibited ter:UsernameTooShort	*Le nom d'utilisateur est trop court*

8.5.6 Suppression d'utilisateurs

Cette opération supprime des utilisateurs sur un dispositif comme décrit dans le Tableau 61. Le dispositif doit prendre en charge la suppression d'utilisateurs de dispositif et leurs identifiants à des fins d'authentification grâce à la commande DeleteUsers. Un dispositif peut avoir un ou plusieurs utilisateurs fixes qui ne peuvent pas être supprimés pour assurer l'accès à l'unité. Soit la suppression de tous les utilisateurs aboutit, soit un message de défaut doit être retourné et aucun utilisateur ne doit être supprimé.

Tableau 61 – Commande DeleteUsers

DeleteUsers	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
DeleteUsersRequest	*Ce message contient le nom de l'utilisateur ou des utilisateurs à supprimer.* xs:string: **Username** [1][non limité]
DeleteUsersResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:UsernameMissing	*Nom d'utilisateur non reconnu.*
env:Sender ter:InvalidArgVal ter:FixedUser	*Le nom d'utilisateur ne peut pas être supprimé*

8.5.7 Définition des paramètres d'utilisateur

Cette opération permet de mettre à jour les paramètres d'un ou de plusieurs utilisateurs sur un dispositif à des fins d'authentification, voir 5.13 pour plus de détails. Le dispositif doit prendre en charge la mise à jour d'utilisateurs de dispositif et leurs identifiants grâce à la commande SetUser comme décrit dans le Tableau 62. Soit le traitement de toutes les demandes de modification aboutit, soit un message de défaut doit être retourné et aucune demande de modification ne doit être traitée.

Si la valeur de mot de passe facultative est ignorée, le dispositif envisage d'effacer le mot de passe de l'utilisateur. Si le dispositif ne peut pas accepter un mot de passe ayant une longueur égale à zéro, le message de défaut "ter:PasswordTooWeak" est retourné.

Tableau 62 – Commande SetUser

SetUser	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetUserRequest	*Ce message contient une liste des utilisateurs et des paramètres correspondants à mettre à jour.* • *Le nom d'utilisateur* • *Le mot de passe* • *Le niveau d'utilisateur* tt:User: User [1][non limité]
SetUserResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:UsernameMissing	*Nom d'utilisateur non reconnu.*
env:Sender ter:OperationProhibited ter:PasswordTooLong	*Le mot de passe est trop long*
env:Sender ter:OperationProhibited ter:PasswordTooWeak	*Mot de passe trop faible.*
env:Sender ter:OperationProhibited ter:AnonymousNotAllowed	*Le niveau d'utilisateur anonyme n'est pas autorisé.*
env:Sender ter:InvalidArgVal ter:FixedUser	*Le mot de passe ou le niveau d'utilisateur ne peut pas être modifié.*

8.5.8 Configuration IEEE 802.1X

8.5.8.1 Généralités

Le présent document définit les paramètres ci-dessous sous la forme d'un ensemble de paramètres de configuration IEEE 802.1X.

- Jeton de configuration

  Ce paramètre indique un jeton de référence des paramètres de configuration IEEE 802.1X. Il est défini sous la forme 'Dot1XConfigurationToken' dans le schéma commun, voir l'Article B.4. Cette convention de dénomination de "Dot1X", qui représente en réalité "IEEE 802.1X", est utilisée pour assurer une meilleure lisibilité de l'élément de schéma dans le code source généré.

- Identité EAP

  Ce paramètre indique le nom d'utilisateur du demandeur qui se connecte au réseau géré IEEE 802.1X. Il est défini sous la forme "Identity" "Identity" dans le schéma commun, voir l'Article B.4.

- Méthode EAP

  Ce paramètre indique la méthode d'authentification utilisée. Il est défini sous la forme "EAPMethod" dans le schéma commun, voir l'Article B.4.

- ID de certificat CA

  Ce paramètre indique l'ID du certificat CA utilisé pour la vérification du serveur d'authentification. Il est défini sous la forme "CACertificateID" dans le schéma commun, voir l'Article B.4.

- Paramètres de configuration respectifs pour la méthode EAP sélectionnée

  Selon la méthode EAP sélectionnée, certains paramètres spécifiques sont nécessaires:

  – **[EAP-MD5], [EAP-PEAP/MSCHAP-V2], [types EAP-TTLS]**: Mot de passe d'identité permettant au serveur d'authentification de vérifier l'utilisateur (le dispositif) à l'aide du mot de passe spécifié. La méthode [EAP-MD5] ne s'applique pas à l'utilisation de 802.11 (WPA-Enterprise).

  – **[EAP-TLS]**: ID de certificat client permettant au serveur RADIUS de vérifier l'utilisateur (le dispositif) à l'aide du certificat spécifié.

Ces paramètres IEEE 802.1X sont appelés par la configuration de sécurité dans le cadre d'une certaine configuration d'interface réseau. Pour plus de détails, se reporter à 8.3.21.

Le présent document part du principe que la configuration IEEE 802.1X du dispositif est réalisée à l'extérieur du réseau géré IEEE 802.1X. En cas de reconfiguration des paramètres IEEE 802.1X, il est également pris par hypothèse que cette opération est réalisée à l'extérieur du réseau géré 802.1X.

À noter que la prise en charge d'IEEE 802.1X est limitée aux interfaces IEEE 802.11.

8.5.8.2 Création d'une configuration IEEE 802.1X

Cette opération permet de créer un nouvel ensemble de paramètres de configuration IEEE 802.1X du dispositif comme décrit dans le Tableau 63. Le dispositif doit prendre en charge cette commande si la prise en charge de IEEE 802.1X est signalée via la fonctionnalité Security Dot1X. Si le dispositif reçoit cette demande avec une spécification de jeton de configuration qui existe déjà (Dot1XConfigurationToken), il convient qu'il réponde par une erreur "*ter:ReferenceToken*" afin de signaler un conflit de configuration.

Tableau 63 – Commande CreateDot1XConfiguration

CreateDot1XConfiguration	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
CreateDot1XConfigurationRequest	*Ce message contient:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
CreateDot1XConfigurationResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:EAPMethodNotSupported	*La méthode EAP suggérée n'est pas prise en charge.*
env:Receiver ter:Action ter:MaxDot1X	*Nombre maximal de configurations IEEE 802.1X atteint.*
env:Sender ter:OperationProhibited ter:CertificateID	*Erreur d'ID de certificat non valide.*
env:Sender ter:InvalidArgVal ter:ReferenceToken	*Dot1XConfigurationToken existe déjà.*
env:Sender ter:InvalidArgVal ter:InvalidDot1X	*Configuration IEEE 802.1X non valide.*

8.5.8.3 Définition d'une configuration IEEE 802.1X

Si la commande CreateDot1XConfiguration tente de créer un nouvel ensemble de paramètres de configuration, cette opération modifie l'ensemble de paramètres de configuration IEEE 802.1X existants du dispositif comme décrit dans le Tableau 64. Le dispositif doit prendre en charge cette commande si la prise en charge de IEEE 802.1X est signalée via la fonctionnalité Security Dot1X.

Tableau 64 – Commande SetDot1XConfigurationRequest

SetDot1XConfiguration	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetDot1XConfigurationRequest	*Ce message contient:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
SetDot1XConfigurationResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:EAPMethodNotSupported	*La méthode EAP suggérée n'est pas prise en charge.*
env:Sender ter:OperationProhibited ter:CertificateID	*Erreur d'ID de certificat non valide.*
env:Sender ter:OperationProhibited ter:ReferenceToken	*Erreur Dot1XConfigurationToken non valide*
env:Sender ter:InvalidArgVal ter:InvalidDot1X	*Configuration IEEE 802.1X non valide.*

8.5.8.4 Obtention d'une configuration IEEE 802.1X

Cette opération permet d'obtenir un ensemble de paramètres de configuration IEEE 802.1X du dispositif en spécifiant le jeton de configuration (Dot1XConfigurationToken).

Le dispositif doit prendre en charge cette commande si la prise en charge de IEEE 802.1X est signalée via la fonctionnalité Security Dot1X comme décrit dans le Tableau 65.

Que la méthode 802.1X de la configuration extraite contienne ou pas un mot de passe, le dispositif ne doit pas inclure l'élément Password dans la réponse.

Tableau 65 – Commande GetDot1XConfiguration

GetDot1XConfiguration	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDot1XConfigurationRequest	*Ce message contient:* tt:ReferenceToken **Dot1XConfigurationToken**[1][1]
GetDot1XConfigurationResponse	*Ce message contient:* tt:Dot1XConfiguration **Dot1XConfiguration**[1][1]
Codes de défaut	**Description**
env:Sender ter:OperationProhibited ter:ReferenceToken	*Erreur Dot1XConfigurationToken non valide*

8.5.8.5 Obtention des configurations IEEE 802.1X

Cette opération permet d'obtenir tous les ensembles de paramètres de configuration IEEE 802.1X existants du dispositif. Le dispositif doit répondre avec toutes les configurations IEEE 802.1X afin que le client puisse savoir combien de configurations IEEE 802.1X existent et comment elles sont configurées.

Le dispositif doit prendre en charge cette commande si la prise en charge de IEEE 802.1X est signalée via la fonctionnalité Security Dot1X comme décrit dans le Tableau 66.

Que la méthode 802.1X de la configuration extraite contienne ou pas un mot de passe, le dispositif ne doit pas inclure l'élément Password dans la réponse.

Tableau 66 – Commande GetDot1XConfigurations

GetDot1XConfigurations	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetDot1XConfigurationsRequest	*Ceci est un message vide.*
GetDot1XConfigurationsResponse	*Ce message contient:* tt:Dot1XConfiguration **Dot1XConfiguration**[0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.8.6 Suppression d'une configuration IEEE 802.1X

Cette opération permet de supprimer un ensemble de paramètres de configuration IEEE 802.1X du dispositif comme décrit dans le Tableau 67. La configuration qu'il convient de supprimer est indiquée par "Dot1XConfigurationToken" dans la demande. Le dispositif doit prendre en charge cette commande si la prise en charge de IEEE 802.1X est signalée via la fonctionnalité Security Dot1X.

Tableau 67 – Commande DeleteDot1XConfigurations

DeleteDot1XConfigurations	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
DeleteDot1XConfigurationRequest	*Ce message contient:* tt:ReferenceToken **Dot1XConfigurationToken**[1][1]
DeleteDot1XConfigurationResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:OperationProhibited ter:ReferenceToken	*Erreur Dot1XConfigurationToken non valide*
env:Receiver ter:OperationProhibited ter:ReferenceToken	*La configuration IEEE 802.1X spécifiée ne peut pas être supprimée.*

8.5.9 Création d'un certificat autosigné

Cette opération génère une paire de clés privée/publique et peut également créer un certificat de dispositif autosigné par suite de la génération de la paire de clés comme décrit dans le Tableau 68. Le certificat est créé à l'aide d'un mécanisme de génération de paire de clés intégrées adapté.

Si le dispositif prend en charge la génération de paire de clés intégrées, celui qui prend en charge TLS doit prendre en charge cette commande de création de certificat. De même, si un dispositif prend en charge la génération de paire de clés intégrées, celui qui signale la prise en charge de IEEE 802.1X via la fonctionnalité Security Dot1X doit prendre en charge cette commande pour générer la paire de clés. Les certificats et les paires de clés sont identifiés par des ID de certificat. Ces ID sont choisis par le demandeur de génération de certificat ou par le dispositif (si aucune valeur d'ID n'est donnée).

Tableau 68 – Commande CreateCertificate

CreateCertificate	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
CreateCertificateRequest	*Ce message contient (le cas échéant) l'ID de certificat demandé et les paramètres supplémentaires demandés: subject, valid not before et valid not after.* xs:token **CertificateID** [0][1] xs:string **Subject** [0][1] xs:dateTime **ValidNotBefore** [0][1] xs:dateTime **ValidNotAfter** [0][1]
CreateCertificateResponse	*Ce message contient le certificat autosigné généré.* tt:Certificate **NvtCertificate** [1][1]
Codes de défaut	**Description**
env:Receiver ter:Action ter:KeyGeneration	*La génération de clé privée/publique n'a pas abouti.*
env:Sender ter:InvalidArgVal ter:CertificateID	*L'ID de certificat existe déjà.*
env:Sender ter:InvalidArgVal ter:InvalidDateTime	*Le paramètre ValidNotBefore ou ValidNotAfter spécifié n'est pas valide.*

8.5.10 Obtention de certificats

Cette opération permet d'obtenir tous les certificats de serveur du dispositif (y compris les certificats autosignés) pour l'authentification TLS, ainsi que tous les certificats client du dispositif pour l'authentification IEEE 802.1X. Cette commande répertorie uniquement les certificats de serveur TLS et les certificats client IEEE 802.1X du dispositif (pas les certificats CA, ni les certificats racines dignes de confiance). Les certificats sont retournés sous la forme de données binaires. Un dispositif qui prend en charge TLS doit prendre en charge cette commande comme décrit dans le Tableau 69 et les certificats doivent être codés en utilisant les règles de codage ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1].

Tableau 69 – Commande GetCertificates

GetCertificates	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetCertificatesRequest	*Ceci est un message vide.*
GetCertificatesResponse	*Ce message contient une liste de certificats de dispositif.* tt:Certificate **NvtCertificate** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.11 Obtention de certificats CA

Les certificats CA sont chargés dans un dispositif et utilisés dans les deux cas suivants. L'un est destiné à l'authentification de client TLS dans la fonction de serveur TLS. L'autre est destiné à l'authentification du serveur d'authentification dans la fonction IEEE 802.1X. Cette opération permet d'obtenir tous les certificats CA chargés dans un dispositif. Un dispositif qui prend en charge l'authentification client TLS ou IEEE 802.1X doit prendre en charge cette commande comme décrit dans le Tableau 70, et les certificats retournés doivent être codés en utilisant les règles de codage ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1].

Tableau 70 – Commande GetCACertificates

GetCACertificates	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetCACertificatesRequest	*Ceci est un message vide.*
GetCACertificatesResponse	*Ce message contient une liste de certificats CA.* tt:Certificate **CACertificate** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.12 Obtention de statut de certificat

Cette opération est spécifique à la fonctionnalité TLS. Cette opération permet d'obtenir le statut (activé/désactivé) des certificats de serveur TLS du dispositif. Un dispositif qui prend en charge TLS doit prendre en charge cette commande dans le Tableau 71.

Tableau 71 – Commande GetCertificatesStatus

GetCertificatesStatus	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetCertificatesStatusRequest	*Ceci est un message vide.*
GetCertificatesStatusResponse	*Ce message contient une liste de certificats de serveur de dispositif référencés par ID et leur statut. Le statut est défini comme étant une valeur booléenne (vrai = activé, faux = désactivé).* tt:CertificateStatus **CertificateStatus** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.13 Définition de statut de certificat

Cette opération est spécifique à la fonctionnalité TLS. Cette opération définit le statut (activé/désactivé) des certificats de serveur TLS du dispositif. Un dispositif qui prend en charge TLS doit prendre en charge cette commande comme décrit dans le Tableau 72. En règle générale, un seul certificat de serveur de dispositif peut être activé à la fois.

Tableau 72 – Commande SetCertificatesStatus

SetCertificatesStatus	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetCertificatesStatus-Request	*Ce message contient une liste de certificats de serveur de dispositif référencés par ID et le statut de certificat demandé, c'est-à-dire, activé ou désactivé.* tt:CertificateStatus **CertificateStatus** [0][non limité]
SetCertificatesStatus-Response	*Ceci est un message vide*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Référence de certificat inconnue.*

8.5.14 Obtention de demande de certificat

Cette opération permet de demander une signature de certificat PKCS #10 au dispositif. Le champ d'informations renvoyé doit être mis en forme comme indiqué dans [RFC 2986] ou au format codé PEM [RFC 2986]. Pour que cette commande fonctionne, le dispositif doit déjà disposer d'une paire de clés privée/publique. Il convient que cette paire de clés soit référencée par CertificateID comme indiqué dans la description de paramètre d'entrée. CertificateID fait référence à la paire de clés générée à l'aide de la commande CreateCertificate définie en 8.5.9.

Un dispositif qui prend en charge la génération de paire de clés intégrées prenant en charge TLS ou IEEE 802.1X à l'aide du certificat client, doit prendre en charge cette commande comme décrit dans le Tableau 73.

Tableau 73 – Commande GetPkcs10Request

GetPkcs10Request	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetPkcs10RequestRequest	*Ce message contient une référence au certificat (paire de clés) et des paramètres de certificat facultatifs pour la demande de certificat. Il est nécessaire que ces attributs soient codés en tant qu'objets DER ASN.1.* xs:token **CertificateID** [1][1] xs:string **Subject** [0][1] xs:BinaryData **Attributes** [0][1]
GetPkcs10RequestResponse	*Ce message contient la structure de données de demande PKCS#10.* tt:BinaryData **Pkcs10Request** [1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*CertificateID non valide*
env:Receiver ter:Action ter:Signature	*La création d'une signature PKCS#10 n'a pas abouti.*

8.5.15 Obtention de statut de certificat de client

Cette opération est spécifique à la fonctionnalité TLS. Cette opération permet d'obtenir le statut (activé/désactivé) d'authentification de client TLS du dispositif. Un dispositif qui prend en charge TLS doit prendre en charge cette commande comme décrit dans le Tableau 74.

Tableau 74 – Commande GetClientCertificateMode

GetClientCertificateMode	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetClientCertificateMode-Request	*Ceci est un message vide.*
GetClientCertificateMode-Response	*Ce message contient le statut d'authentification de client du dispositif, c'est-à-dire, activé ou désactivé.* xs:boolean **Enabled** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.5.16 Définition de statut de certificat de client

Cette opération est spécifique à la fonctionnalité TLS. Cette opération définit le statut (activé/désactivé) d'authentification de client TLS du dispositif. Un dispositif qui prend en charge TLS doit prendre en charge cette commande comme décrit dans le Tableau 75.

Tableau 75 – Commande SetClientCertificateMode

SetClientCertificateMode	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetClientCertificateMode-Request	*Ce message contient le statut d'authentification de client du dispositif demandé, c'est-à-dire, activé ou désactivé.* xs:boolean **Enabled** [1][1]
SetClientCertificateMode-Response	*Ceci est un message vide*
Codes de défaut	**Description**
env:Receiver ter:InvalidArgVal ter:ClientAuth	*Tentative d'activation d'authentification de client, mais l'authentification de client n'est pas prise en charge ou non configurée.*

8.5.17 Chargement de certificat de dispositif

Les certificats de serveur TLS ou les certificats de client IEEE 802.1X créés à l'aide de la commande de demande de certificat PKCS#10 peuvent être chargés dans le dispositif grâce à cette commande (voir 8.5.14). L'ID de certificat de la demande doit être présent. Le dispositif peut trier le ou les certificats reçus en fonction des informations de clé publique et de sujet qu'ils contiennent.

L'ID de certificat de la demande correspond à la valeur d'ID souhaitée par le client. Le dispositif est censé balayer les paires de clés générées présentes dans le dispositif afin d'identifier celle correspondant au certificat chargé, puis d'associer le certificat à la paire de clés.

Un dispositif qui prend en charge la génération de paire de clés intégrées prenant en charge TLS ou IEEE 802.1X doit prendre en charge cette commande comme décrit dans le Tableau 76.

Les certificats doivent être codés à l'aide des règles de codage ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1].

Cette commande s'applique à tous les types de dispositifs, même si le nom de paramètre est appelé NVTCertificate pour des raisons historiques.

Tableau 76 – Commande LoadCertificates

LoadCertificates	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
LoadCertificatesRequest	*Ce message contient une liste de certificats de dispositif à charger.* tt:Certificate **NVTCertificate** [1][non limité]
LoadCertificatesResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateFormat	*Format de certificat incorrect ou bien le format n'est pas pris en charge par le dispositif.*
env:Sender ter:InvalidArgVal ter:CertificateID	*L'ID de certificat existe déjà.*
env:Sender ter:InvalidArgVal ter:InvalidCertificate	*Certificat non valide.*

8.5.18 Chargement de certificat de dispositif avec sa clé privée

Dans certains cas, une autorité de certification ou un organisme équivalent crée un certificat sans disposer d'une demande de signature de certificat PKCS#10. Dans ces cas, le certificat est associé à sa clé privée. Cette commande est utilisée pour ce type de situation. La valeur d'ID souhaitée par le client est éventuellement attribuée à l'ID de certificat dans la demande. Si l'ID de certificat n'est pas précisé dans la demande, le dispositif peut choisir l'ID en conséquence.

Cette opération permet d'importer une paire de clés privée/publique dans le dispositif.

Les certificats doivent être codés à l'aide des règles de codage ASN.1 [ISO/IEC 8824-2], [ISO/IEC 8824-3], [ISO/IEC 8824-4] DER [ISO/IEC 8825-1].

Il convient qu'un dispositif qui ne prend pas en charge la génération de paire de clés intégrées et prend en charge TLS ou IEEE 802.1X à l'aide du certificat client, prenne en charge cette commande. Un dispositif qui prend en charge la génération de paire de clés intégrées peut prendre en charge cette commande comme décrit dans le Tableau 77. Il convient que la politique de sécurité d'un dispositif qui prend en charge cette opération s'assure que la clé privée est suffisamment protégée.

Tableau 77 – Commande LoadCertificateWithPrivateKey

LoadCertificateWithPrivateKey	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
LoadCertificateWithPrivateKeyRequest	*Ce message contient une paire de clés privée/publique à importer.* tt:CertificateWithPrivateKey **CertificateWithPrivateKey**[1][non limité]
LoadCertificateWithPrivateKeyResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env: Sender ter:InvalidArgVal ter:CertificateFormat	*Format de certificat incorrect ou bien le format n'est pas pris en charge par le dispositif.*
env:Sender ter:InvalidArgVal ter:CertificateID	*L'ID de certificat existe déjà.*
env: Sender ter:InvalidArgVal ter: KeysNotMatching	La clé publique et la clé privée ne correspondent pas.

8.5.19 Obtention de demande d'informations de certificat

Cette opération permet de demander les informations de certificat spécifiées par l'ID de certificat. Il convient que le dispositif réponde par les données "Issuer DN", "Subject DN", "Key usage", "Extended key usage", "Key Length", "Version", "Serial Number", "Signature Algorithm" et "Validity" comme informations du certificat, tant que le dispositif peut extraire ces informations du certificat spécifié. IssuerDN et SubjectDN doivent être codés à l'aide des règles de la norme [RFC 4514].

Il convient qu'un dispositif qui prend en charge TLS ou IEEE 802.1X prenne en charge cette commande comme décrit dans le Tableau 78.

Tableau 78 – Commande GetCertificateInformation

GetCertificateInformation	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetCertificateInformationRequest	*Ce message contient:* *CertificateID: Jeton du certificat.* xs: *token CertificateID* [1][1]
GetCertificateInformationResponse	*Ce message contient:* tt:CertificateInformation **CertificateInformation**[1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*ID de certificat non valide*

8.5.20 Chargement de certificats CA

Cette commande est utilisée lorsqu'il est nécessaire de charger des certificats CA ou racines dignes de confiance afin de vérifier leur homologue, c'est-à-dire, vérification du certificat de client dans la fonction TLS ou du certificat de serveur dans la fonction IEEE 802.1X.

Un dispositif qui prend en charge TLS ou IEEE 802.1X doit prendre en charge cette commande comme décrit dans le Tableau 79. Le dispositif doit prendre en charge le format DER. Les autres formats peuvent être pris en charge par le dispositif. Le dispositif peut trier le ou les certificats reçus en fonction des informations de clé publique et de sujet qu'ils contiennent. Soit le chargement de tous les certificats CA aboutit, soit un message de défaut doit être retourné sans charger de certificat CA.

Tableau 79 – Commande LoadCACertificates

LoadCACertificates	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
LoadCACertificatesRequest	*Ce message contient une liste de certificats CA de dispositif à charger.* tt:Certificate **CACertificate** [1][non limité]
LoadCACertificatesResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateFormat	*Format de certificat incorrect ou bien le format n'est pas pris en charge par le dispositif.*
env:Sender ter:InvalidArgVal ter:CACertificateID	*L'ID de certificat CA existe déjà.*
env:Receiver ter:OperationProhibited ter:MaxCertificates	*Nombre maximal de certificats déjà chargés.*

8.5.21 Suppression de certificat

Cette opération permet de supprimer un ou plusieurs certificats. Le dispositif peut également supprimer la paire de clés privée/publique associée au certificat à supprimer. Le dispositif qui

prend en charge TLS ou IEEE 802.1X doit prendre en charge la suppression d'un ou de plusieurs certificats grâce à cette commande comme décrit dans le Tableau 80. Soit la suppression de tous les certificats aboutit, soit un message de défaut doit être retourné sans supprimer de certificat.

Tableau 80 – Commande DeleteCertificates

DeleteCertificates	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
DeleteCertificatesRequest	*Ce message supprime des certificats identifiés avec le paramètre CertificateID.* xs:token **CertificateID**[1][non limité]
DeleteCertificatesResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:CertificateID	*Référence de certificat inconnue.*
env:Receiver ter:OperationProhibited ter:CertificateID	*Les certificats spécifiés ne peuvent pas être supprimés.*

8.5.22 Obtention de l'utilisateur distant

Cette opération renvoie l'utilisateur distant configuré (le cas échéant) comme décrit dans le Tableau 81. Un dispositif qui signale la prise en charge du traitement des utilisateurs distants via la fonctionnalité de sécurité RemoteUserHandling doit prendre en charge cette opération. L'utilisateur est uniquement valide en tant qu'utilisateur HTTP/RTSP.

UseDerivedPassword est inclus pour assurer la compatibilité en amont et il convient qu'il soit toujours défini sur "False" (faux).

Tableau 81 – Commande GetRemoteUser

GetRemoteUser	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetRemoteUserRequest	*Ceci est un message vide.*
GetRemoteUserResponse	*Ce message contient l'utilisateur distant configuré (le cas échéant). Les valeurs retournées sont:* • *xs:string **Username** [1][1]* • *xs:boolean **UseDerivedPassword** [1][1]* *Un dispositif ne doit jamais retourner le champ Password de **RemoteUser**.* tt:RemoteUser: **RemoteUser** [0][1]
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:NotRemoteUser	*Le traitement des utilisateurs distants n'est pas pris en charge*

8.5.23 Définition de l'utilisateur distant

Cette opération permet de définir l'utilisateur distant comme décrit dans le Tableau 82. Un dispositif qui signale la prise en charge du traitement des utilisateurs distants via la fonctionnalité de sécurité RemoteUserHandling doit prendre en charge cette opération. L'utilisateur est uniquement valide en tant qu'utilisateur HTTP/RTSP.

Pour supprimer l'utilisateur distant, il convient d'appeler SetRemoteUser sans le paramètre RemoteUser.

UseDerivedPassword est inclus pour assurer la compatibilité en amont et il convient qu'il soit toujours défini sur "false" (faux).

Tableau 82 – Commande SetRemoteUser

SetRemoteUser	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetRemoteUserRequest	*Ce message contient l'utilisateur distant. Les valeurs possibles sont:* • *xs:string* ***Username*** *[1][1]* • *xs:string* ***Password*** *[0][1]* • *xs:boolean* ***UseDerivedPassword*** *[1][1]* tt:RemoteUser: **RemoteUser** [0][1]
SetRemoteUserResponse	*Ceci est un message vide*
Codes de défaut	**Description**
env:Receiver ter:ActionNotSupported ter:NotRemoteUser	*Traitement des utilisateurs distants non pris en charge*

8.5.24 Obtention de la référence de point terminal

Un client peut demander la propriété d'adresse de référence de point terminal de service du dispositif qui peut permettre de déduire le mot de passe équivalent pour l'opération d'utilisateur distant. Il convient que le dispositif prenne en charge la commande GetEndpointReference comme décrit dans le Tableau 83 retournant la propriété d'adresse de la référence de point terminal de service du dispositif.

Tableau 83 – Commande GetEndpointReference

GetEndpointReference	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetEndpointReferenceRequest	*Ceci est un message vide.*
GetEndpointReference-Response	*L'URL demandée.* xs:string **GUID** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

8.6 Opérations auxiliaires

Le Paragraphe 8.6 décrit les opérations de gestion des commandes auxiliaires prises en charge par un dispositif comme décrit dans le Tableau 84, telles que le contrôle d'une lampe à infrarouge, d'un chauffage, d'un dispositif d'essuyage ou d'un thermomètre connecté au dispositif.

Les commandes prises en charge par le dispositif sont reportées dans l'attribut Misc – AuxiliaryCommands retourné par les commandes GetServiceCapabilities, voir 8.2.2.3. Il convient que la commande transmise à l'aide de cette commande corresponde à l'une des commandes prises en charge par le dispositif. Par exemple, si la réponse de la commande GetServiceCapabilities répertorie uniquement la commande irlampon, l'argument SendAuxiliaryCommand est alors irlampon, ce qui peut indiquer l'allumage de la lampe à infrarouge connectée.

Même si le nom des commandes auxiliaires peut être défini librement, les commandes commençant par le préfixe tt: servent exclusivement à définir les commandes fréquemment utilisées et doivent toutes partager la même syntaxe "tt:command|parameter".

- tt:Wiper|On – Demande le démarrage du dispositif d'essuyage.
- tt:Wiper|Off – Demande l'arrêt du dispositif d'essuyage.
- tt:Washer|On – Demande le démarrage du dispositif de lavage.
- tt:Washer|Off – Demande l'arrêt du dispositif de lavage.

- tt:WashingProcedure|On – Demande le démarrage de la procédure de lavage.
- tt:WashingProcedure|Off – Demande l'arrêt de la procédure de lavage.

Un dispositif qui indique une fonctionnalité de service auxiliaire doit prendre en charge cette commande.

Tableau 84 – Commande SendAuxiliary

SendAuxiliaryCommand	Classe d'accès: ACTUATE
Nom de message	**Description**
SendAuxiliaryCommandRequest	*Ce message contient la commande auxiliaire.* tt:AuxiliaryData **AuxiliaryCommand**[1][1]
SendAuxiliaryCommandResponse	*La réponse contient la réponse auxiliaire.* tt:AuxiliaryData **AuxiliaryCommandResponse**[0][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:AuxiliaryDataNotSupported	*Le AuxiliaryCommand demandé n'est pas pris en charge.*

8.7 Événements de surveillance

8.7.1 Utilisation du processeur

Si un dispositif prend en charge la surveillance de l'utilisation de l'unité de traitement, il convient qu'il fournisse l'événement de surveillance de l'utilisation de l'unité de traitement afin d'indiquer au client l'utilisation réelle de son unité de traitement en pourcentage:

```
Topic: tns1:Monitoring/ProcessorUsage
<tt:MessageDescription IsProperty="true">
  <tt:Source>
  <tt:SimpleItemDescription Name=”Token” Type=”tt:ReferenceToken”/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="Value" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.2 Statut de liaison

Si un dispositif prend en charge la surveillance du statut de liaison, il convient qu'il fournisse l'événement de surveillance du statut de liaison afin d'indiquer au client le statut réel de sa liaison.

```
Topic: tns1:Monitoring/LinkStatus
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Link"
Type="tt:NetworkInterfaceConnectionSetting"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.3 Statut de chargement

Si un dispositif prend en charge la surveillance de son micrologiciel de chargement (chargement des informations du micrologiciel ou du système), il convient qu'il fournisse le statut en pourcentage d'une mise à jour en cours de réalisation à l'aide de l'événement de statut de chargement.

```
Topic: tns1:Monitoring/UploadStatus
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.4 Temps de fonctionnement

L'ensemble d'événements défini dans le présent paragraphe porte sur le temps de fonctionnement. Il convient qu'un dispositif prenant en charge les événements liés au temps de fonctionnement fournisse les événements suivants.

Il convient que l'événement suivant soit généré avec une valeur "true" (vrai) lorsque la limite de temps de fonctionnement est atteinte.

```
Topic: tns1:Monitoring/OperatingTime/DefinedLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Il convient que l'événement suivant soit généré avec une valeur "true" (vrai) lorsque la limite par défaut de la moyenne des temps de bon fonctionnement (MTBF) des dispositifs a été atteinte.

```
Topic:
tns1:Monitoring/OperatingTime/MeanTimeBetweenFailuresDefaultLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Il convient que l'événement suivant soit généré avec une valeur "true" (vrai) lorsque la limite de fonctionnement de la moyenne des temps de bon fonctionnement (MTBF) des dispositifs a été atteinte.

```
Topic:
tns1:Monitoring/OperatingTime/MeanTimeBetweenFailuresOperationLimitReached
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique à quelle date le dispositif a été réinitialisé à ses paramètres d'usine pour la dernière fois.

```
Topic: tns1:Monitoring/OperatingTime/LastReset
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique à quelle date le dispositif a été redémarré pour la dernière fois.

```
Topic: tns1:Monitoring/OperatingTime/LastReboot
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique à quelle date l'horloge du dispositif a été synchronisée pour la dernière fois, soit via un message NTP, soit via un appel SetSystemDateAndTime.

```
Topic: tns1:Monitoring/OperatingTime/LastClockSynchronization
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique la dernière activité de maintenance sur le dispositif.

```
Topic: tns1:Monitoring/Maintenance/Last
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique la prochaine activité de maintenance sur le dispositif.

```
Topic: tns1:Monitoring/Maintenance/NextScheduled
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique à quelle date la dernière sauvegarde de la configuration de dispositif a été extraite.

```
Topic: tns1:Monitoring/Backup/Last
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:dateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

Il convient que l'événement suivant soit généré avec une valeur "true" (vrai) lorsque la zone d'exploitation pour laquelle le dispositif est certifié n'est pas respectée en raison d'influences extérieures.

```
Topic: tns1:Monitoring/AreaOfOperation/OutsideCertifiedArea
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Il convient que l'événement suivant soit généré avec une valeur "true" (vrai) lorsque la zone d'exploitation pour laquelle le dispositif est configuré n'est pas respectée en raison d'influences extérieures.

```
Topic: tns1:Monitoring/AreaOfOperation/OutsideConfiguredArea
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.5 Conditions d'environnement

Si les mesurages des conditions d'environnement sont pris en charge, il convient qu'un dispositif fournisse les événements suivants.

L'événement suivant indique l'humidité relative en pourcentage.

```
Topic: tns1:Monitoring/EnvironmentalConditions/RelativeHumidity
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

L'événement suivant indique la température de fonctionnement du dispositif en degré Celsius.

```
Topic: tns1:Monitoring/EnvironmentalConditions/Temperature
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Status" Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.6 Capacité de la batterie

Si les mesurages du niveau de batterie sont pris en charge, il convient qu'un dispositif fournisse les données grâce à l'événement BatteryCapacity.

```
Topic: tns1:Monitoring/BatteryCapacity
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="PercentageRemainingCapacity"
                              Type="xs:float"/>
  </tt:Data>
</tt:MessageDescription>
```

8.7.7 Gestion de dispositif

Les rubriques suivantes signalent des informations importantes sur le statut du dispositif:

```
tns1:Device/OperationMode/ShutdownInitiated
tns1:Device/OperationMode/UploadInitiated
tns1:Device/HardwareFailure/FanFailure
tns1:Device/HardwareFailure/PowerSupplyFailure
tns1:Device/HardwareFailure/StorageFailure
tns1:Device/HardwareFailure/TemperatureCritical
```

8.8 Codes de défaut spécifiques au service

Le Tableau 85 répertorie les codes de défaut spécifiques au service de dispositif. De plus, chaque commande peut également générer un défaut générique, voir Tableau 5.

Les défauts spécifiques sont définis sous la forme d'un sous-code de défaut générique, voir 5.12.3.2. Le sous-code générique parent est le sous-code en haut de chaque ligne ci-dessous, le sous-code de défaut spécifique se trouvant en bas de la cellule.

Tableau 85 – Codes de défaut spécifiques au service de dispositif

Code de défaut	Sous-code parent Sous-code	Raison de défaut	Description
env:Receiver	ter:Action ter:EmptyPolicy	La politique est vide	Le fichier de politique du dispositif n'existe pas ou est vide.
env:Receiver	ter:Action ter:EmptyScope	La liste de domaines d'application est vide.	La liste de domaines d'application est vide.
env:Receiver	ter:Action ter:FirmwareUpgradeFailed	La mise à niveau a échoué	La mise à niveau de micrologiciel a échoué.
env:Receiver	ter:Action ter:KeyGeneration	La génération de clé a échoué	La génération de clé privée/publique n'a pas abouti.
env:Receiver	ter:Action ter:Signature	La création d'une signature a échoué	La création d'une signature PKCS#10 n'a pas abouti.
env:Receiver	ter:InvalidArgVal ter:ClientAuth	Authentification de client non prise en charge	Tentative d'activation d'authentification de client, mais l'authentification de client n'est pas prise en charge ou n'est pas configurée
env:Receiver	ter:Action ter:TooManyUsers	Trop grand nombre d'utilisateurs	Nombre maximal d'utilisateurs pris en charge dépassé.
env:Receiver	ter:Action ter:TooManyScopes	Liste de domaines d'application trop grande	La liste de domaines d'application dépasse le nombre de domaines d'application pris en charge.
env:Receiver	ter:ActionNotSupported ter:NoSuchService	Le service n'est pas pris en charge	La catégorie de service WSDL demandée n'est pas prise en charge par le dispositif.
env:Sender	ter: InvalidArgs ter:AccesslogUnavailable	Aucun journal d'accès disponible	Aucune information de journal d'accès n'est disponible.
env:Sender	ter:InvalidArgVal ter:CertificateFormat	Format non valide	Format de certificat incorrect ou bien le format n'est pas pris en charge par le dispositif.
env:Sender	ter:InvalidArgVal ter:CertificateID	ID de certificat non valide	Référence de certificat inconnue ou bien l'ID de certificat existe déjà.
env:Sender	ter:InvalidArgVal ter:CACertificateID	ID de certificat CA non valide	Référence de certificat CA inconnue ou bien l'ID de certificat CA existe déjà.
env:Sender	ter:InvalidArgVal ter:InvalidBackupFile	Fichier non valide	Le ou les fichiers de sauvegarde ne sont pas valides.
env:Sender	ter:InvalidArgVal ter:InvalidDateTime	Date et heure non valides.	Une date ou une heure non valide a été spécifiée.
env:Sender	ter:InvalidArgVal ter:NtpServerUndefined	Serveur NTP non défini.	Impossible de remplacer DateTimeType par NTP, car aucun serveur NTP n'est défini.
env:Sender	ter:InvalidArgVal ter:InvalidDnsName	Nom non valide	Le nom de serveur NTP suggéré est non valide.
env:Sender	ter:InvalidArgVal ter:TimeSyncedToNtp	Horloge synchronisée via NTP	Le DateTimeType actuel exige un serveur NTP.
env:Sender	ter: InvalidArgs ter:InvalidFirmware	Micrologiciel non valide	Le micrologiciel est non valide, c'est-à-dire qu'il n'est pas pris en charge par ce dispositif.

Code de défaut	Sous-code parent Sous-code	Raison de défaut	Description
env:Sender	ter:InvalidArgVal ter:InvalidGatewayAddress	Adresse non valide	L'adresse de passerelle fournie était non valide.
env:Sender	ter:InvalidArgVal ter:InvalidHostname	Nom non valide	Le dispositif ne peut pas accepter le nom d'hôte demandé.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceSpeed	Vitesse non valide	La vitesse suggérée n'est pas prise en charge.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceType	Type non valide	Le type d'interface réseau suggéré n'est pas pris en charge.
env:Sender	ter:InvalidArgVal ter:InvalidIPv4Address	Adresse non valide	L'adresse IPv4 suggérée est non valide.
env:Sender	ter:InvalidArgVal ter:NoIPv4Address	L'adresse n'existe pas	L'adresse IPv4 à supprimer n'existe pas.
env:Sender	ter:InvalidArgVal ter:InvalidIPv6Address	Adresse non valide	L'adresse IPv6 suggérée est non valide.
env:Sender	ter:InvalidArgVal ter:NoIPv6Address	L'adresse n'existe pas	L'adresse IPv6 à supprimer n'existe pas.
env:Sender	ter:InvalidArgVal ter:InvalidMtuValue	Données non valides	La valeur MTU est non valide.
env:Sender	ter:InvalidArgVal ter:InvalidNetworkInterface	Jeton non valide	Le jeton d'interface réseau fourni n'existe pas.
env:Sender	ter:InvalidArgVal ter:InvalidTimeZone	Données non valides	Un fuseau horaire non valide a été spécifié.
env:Sender	ter:InvalidArgVal ter:IPFilterListIsFull	La liste est pleine	Aucun filtre IP supplémentaire ne peut être ajouté, car la liste de filtres IP est pleine.
env:Sender	ter:InvalidArgs ter:PolicyFormat	Format non valide	La politique demandée ne peut pas être définie en raison d'un format de politique inconnu.
env:Sender	ter:InvalidArgVal ter:ServiceNotSupported	Le service n'est pas pris en charge	Le service réseau fourni n'est pas pris en charge.
env:Sender	ter:InvalidArgVal ter:SupportInformationUnavailable	Aucune information de prise en charge disponible	Aucune information de prise en charge n'est disponible.
env:Sender	ter: InvalidArgs ter:SystemlogUnavailable	Aucun journal système disponible	Aucune information de journal système n'est disponible.
env:Sender	ter:InvalidArgVal ter:UsernameMissing	Nom d'utilisateur non reconnu	Nom d'utilisateur non reconnu.
env:Sender	ter:OperationProhibited ter:FixedScope	Tentative de suppression de paramètre de domaine d'application fixe	Tentative de suppression de paramètre de domaine d'application fixe. La commande est rejetée.
env:Sender	ter:InvalidArgVal ter:NoScope	Le domaine d'application n'existe pas	Tentative de suppression d'un domaine d'application qui n'existe pas.
env:Sender	ter:OperationProhibited ter:Password	Mot de passe trop faible	Mot de passe trop faible.
env:Sender	ter:OperationProhibited	Mot de passe trop long	Le mot de passe est trop long.

Code de défaut	Sous-code parent Sous-code	Raison de défaut	Description
	ter:PasswordTooLong		
env:Sender	ter:OperationProhibited ter:UsernameTooShort	Mot de passe trop court	Le mot de passe est trop court.
env:Sender	ter:OperationProhibited ter:ScopeOverwrite	Tentative d'écrasement de réglage de domaine d'application de dispositif permanent	Un paramètre de domaine d'application remplace un réglage de domaine d'application de dispositif permanent, commande rejetée.
env:Sender	ter:OperationProhibited ter:UsernameClash	Le nom d'utilisateur existe déjà	Le nom d'utilisateur existe déjà.
env:Sender	ter:OperationProhibited ter:UsernameTooLong	Nom d'utilisateur trop long	Le nom d'utilisateur est trop long.
env:Receiver	ter:ActionNotSupported ter:InvalidDot11	Non pris en charge	La configuration IEEE 802.11 n'est pas prise en charge.
env:Sender	ter:InvalidArgVal ter:InvalidSecurityMode	Non pris en charge	Le mode de sécurité sélectionné n'est pas pris en charge.
env:Sender	ter:InvalidArgVal ter:InvalidStationMode	Non pris en charge	Le mode de station sélectionné n'est pas pris en charge.
env:Sender	ter:InvalidArgVal ter:MissingDot11	Valeur IEEE 802.11 absente	La configuration de sécurité ne contient pas de valeur IEEE 802.11.
env:Sender	ter:InvalidArgVal ter:MissingPSK	Valeur PSK absente	La configuration de sécurité ne contient pas de valeur PSK.
env:Sender	ter:InvalidArgVal ter:MissingDot1X	La valeur IEEE 802.1X est absente	La valeur IEEE 802.1X de la configuration de sécurité est absente ou inexistante.
env:Sender	ter:InvalidArgVal ter:IncompatibleDot1X	La valeur IEEE 802.1X est incompatible	La valeur IEEE 802.1X de la configuration de sécurité n'est pas compatible avec l'interface réseau.
env:Sender	ter:InvalidArgVal ter:NotDot11	Pas IEEE 802.11	L'interface n'est pas une interface IEEE 802.11.
env:Sender	ter:InvalidArgVal ter:InvalidDot1X	Configuration IEEE 802.1X non valide	La configuration IEEE 802.1X spécifiée n'est pas valide.
env:Receiver	ter:Action ter:NotConnectedDot11	IEEE 802.11 non connecté	Le réseau IEEE 802.11 n'est pas connecté.
env:Receiver	ter:ActionNotSupported ter:NotScanAvailable	ScanAvailableIEEE802.11Networks n'est pas pris en charge.	ScanAvailableIEEE802.11Networks n'est pas pris en charge.
env:Receiver	ter:ActionNotSupported ter:NotRemoteUser	Le traitement des utilisateurs distants n'est pas pris en charge.	Le traitement des utilisateurs distants n'est pas pris en charge.
env:Receiver	ter:ActionNotSupported ter:EAPMethodNotSupported	La méthode EAP suggérée n'est pas prise en charge.	La méthode EAP suggérée n'est pas prise en charge.
env:Receiver	ter:Action ter:MaxDot1X	Nombre maximal de configurations IEEE 802.1X atteint.	Le dispositif a atteint le nombre maximal de configurations IEEE 802.1X.
env:Receiver	ter:OperationProhibited ter:ReferenceToken	La configuration IEEE 802.1X spécifiée ne peut pas être supprimée.	La configuration IEEE 802.1X spécifiée ne peut pas être supprimée.
env:Receiver	ter:OperationProhibited	Le(s) certificat(s) spécifié(s) ne	Le(s) certificat(s) spécifié(s) ne

Code de défaut	Sous-code parent Sous-code	Raison de défaut	Description
	ter:CertificateID	peut(vent) pas être supprimé(s).	peut(vent) pas être supprimé(s).
env:Sender	ter:OperationProhibited ter:ReferenceToken	Erreur Dot1XConfigurationToken non valide.	Le jeton de configuration IEEE 802.1X spécifié n'est pas valide.
env:Sender	ter:OperationProhibited ter:CertificateID	Erreur d'ID de certificat non valide.	L'ID de certificat spécifié n'est pas valide.
env:Sender	ter:InvalidArgVal ter:ReferenceToken	Dot1XConfigurationToken existe déjà.	Le Dot1XConfigurationToken spécifié existe déjà dans le dispositif.
env:Sender	ter:InvalidArgVal ter:InvalidCertificate	Certificat non valide.	Le certificat spécifié n'est pas valide.
env:Receiver	ter:OperationProhibited ter:MaxCertificates	Nombre maximal de certificats déjà chargés.	Le dispositif a atteint le nombre maximal de certificats chargés.
env:Sender	ter:OperationProhibited ter:PasswordTooWeak	Mot de passe trop faible	Mot de passe trop faible
env:Sender	ter:InvalidArgVal ter:AuxiliaryDataNotSupported	Le AuxiliaryCommand demandé n'est pas pris en charge.	Le AuxiliaryCommand demandé n'est pas pris en charge.
env:Sender	ter:InvalidArgVal ter:InvalidTimeOutValue	Valeur de délai d'expiration spécifiée non valide.	La valeur TimeOut spécifiée n'est pas valide.
env:Receiver	ter:OperationProhibited ter:InvalidMode	Le dispositif n'est pas prêt à fonctionner en mode de commande.	Le dispositif n'est pas prêt à fonctionner en mode de commande.
env:Sender	ter:InvalidArgVal ter:FixedUser	Suppression d'utilisateur fixe	Le client tente de supprimer un utilisateur fixe.
env:Sender	ter:OperationProhibited ter:AnonymousNotAllowed	Le niveau d'utilisateur anonyme n'est pas autorisé.	Le niveau d'utilisateur anonyme n'est pas autorisé.
env:Sender	ter:InvalidArgVal ter:FixedUser	Le nom d'utilisateur ou le niveau d'utilisateur ne peut pas être modifié.	Le nom d'utilisateur ou le niveau d'utilisateur de l'utilisateur ou des utilisateurs spécifiés ne peut pas être modifié.
env:Sender	ter:InvalidArgVal ter:KeysNotMatching	Les clés ne correspondent pas	La clé publique et la clé privée ne correspondent pas.
env:Sender	ter:InvalidArgVal ter:PortAlreadyInUse	Port utilisé	Le port sélectionné est déjà utilisé.
env:Receiver	ter:ActionNotSupported ter:EnablingTlsFailed	L'activation de la TLS a échoué	Le dispositif ne prend pas en charge la TLS ou la TLS n'est pas correctement configurée.
env:Receiver	ter:ActionNotSupported ter:InvalidDHCPv6	L'activation du DHCPv6 a échoué	Le mode DHCPv6 avec état demandé n'est pas pris en charge.

9 Dispositif ES

9.1 Généralités

Ce service propose des commandes permettant d'extraire et de configurer les entrées et sorties physiques d'un dispositif.

Les commandes permettant de demander les entrées et sorties disponibles sont définies avec les commandes permettant de demander les relais disponibles. Ce service offre également des fonctions de demande et de modification de la configuration de ces entités.

Un dispositif disposant d'entrées et de sorties physiques doit prendre en charge ce service comme décrit à l'Article B.2.

9.2 Sorties de relais

9.2.1 Présentation

Les commandes d'entrée/sortie (E/S) sont utilisées pour contrôler l'état ou observer le statut des ports E/S. Si le dispositif comporte des ports E/S, il doit prendre en charge les commandes E/S.

9.2.2 Obtention de sorties de relais

Cette opération permet d'obtenir une liste de l'ensemble des sorties de relais disponibles et leurs paramètres comme décrit dans le Tableau 86.

Tableau 86 – Commande GetRelayOutputs

GetRelayOutputs	Classe d'accès: READ_MEDIA
Nom de message	**Description**
GetRelayOutputsRequest	*Ceci est un message vide.*
GetRelayOutputsResponse	*Ce message contient un ensemble de sorties de relais.* tt:RelayOutput **RelayOutputs** [0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

9.2.3 Obtention d'options de sortie de relais

Demande les paramètres et les plages disponibles pour une seule ou l'ensemble des sorties de relais comme décrit dans le Tableau 87. La méthode doit renvoyer les informations pour une seule sortie lorsqu'un RelayOutputToken est fourni sous la forme d'un paramètre de demande. Autrement, la méthode doit renvoyer les informations pour l'ensemble des sorties de relais.

Deux exemples:

1) Le dispositif prend en charge PT1S jusqu'à PT120S:

```
<tmd:RelayOutputOptions token='44'>
  <tmd:Mode>Monostable</tmd:Mode>
  <tmd:DelayTimes>1 120</tmd:DelayTimes>
</tmd:RelayOutputOptions>
```

2) Le dispositif prend en charge les valeurs PT0.5S, PT1S, PT2s et PT1M:

```
<tmd:RelayOutputOptions token='123'>
  <tmd:Mode>Monostable</tmd:Mode>
  <tmd:DelayTimes Discrete='True'>0.5 1 2 60</tmd:DelayTimes>
</tmd:RelayOutputOptions>
```

Tableau 87 – Commande GetRelayOutputOptions

GetRelayOutputOptions	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetRelayOutputOptionsRequest	*Ce message contient:* • *"RelayOutputToken": Référence de jeton facultative vers la sortie de relais demandée* tt:ReferenceToken **RelayOutputToken** [0][1]
GetRelayOutputOptionsResponse	*Ce message contient un ensemble d'options de sortie de relais.* tmd:RelayOutputOptions **RelayOutputOptions**[0][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

9.2.4 Définition de paramètres de sortie de relais

Cette opération définit les paramètres d'une sortie de relais comme décrit dans le Tableau 88.

Le relais peut fonctionner dans deux modes de relais:

- Bistable – Après la définition de l'état, le relais reste dans cet état.
- Monostable – Après la définition de l'état, le relais retourne à son état vacant après le temps spécifié.

L'état vacant physique d'une sortie de relais peut être configuré en attribuant la valeur "open" (ouvert) ou "closed" (fermé) à IdleState (inversion du comportement de relais).

L'état vacant "open" signifie que le relais est ouvert lorsque l'état du relais est "inactive" (inactif) par l'intermédiaire de la commande de déclenchement (voir 9.2.5) et fermé lorsque l'état est "active" (actif) à l'aide de la même commande.

L'état vacant "closed" signifie que le relais est fermé lorsque l'état du relais est "inactive" par l'intermédiaire de la commande de déclenchement (voir 9.2.5) et ouvert lorsque l'état est "active" à l'aide de la même commande.

Le paramètre de durée du champ de propriétés "DelayTime" décrit la période après laquelle le relais retourne à son état vacant s'il est en mode monostable. Si le relais est défini en mode bistable, la valeur du paramètre doit être ignorée.

Tableau 88 – Commande SetRelayOutputSettings.

SetRelayOutputSettings	Classe d'accès: ACTUATE
Nom de message	**Description**
SetRelayOutputSettingsRequest	*Ce message contient:* • *"RelayOutputToken": Référence de jeton vers la sortie de relais demandée.* • *"RelayOutputSettings": Paramètres du relais* tt:ReferenceToken **RelayOutputToken**[1][1] tt:RelayOutputSettings **RelayOutputSettings**[1][1]
SetRelayOutputSettingsResponse	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:RelayToken	*Référence de jeton de relais inconnue.*
Env:Sender ter:InvalidArgVal ter:ModeError	*Délai monostable non valide*

9.2.5 Déclenchement de sortie de relais

Cette opération déclenche une sortie de relais comme décrit dans le Tableau 89.

NOTE Il n'existe pas de commande GetRelayState. L'état logique réel de la sortie de relais est transmis par notification et leurs propriétés.

Tableau 89 – Commande SetRelayOutputState

SetRelayOutputState		Classe d'accès: ACTUATE
Nom de message	**Description**	
SetRelayOutputStateRequest	*Ce message contient:* • *RelayOutputToken": Référence de jeton vers la sortie de relais demandée.* • *"LogicalState": Demande de déclenchement, c'est-à-dire, actif ou inactif.* tt:ReferenceToken **RelayOutputToken** [1][1] tt:RelayLogicalState **LogicalState** [1][1]	
SetRelayOutputStateResponse	*Ceci est un message vide.*	
Codes de défaut	**Description**	
env:Sender ter:InvalidArgVal ter:RelayToken	*Référence de jeton de relais inconnue.*	

9.3 Entrées numériques

9.3.1 Présentation

Le type DigitalInput représente les entrées numériques physiques intégrées d'un dispositif qui surveillent l'état entités externes pouvant passer d'un circuit ouvert à un circuit fermé. Il s'agit par exemple des interrupteurs de sonnettes, des détecteurs de mouvement et des interrupteurs permettant de surveiller la position d'une porte.

9.3.2 GetDigitalInputs

Cette commande répertorie l'ensemble des entrées numériques disponibles d'un dispositif. Un dispositif signalant la prise en charge des entrées numériques par l'intermédiaire de ses fonctionnalités doit prendre en charge l'énumération des entrées disponibles grâce à la commande GetDigitalInputs comme décrit dans le Tableau 90.

Tableau 90 – Commande GetDigitalInputs

GetDigitalInputs		Classe d'accès: READ_MEDIA
Nom de message	**Description**	
GetDigitalInputsRequest	*Ceci est un message vide.*	
GetDigitalInputsResponse	*Contient une liste de structures décrivant l'ensemble des entrées numériques disponibles du dispositif. Si un dispositif ne possède aucune entrée numérique, une liste vide est renvoyée.* tt:DigitalInput **DigitalInputs** [0][non limité]	
Codes de défaut	**Description**	
Pas de codes de défaut spécifiques.		

9.4 SerialPorts

9.4.1 Présentation

Le type SerialPort représente le port série physique du dispositif et permet la lecture et l'écriture de données en série.

9.4.2 GetSerialPorts

Cette commande répertorie l'ensemble des ports série disponibles d'un dispositif. Un dispositif possédant un ou plusieurs ports série physiques doit prendre en charge l'énumération des ports série disponibles grâce à la commande GetSerialPorts comme décrit dans le Tableau 91.

Tableau 91 – Commande GetSerialPorts

GetSerialPorts	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetSerialPortsRequest	*Ceci est un message vide.*
GetSerialPortsResponse	*Contient une liste de structures décrivant l'ensemble des ports série disponibles du dispositif. Si un dispositif ne possède aucun port série, une liste vide est renvoyée* tmd:SerialPort **SerialPort**[0][non limité]
Codes de défaut	**Description**
Pas de codes de défaut spécifiques.	

9.4.3 GetSerialPortConfiguration

Cette opération permet d'obtenir une liste de l'ensemble des ports série disponibles avec leurs paramètres comme décrit dans le Tableau 92.

Tableau 92 – Commande GetSerialPortConfiguration

GetSerialPortConfiguration	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetSerialPortConfiguration-Request	*Ce message contient le jeton du port série.* tt:ReferenceToken **SerialPortToken**[1][1]
GetSerialPortConfiguration-Response	*Ce message contient un ensemble de SerialPortConfiguration.* Tmd:SerialPortConfiguration **SerialPortConfiguration**[1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*Le jeton de port série fourni n'existe pas.*

9.4.4 SetSerialPortConfiguration

Cette opération définit les paramètres d'un port série comme décrit dans le Tableau 93.

Tableau 93 – Commande SetSerialPortConfiguration

SetSerialPortConfiguration	Classe d'accès: WRITE_SYSTEM
Nom de message	**Description**
SetSerialPortConfiguration-Request	*L'élément **SerialPortToken** spécifie le port série dont la configuration doit être modifiée.* *L'élément **SerialPortConfiguration** contient la configuration de port série modifiée.* *L'élément **ForcePersistence** détermine si les modifications de configuration doivent être conservées et subsister après le redémarrage. Si la valeur est égale à "true" (vrai), les modifications doivent subsister. Si la valeur est égale à "false" (faux), les modifications PEUVENT revenir aux valeurs précédentes après le redémarrage.* tt:ReferenceToken **SerialPortToken**[1][1] tmd:SerialPortConfiguration **SerialPortConfiguration** [1][1] xs:boolean **ForcePersistence**[1][1]
SetSerialPortConfiguration-Response	*Ceci est un message vide.*
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*Le jeton de port série fourni n'existe pas.*
Env:Sender ter:InvalidArgVal ter:ConfigModify	*Les paramètres de configuration ne peuvent pas être définis.*

9.4.5 GetSerialPortConfigurationOptions

Cette opération permet de demander les options de configuration d'un port série. Un dispositif possédant un ou plusieurs ports série doit prendre en charge cette commande comme décrit dans le Tableau 94.

Tableau 94 – Commande GetSerialPortConfigurationOptions

GetSerialPortConfigurationOptions	Classe d'accès: READ_SYSTEM
Nom de message	**Description**
GetSerialPortConfigurationOptionsRequest	*L'élément **SerialPortToken** spécifie le port série dont les options sont demandées.* tt:ReferenceToken **SerialPortToken**[1][1]
GetSerialPortConfigurationOptionsResponse	*Ce message contient:* tmd:SerialPortConfigurationOptions **SerialPortConfigurationOptions** [1][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*Le jeton de port série fourni n'existe pas.*

9.4.6 Commande série Envoyer et/ou Recevoir

Le Paragraphe 9.4.6 décrit les opérations permettant de transmettre/recevoir des données de contrôle génériques vers/à partir d'un dispositif série connecté au port série du dispositif.

Cette opération peut être utilisée aux fins suivantes.

- Transmettre des données arbitraires au dispositif série connecté.
- Recevoir des données du dispositif série connecté.
- Transmettre des données arbitraires au dispositif série connecté, puis recevoir ses données de réponse.

Afin d'utiliser cette commande aux fins indiquées ci-dessus, le présent document définit la structure du paramètre d'entrée de la façon suivante.

- jeton

  Cet élément doit être présent dans la demande. Il indique le port série physique de référence à utiliser lorsque cette demande est invoquée.

- SerialData

  Cet élément peut ne pas figurer dans la demande. Lorsqu'il est nécessaire de transmettre des données en série, il convient que la demande comporte l'élément.

- TimeOut

  Cet élément peut ne pas figurer dans la demande. En fonction de la valeur spécifiée, il est possible pour les configurations suivantes.

  i) TimeOut > PT0S: Indique qu'il convient de répondre à la commande dans le délai imparti. Si le dispositif a reçu les données satisfaisant aux conditions de DataLength (longueur des données) ou de Delimiter (délimiteur), il convient qu'il renvoie ces données sans attendre la fin du délai imparti.

  ii) TimeOut = PT0S: Indique qu'il convient de répondre immédiatement à la commande (non bloquant). Cet élément est utilisé dans le cadre de la transmission de données uniquement.

  iii) TimeOut = -PT1S: Indique qu'il convient de répondre à la commande après satisfaction de l'une des conditions suivantes (DataLength/Delimiter). La durée pendant laquelle le dispositif peut maintenir l'état de blocage est *spécifique au fournisseur.*

  Si cet élément est absent de la demande, il convient de répondre à la commande après satisfaction de l'une des conditions suivantes (DataLength/Delimiter). La durée pendant laquelle le dispositif peut maintenir l'état de blocage est spécifique au fournisseur.

- DataLength

  Cet élément peut ne pas figurer dans la demande. Cet élément peut être inséré si la longueur des données renvoyées par le dispositif série connecté est déjà déterminée sous la forme d'une longueur d'octets fixe. Il indique la longueur des données reçues pouvant être considérée comme disponible.

- Delimiter

  Cet élément peut ne pas figurer dans la demande. Cet élément peut être inséré si les caractères délimiteurs renvoyés par le dispositif série connecté sont déjà connus. Il indique la séquence des données finales figurant dans la réponse. Si la chaîne comporte plusieurs caractères, le dispositif doit interpréter l'ensemble de la chaîne comme un délimiteur unique. De plus, le dispositif doit renvoyer le(s) caractère(s) de délimitation au client.

Un dispositif qui indique une fonctionnalité de service de communication série générique doit prendre en charge cette commande comme décrit dans le Tableau 95.

Tableau 95 – Commande série Envoyer et/ou Recevoir

SendReceiveSerialCommand	Classe d'accès: ACTUATE
Nom de message	**Description**
SendReceiveSerialCommandRequest	*Voir ci-dessus pour des informations sur les paramètres. Ce message contient:* tmd:SerialData**SerialData** [0][1] xs:duration **TimeOut** [0][1] xs:integer **DataLength** [0][1] xs:string **Delimiter** [0][1]
SendReceiveSerialCommandResponse	*Ce message contient les données en série.* tmd:SerialData **SerialData** [0][1]
Codes de défaut	**Description**
env:Sender ter:InvalidArgVal ter:InvalidSerialPort	*Le jeton de port série fourni n'existe pas.*
Env:Sender ter:OperationProhibited ter:DataLengthOver	*Nombre d'octets disponibles dépassé.*
Env:Sender ter:OperationProhibited ter:DelimiterNotSupported	*La séquence de caractères (délimiteur) n'est pas prise en charge.*

9.5 Fonctionnalités

Les fonctionnalités reflètent les fonctions facultatives et la fonctionnalité d'un service comme décrit dans le Tableau 96. Les informations sont statiques et ne changent pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

RelayOutputs: Nombre de sorties de relais (par défaut aucune).

DigitalInputs: Nombre d'entrées numériques (par défaut aucune).

SerialPorts: Nombre de ports série (par défaut aucune).

Tableau 96 – Commande GetServiceCapabilities

GetServiceCapabilities	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetServiceCapabilitiesRequest	*Ceci est un message vide.*
GetServiceCapabilitiesResponse	*Le message de réponse de fonctionnalité contient les fonctionnalités de service demandées en utilisant une structure de capacité XML hiérarchique.* tmd:Capabilities **Capabilities** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

9.6 Événements

9.6.1 Modification d'état DigitalInput

Un dispositif signalant la prise en charge des entrées numériques dans ses fonctionnalités doit fournir l'événement suivant chaque fois qu'une modification est apportée à l'un de ses états d'entrée:

```
Topic: tns1:Device/Trigger/DigitalInput

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="InputToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Soit une valeur "true" (vrai) est affectée à l'élément LogicalState de l'entrée numérique pour représenter le circuit dans son état fermé, soit une valeur "false" (faux) est affectée pour représenter le circuit dans son état ouvert.

9.6.2 Déclenchement de sortie de relais

Un dispositif signalant l'élément RelayOutputs dans ses fonctionnalités doit fournir l'événement Trigger (déclenchement) chaque fois que l'état d'une sortie de relais est modifié. Un dispositif doit utiliser le format de rubrique et de message suivant:

```
Topic: tns1:Device/Trigger/Relay

<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RelayToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState"
type="tt:RelayLogicalState"/>
  </tt:Data>
</tt:MessageDescription>
```

9.7 Codes de défaut spécifiques au service

Le Tableau 97 ci-dessous répertorie les codes de défaut spécifiques au service DeviceIO. De plus, chaque commande peut générer un défaut générique tel que défini dans le présent document.

Tableau 97 – Codes de défaut spécifiques au service DeviceIO

Code de défaut	Sous-code parent Sous-code	Raison de défaut	Description
env:Sender	ter:InvalidArgVal Ter:ConfigModify	Paramètres de configuration non valides	Les paramètres de configuration ne peuvent pas être définis.
env:Sender	ter:InvalidArgVal Ter:RelayToken	Référence de jeton de relais inconnue	Le RelayOutput demandé indiqué avec RelayOutputToken n'existe pas.
env:Sender	ter:InvalidArgVal ter:ModeError	Délai monostable non valide	Le délai monostable demandé n'est pas valide
env:Sender	ter:InvalidArgVal Ter:InvalidSerialPort	Jeton de port série non valide	Le jeton de port série fourni n'existe pas.
env:Sender	ter:OperationProhibited Ter:DataLengthOver	Longueur de données atteinte	Nombre d'octets disponibles dépassé.
env:Sender	ter:OperationProhibited	Le délimiteur n'est pas pris en charge	La séquence de caractères (délimiteur) n'est pas prise en charge.

10 Traitement des événements

10.1 Généralités

Un événement est une action ou occurrence détectée par un dispositif à laquelle un client peut s'abonner. Les événements sont gérés par l'intermédiaire du service d'événement. Le traitement des événements défini dans le présent document repose sur les spécifications [OASIS WS-BaseNotification] et [OASIS WS-Topics]. Le présent document étend la notion d'événement afin de permettre aux clients de suivre les propriétés d'objet (telles que les propriétés d'entrée numérique et d'alarme de mouvement) grâce aux événements. Les propriétés sont définies en 10.4.

La description de la charge utile d'événement et leur filtrage dans les abonnements est présentée en 10.5. Le Paragraphe 10.6 décrit comment le client peut demander un point de synchronisation à l'aide de l'une des interfaces de notification. Le Paragraphe 10.7 décrit l'intégration de rubriques et 10.10 décrit la gestion des défauts.

Le Paragraphe 10.11 décrit l'utilisation de l'interface de notification Real-Time Pull-Point comprenant le filtrage des messages et la définition des rubriques. Des exemples pour l'interface de notification de base peuvent être trouvés dans la spécification correspondante [OASIS WS-BaseNotification].

Un dispositif doit fournir un service d'événement tel que défini à l'Article B.3. Le dispositif et le client doivent prendre en charge [W3C WS-Addressing] pour les services d'événement

10.2 Interface de notification Real-time Pull-Point

10.2.1 Généralités

Le Paragraphe 10.2 présente l'interface de notification Real-time Pull-Point. Il s'agit d'une interface de notification conviviale de pare-feu qui permet de formuler une invitation à émettre en temps réel et initie toutes les communications de client.

Cette interface est utilisée de la façon suivante:

1) Le client demande au dispositif un pull point avec le message CreatePullPointSubscriptionRequest.
2) Le dispositif évalue la demande d'abonnement et retourne soit une CreatePullPointSubscriptionResponse, soit l'un des codes de défaut.
3) Si l'abonnement est accepté, la réponse contient une WS-EndpointReference pour le pull point instancié. Ce WS-Endpoint produit une opération PullMessages, utilisée par le client pour extraire des notifications. Il produit en outre les opérations Renew et Unsubscribe de l'interface de gestionnaire d'abonnements de base. Le schéma de séquence de l'interaction est représenté à la Figure 2.

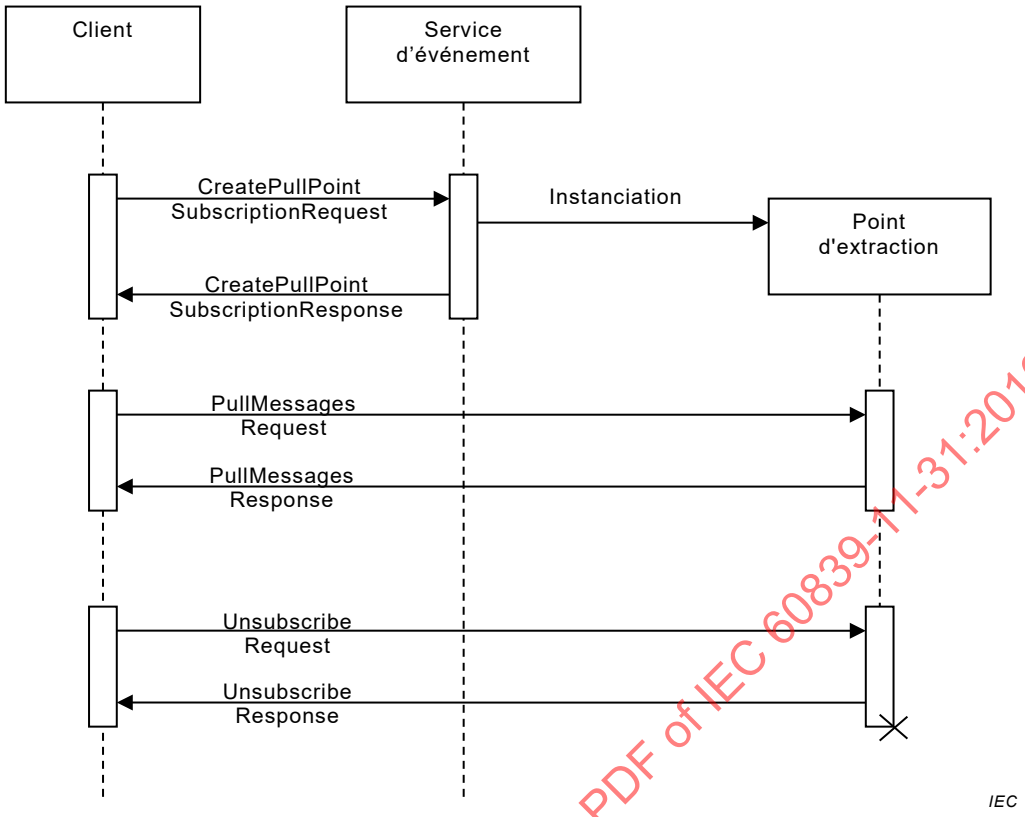


Figure 2 – Schéma de séquence de l'interface de notification Real-time Pull-Point

4) Le dispositif doit répondre immédiatement avec des notifications qui ont été agrégées au nom du client. S'il n'existe pas de notifications agrégées, le dispositif ne répond pas tant qu'une notification n'est pas produite pour le client ou que le délai d'attente spécifié n'est pas dépassé. Dans tous les cas, la réponse contient, au plus, le nombre de notifications spécifié par le paramètre MessageLimit. Le client peut grouper les notifications en temps réel au démarrage d'une nouvelle PullMessagesRequest immédiatement après chaque PullMessagesResponse.

Pour la mise en œuvre d'un dispositif, il est important de prendre en charge plusieurs pull points (y compris plusieurs pull points par client) afin de permettre une génération précise des événements. Si un dispositif ne prend en charge qu'un seul abonnement à la fois, il est nécessaire que le client souscrive à un abonnement sans aucune restriction de domaine d'application, car l'abonnement à un événement ne peut pas être modifié. Par conséquent, cela exige que le dispositif fournisse tous les événements disponibles. Il doit pour cela activer l'ensemble des sous-systèmes générant des événements. Cela peut entraîner une charge non nécessaire sur le dispositif. De plus, le trafic généré par tous ces événements peut surcharger considérablement le réseau.

Si le dispositif prend en charge le stockage continu des notifications, voir 10.2.8. Le WS-Endpoint produit également une opération Seek. Cette opération permet de replacer le pointeur d'extraction dans le passé. L'opération Seek permet également d'inverser la direction d'extraction. Il existe par ailleurs un événement BeginOfBuffer, tel que défini en 10.12, qui signale le début de la mémoire tampon.

Un dispositif doit mettre en œuvre l'interface de notification Real-time Pull-Point.

10.2.2 Création d'abonnement de point d'extraction

Un dispositif doit fournir la commande CreatePullPointSubscription comme décrit dans le Tableau 98. Si aucun élément de filtrage n'est spécifié, le pull point (point d'extraction) doit signaler au client tous les événements qui se produisent.

Par défaut, le maintien du point d'extraction est contrôlé via l'opération PullMessages. Dans ce cas, après le renvoi d'une réponse PullMessages, il convient que l'abonnement soit actif au moins pendant le délai d'attente spécifié dans la demande PullMessages.

Si un client souhaite contrôler la durée de vie du point d'extraction via des appels Renew, il doit utiliser le paramètre facultatif InitialTerminationTime. Un dispositif doit prendre en charge une valeur temporelle absolue spécifiée en UTC ainsi qu'une valeur temporelle relative pour le paramètre InitialTerminationTime. Un dispositif doit répondre aux deux paramètres CurrentTime et TerminationTime en UTC à l'aide de l'indicateur 'Z'.

Les éléments facultatifs suivants de politique d'abonnement sont définis dans tev:SubscriptionPolicy:

- **tev:ChangedOnly** Il convient qu'un pull point ne fournisse pas d'événements initiés ou supprimés pour les propriétés.

Tableau 98 – Commande CreatePullPointSubscription

CreatePullPointSubscription	Classe d'accès: READ_MEDIA
Nom de message	**Description**
CreatePullPointSubscription-Request	*Ce message contient les mêmes éléments que la SubscriptionRequest (demande d'abonnement) de la spécification [OASIS WS-BaseNotification] sans la ConsumerReference (référence de consommateur):* wsnt:FilterType **Filter** [0][1] wsnt:AbsoluteOrRelativeTimeType **InitialTerminationTime** [0][1] xs:any **SubscriptionPolicy** [0][1]
CreatePullPointSubscription-Response	*La réponse contient les mêmes éléments que la SubscriptionResponse de la [OASIS WS-BaseNotification]:* wsa:EndpointReferenceType **SubscriptionReference** [1][1] xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1]
Codes de défaut	**Description**
	Les mêmes défauts que pour la demande d'abonnement de la spécification [OASIS WS-BaseNotification] sont utilisés.

10.2.3 Messages Pull

Le dispositif doit produire la commande PullMessages suivante pour tous les points terminaux SubscriptionManager retournés par la commande CreatePullPointSubscription comme décrit dans le Tableau 99.

La commande doit au moins prendre en charge un délai d'attente d'une minute. Si un dispositif prend en charge l'extraction d'un nombre de messages inférieur au nombre demandé, il doit retourner ces messages sans générer de défaut.

Un dispositif doit répondre aux deux paramètres CurrentTime et TerminationTime en UTC à l'aide de l'indicateur 'Z'.

Après une opération de recherche, le dispositif doit retourner les messages par ordre de messages en heure UTC. À noter que cette exigence ne s'applique pas à la livraison normalisée de messages en temps réel en ce que l'ordre de livraison peut être influencé par des calculs internes au dispositif.

Il convient qu'un dispositif retourne une erreur (UnableToGetMessagesFault) après réception d'une demande PullMessages pour un abonnement si une demande de blocage des messages Pull existe déjà.

Tableau 99 – Commande PullMessages

PullMessages	Classe d'accès: READ_MEDIA
Nom de message	**Description**
PullMessagesRequest	*Ce message doit être adressé à un SubscriptionManager afin d'extraire des notifications:* xs:duration **Timeout** [1][1] xs:int **MessageLimit** [1][1]
PullMessagesResponse	*La réponse contient une liste de notifications conjointement avec un TerminationTime mis à jour pour le SubscriptionManager:* xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1] wsnt:NotificationMessageHolderType **NotificationMessage** [0][non limité]
PullMessagesFaultResponse	*Le délai d'attente (Timeout) dépasse la limite supérieure prise en charge par le dispositif. Le message de défaut doit contenir les limites supérieures pour les deux paramètres.* xs:duration **MaxTimeout**[1][1] xs:int **MaxMessageLimit**[1][1]
Codes de défaut	**Description**
	Pas de codes de défaut spécifiques.

10.2.4 Renew

Le dispositif doit produire la commande Renew suivante pour tous les points terminaux SubscriptionManager retournés par la commande CreatePullPointSubscription comme décrit dans le Tableau 100.

La commande doit au moins prendre en charge un délai d'attente d'une minute. Un dispositif doit répondre aux deux paramètres CurrentTime et TerminationTime en UTC à l'aide de l'indicateur 'Z'.

Tableau 100 – Commande Renew

Renew	Classe d'accès: READ_MEDIA
Nom de message	**Description**
RenewRequest	*Ce message contient le nouveau temps d'expiration relatif ou absolu:* wsnt:AbsoluteOrRelativeTimeType **TerminationTime** [1][1]
RenewResponse	*La réponse contient une liste de notifications conjointement avec un TerminationTime mis à jour pour le SubscriptionManager:* xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1]
ResourceUnknownFaultResponse	*La référence de point d'extraction est non valide* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
UnacceptableTerminationTimeFaultResponse	*Le délai d'attente (Timeout) dépasse la limite supérieure prise en charge par le dispositif.* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
Codes de défaut	**Description**
	Pas de codes de défaut spécifiques.

10.2.5 Unsubscribe

Le dispositif doit produire la commande Unsubscribe suivante pour tous les points terminaux SubscriptionManager retournés par la commande CreatePullPointSubscription comme décrit dans le Tableau 101.

Cette commande doit mettre fin à la vie d'un point d'extraction.

Tableau 101 – Commande Unsubscribe

Unsubscribe	Classe d'accès: READ_MEDIA
Nom de message	**Description**
UnsubscribeRequest	*Ce message est vide.*
UnsusbscribeResponse	*Ce message est vide.*
ResourceUnknownFaultResponse	*La référence de point d'extraction est non valide* xs:dateTime **Timestamp**[1][1] wsa:EndpointReferenceType **Originator**[0][1] xs:any **ErrorCode**[0][1]
Codes de défaut	**Description**
	Pas de codes de défaut spécifiques.

10.2.6 Seek

Un dispositif prenant en charge le stockage continu des notifications tel que défini en 10.2.8 doit produire la commande Seek suivante pour tous les points terminaux SubscriptionManager retournés par la commande CreatePullPointSubscription comme décrit dans le Tableau 102.

Sur une commande Seek, un pull point doit interrompre toute livraison d'événement, y compris les états initiaux des propriétés. De plus, il convient que le pull point supprime de la file d'attente de transmission les événements qui n'ont pas déjà été mis en file d'attente.

Après une demande Seek, un pull point doit ignorer le comportement décrit en 10.6 pour les propriétés.

Un dispositif doit uniquement définir l'abonnement en mode d'extraction inverse si l'argument Reverse est présent et défini sur la valeur "true".

L'argument UtcTime de la demande Seek doit être comparé à l'attribut UtcTime des notifications dans le stockage continu des notifications.

Lorsque la commande Seek est utilisée dans le mode normal, un dispositif doit positionner le pointeur d'extraction de façon à inclure tous les NotificationMessages dans le stockage continu avec un attribut UtcTime supérieur ou égal à l'argument Seek.

Lorsque la commande Seek est utilisée dans le mode inverse, un dispositif doit positionner le pointeur d'extraction de façon à inclure tous les NotificationMessages dans le stockage continu avec un attribut UtcTime inférieur ou égal à l'argument Seek.

Un dispositif ne doit pas fournir d'informations relatives à l'état initial de génération de propriété en réponse à un appel à la méthode Seek.

Tableau 102 – Commande Seek

Seek	Classe d'accès: READ_MEDIA
Nom de message	**Description**
SeekRequest	*Ce message doit être adressé à un PullPoint afin de réajuster la position d'extraction:* xs:datetime Utc**Time**[1][1] xs:bool **Reverse**[0][1]
SeekResponse	*Ce message est vide*
Codes de défaut	**Description**
	Pas de codes de défaut spécifiques.

10.2.7 Cycle de vie d'un pull point

La Figure 2 décrit le fonctionnement de base d'un point d'extraction. Le Paragraphe 10.2.7 énonce les exigences applicables au cycle de vie d'un point d'extraction.

Un dispositif doit créer un nouveau point d'extraction sur chaque commande CreatePullPointSubscription tant que le nombre de points d'extraction instanciés ne dépasse pas la fonctionnalité MaxPullPoints. Chaque point d'extraction doit avoir une référence de point terminal unique vers laquelle le client peut diriger ses opérations consécutives sur le point d'extraction.

Un point d'extraction doit exister jusqu'à ce que son temps d'expiration soit écoulé, ou bien jusqu'à ce que le client requière sa suppression par l'intermédiaire d'une demande Unsubscribe. Aucune exigence n'est applicable à la persistance d'un point d'extraction à travers le cycle d'alimentation d'un dispositif.

10.2.8 Stockage continu des notifications

Un dispositif peut stocker les notifications d'un client afin de s'assurer qu'il n'en perde aucune. Les notifications stockées peuvent être extraites à tout moment par le client. Le dispositif doit indiquer s'il prend en charge le stockage continu des notifications avec la fonctionnalité PersistentNotificationStorage. Voir 10.9.

Le présent document précise que l'interface vers le stockage continu permet un accès linéaire via l'heure de l'événement de création de message. Cela s'applique également aux événements livrés en désordre dans le cas d'une diffusion en continu par suite d'un problème informatique, par exemple.

Les détails relatifs aux notifications, à la façon dont elles sont réellement stockées et à leur emplacement de stockage ne relèvent pas du domaine d'application du présent document. La politique de suppression des notifications stockées permettant d'inclure de nouvelles notifications ne relève également pas du domaine d'application du présent document.

10.3 Interface de notification de base

10.3.1 Généralités

Le Paragraphe 10.3.2 présente brièvement l'interface de notification de base de la spécification [OASIS WS-BaseNotification]. Le Paragraphe 10.3.3 résume les interfaces obligatoires et facultatives de la spécification [OASIS WS-BaseNotification]. Se référer à la spécification [OASIS WS-BaseNotification] pour une documentation complète sur l'interface de notification de base.

10.3.2 Résumé

Les entités logiques suivantes participent au profil de notification:

- Client: met en œuvre l'interface NotificationConsumer.
- Service d'événement: met en œuvre l'interface NotificationProducer.
- Gestionnaire d'abonnements: met en œuvre l'interface BaseSubscriptionManager.

Il convient d'instancier le service d'événement et le gestionnaire d'abonnements sur un dispositif.

Des messages types échangés entre les entités sont présentés dans le schéma de séquence de la Figure 3. Le client établit d'abord une connexion au service d'événement. Ensuite, il peut s'abonner pour certaines notifications en envoyant une SubscriptionRequest. Si le service d'événement accepte l'abonnement, il instancie de manière dynamique un

SubscriptionManager représentant l'abonnement. Le service d'événement doit retourner le WS-Endpoint-Address du SubscriptionManager dans la SubscriptionResponse.

Afin de transmettre des notifications correspondant à l'abonnement, une autre connexion est établie entre le service d'événement et le client. Grâce à cette connexion, le service d'événement envoie un message de notification unilatéral à l'interface NotificationConsumer du client. Les notifications correspondantes peuvent être envoyées à tout moment au client par le service d'événement, tant que l'abonnement est actif.

Pour contrôler l'abonnement, le client adresse directement le SubscriptionManager retourné dans la SubscriptionResponse. Dans la SubscriptionRequest, le client peut spécifier un temps d'expiration. Le SubscriptionManager est automatiquement détruit lorsque le temps d'expiration est atteint. Des RenewRequests peuvent être initiées par le client afin de retarder le temps d'expiration. Le client peut également terminer le SubscriptionManager de manière explicite en envoyant une UnsubscribeRequest. Après un désabonnement réussi, le SubscriptionManager n'existe plus.

L'interaction entre un EventService et un SubscriptionManager n'est pas décrite de manière plus détaillée par la spécification [OASIS WS-BaseNotification] et dépend de la mise en œuvre du dispositif.

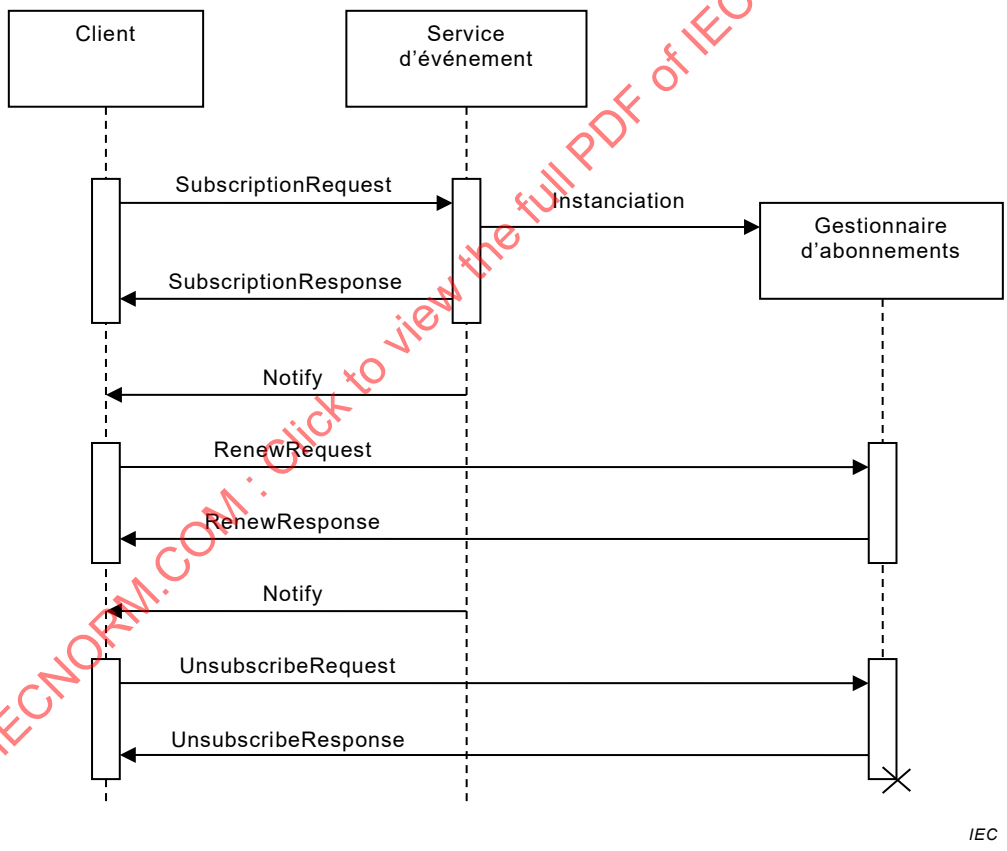


Figure 3 – Schéma de séquence de l'interface de notification de base

10.3.3 Exigences

Le Paragraphe 10.3.3 décrit de manière détaillée les interfaces de [OASIS WS-BaseNotification] qu'un dispositif doit fournir.

Un dispositif doit prendre en charge l'interface NotificationProducer de la spécification [OASIS WS-BaseNotification] si la fonctionnalité MaxNotificationProducers n'est pas égale à zéro. Le dispositif doit prendre en charge les filtres TopicExpression avec les dialectes décrits en

10.7.4. La prise en charge des filtres MessageContent est signalée par l'intermédiaire de la méthode GetEventProperties. Si le dispositif n'accepte pas l'InitialTerminationTime d'un abonnement, il doit en fournir un valide dans le message de défaut. Le dispositif doit être capable de produire des notifications en utilisant l'enveloppe Notify de la spécification [OASIS WS-BaseNotification]. La SubscriptionPolicy `wsnt:UseRaw` est facultative pour le dispositif. Bien que [OASIS WS-BaseNotification] ait un CurrentTime et un TerminationTime en tant qu'éléments facultatifs dans un SubscribeResponse et un RenewResponse, un dispositif doit les énumérer dans des SubscribeResponses et des RenewResponse. Le dispositif peut répondre à une demande GetCurrentMessage par un message de défaut indiquant qu'aucun message actuel n'est disponible sur la rubrique demandée.

La mise en œuvre de l'interface Pull-Point de la spécification [OASIS WS-BaseNotification] sur un dispositif est facultative.

Un dispositif doit mettre en œuvre l'interface de gestionnaire d'abonnements de base de la spécification [OASIS WS-BaseNotification] composée des opérations Renew (Renouveler) et Unsubscribe (Désabonner). L'interface de gestionnaire d'abonnements pausable est facultative. La mise en œuvre des abonnements en tant que WS-Resources est facultative.

Un dispositif doit prendre en charge les valeurs temporelles des paramètres de demande données en UTC avec l'indicateur 'Z' et répondre à toutes les valeurs temporelles en UTC, y compris l'indicateur 'Z'.

10.4 Propriétés

Une propriété est un ensemble de paires nom-valeur représentant un ensemble de données unique et adressable. Ces paires sont identifiées de manière unique par la combinaison de leurs valeurs de rubrique, de source et de clé, et sont formatées comme des événements ordinaires. Une propriété contient également un drapeau supplémentaire indiquant s'il s'agit d'une propriété créée, modifiée ou supprimée.

Si un client s'abonne à une rubrique représentant une certaine propriété, le dispositif doit informer le client de tous les objets comportant la propriété demandée et qui sont actifs lors de l'abonnement. Un client peut également demander à tout moment les valeurs de toutes les propriétés actuellement actives auxquelles le client s'est abonné, en demandant un point de synchronisation (voir 10.6).

L'interface de propriété est définie dans le présent document afin de grouper tous les événements associés à des propriétés et les présenter uniformément aux clients. Il est recommandé d'utiliser l'interface de propriété dans la mesure du possible. Le Paragraphe 10.5 décrit de manière détaillée la structure des événements et des propriétés.

10.5 Structure des notifications

10.5.1 Généralités

Le code suivant est le schéma de la notification `wsnt:NotificationMessage` [OASIS WS-BaseNotification]:

```
<xs:complexType name="NotificationMessageHolderType" >
  <xs:sequence>
    <xs:element ref="wsnt:SubscriptionReference" minOccurs="0" />
    <xs:element ref="wsnt:Topic" minOccurs="0" />
    <xs:element ref="wsnt:ProducerReference" minOccurs="0" />
    <xs:element name="Message">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
```

```
  </xs:sequence>
</xs:complexType>

<xs:element name="NotificationMessage"
            type="wsnt:NotificationMessageHolderType"/>
```

Cela correspond à la structure XML suivante:

```
<wsnt:NotificationMessage>
  <wsnt:SubscriptionReference>
    wsa:EndpointReferenceType
  </wsnt:SubscriptionReference>
  <wsnt:Topic Dialect="xs:anyURI">
    …
  </wsnt:Topic>?
  <wsnt:ProducerReference>
    wsa:EndpointReferenceType
  </wsnt:ProducerReference>
  <wsnt:Message>
    …
  </wsnt:Message>
</wsnt:NotificationMessage>
```

où l'élément `wsnt:Message` contient la charge utile de notification réelle. Le type XML de l'élément Message peut être spécifié dans une définition TopicTree, voir 10.7.

Le Paragraphe 10.5.2 présente les informations qu'un client extrait à l'aide de notifications. Le Paragraphe 10.5.3 présente un formatage détaillé de la charge utile du message, et 10.5.4 un langage de description pour la charge utile du message. Le Paragraphe 10.5.5 définit la grammaire utilisée dans un abonnement pour filtrer les notifications en fonction du contenu de leur message.

10.5.2 Informations de notification

10.5.2.1 Généralités

Une notification répond au moins aux questions suivantes:

- Quand cela s'est-il produit?
- Qui a généré l'événement?
- Que s'est-il passé?

La réponse à la question "quand?" est obtenue en ajoutant un attribut temporel à l'élément Message de l'objet NotificationMessage. Un dispositif doit inclure l'attribut temporel dans l'élément Message.

La question "qui?" est divisée en deux parties. Une partie est le point terminal WS-Endpoint qui identifie le dispositif ou un service dans le dispositif dans lequel la notification a été générée. Par conséquent, il convient que WS-Endpoint soit spécifié dans l'élément ProducerReference du NotificationMessage. La deuxième partie est l'identification du composant dans WS-Endpoint, qui est responsable de la production de la notification. Selon le composant, plusieurs paramètres ou aucun peuvent être nécessaires pour identifier le composant de façon unique. Ces paramètres sont placés en tant qu'éléments (Items) dans l'élément Source du conteneur de Message.

La réponse à la question "que?" est obtenue en deux étapes. Premièrement, l'élément Topic du NotificationMessage est utilisé pour classer l'événement. Deuxièmement, des éléments sont ajoutés à l'élément Data du conteneur de Message afin de décrire les détails de l'événement.

Lorsque la rubrique pointe sur des propriétés, voir 10.4, le client utilise les éléments NotificationProducer, Topic et Source ainsi que les éléments Key facultatifs, voir 10.5.3, pour identifier la propriété. Ces valeurs doivent conduire à un identifiant unique.

10.5.2.2 Exemple d'événement

L'exemple suivant décrit les différentes parties de la notification:

```
<wsnt:NotificationMessage>
  ...
  <wsnt:Topic Dialect="...Concrete">
    tns1:Device/Trigger/DigitalInput
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="..." PropertyOperation="...">
      <tt:Source>
        <tt:SimpleItem Name="InputToken"       Value="2"/>
      </tt:Source>
      <tt:Data>
        <tt:SimpleItem Name="LogicalState"     Value="true"/>
      </tt:Data>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

L'élément "InputToken" identifie de manière unique le composant responsable de la détection de l'événement. Dans cet exemple, le composant est une entrée numérique référencée par le jeton "2". L'événement `tns1:Device/Trigger/DigitalInput` indique que l'état de l'entrée numérique a changé. Le bloc de données contient l'état.

10.5.3 Format de message

L'élément Message du NotificationMessage est défini dans le schéma commun, voir l'Article B.4. La définition est présentée ci-dessous:

NOTE Le schéma est inclus ici à titre informatif uniquement. L'Article B.4 contient la définition de schéma normative.

```
<xs:element name="Message" type="Message">

<xs:element name="Message">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Source" type="tt:ItemList" minOccurs="0"/>
      <xs:element name="Key" type="tt:ItemList" minOccurs="0"/>
      <xs:element name="Data" type="tt:ItemList" minOccurs="0"/>
      ...
    </xs:sequence>

    <xs:attribute name="UtcTime" type="xs:dateTime" use="required"/>
    <xs:attribute name="PropertyOperation" type="tt:PropertyOperationType"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="ItemList">
  <xs:sequence>
    <xs:element name="SimpleItem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Value" type="xs:anySimpleType" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ElementItem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any"/>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
```

```
</xs:complexType>

<xs:simpleType name="PropertyOperationType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Initialized"/>
    <xs:enumeration value="Deleted"/>
    <xs:enumeration value="Changed"/>
  </xs:restriction>
</xs:simpleType>
```

Les éléments de l'élément Message sont groupés en trois catégories: Source, Key et Data. Le groupe Key ne doit pas être utilisé par des notifications qui ne sont pas associées à des propriétés. Plusieurs éléments Simple et Element peuvent être placés dans chaque groupe. Chaque élément a un nom et une valeur. Dans le cas d'un élément ElementItem, la valeur est exprimée par un élément XML dans l'élément ElementItem. Dans le cas d'un élément SimpleItem, la valeur doit être spécifiée par l'attribut de valeur. Le nom de tous les éléments doit être unique pour l'ensemble des éléments contenus dans un groupe du message en question.

Les extensions spécifiques au fournisseur doivent exprimer les attributs SimpleItem Name et ElementItem Name sous forme de qname. Cela permet de prévenir les éventuels conflits de nom entre les extensions spécifiques au fournisseur et les extensions futures de le présent document.

Dans la mesure du possible, il est recommandé d'utiliser des éléments SimpleItem plutôt que des éléments ElementItem, étant donné que les éléments SimpleItem facilitent l'intégration de messages dans un client générique. Les informations de type exactes des éléments SimpleItem et ElementItem peuvent être extraites du TopicSet (voir 10.7), où chaque rubrique peut être complétée par une description de la charge utile du message.

L'élément PropertyOperation doit être présent si la notification concerne une propriété. Le mode de fonctionnement "Initialized" (Initialisé) doit être utilisé pour informer un client de la création d'une propriété. Le mode de fonctionnement "Initialized" doit être utilisé si un point de synchronisation a été demandé.

10.5.4 Langage de description de message

10.5.4.1 Généralités

La structure de la charge utile de message a été présentée en 10.5.3. La structure contient trois groupes: Source, Key et Data. Chaque groupe contient un ensemble d'éléments SimpleItem et ElementItem. Pour chaque rubrique, un dispositif peut décrire l'élément faisant partie d'une notification générée par la rubrique en question, à l'aide d'un langage de description de message. Le langage de description suivant décrit les éléments de message obligatoires:

NOTE Le schéma est inclus ici à titre informatif uniquement. L'Article B.4 contient la définition de schéma normative.

```
<xs:complexType name="MessageDescription">
  <xs:sequence>
    <xs:element name="Source" type="tt:ItemListDescription"
                minOccurs="0"/>
    <xs:element name="Key" type="tt:ItemListDescription" minOccurs="0"/>
    <xs:element name="Data" type="tt:ItemListDescription" minOccurs="0"/>
    ...
  </xs:sequence>
  <xs:attribute name="IsProperty" type="xs:boolean"/>
</xs:complexType>

<xs:complexType name="ItemListDescription">
  <xs:sequence>
    <xs:element name="SimpleItemDescription"
                minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
```

```
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:QName" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ElementItemDescription"
                minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Type" type="xs:QName" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

L'attribut Name d'un élément doit être unique pour l'ensemble des éléments, quel que soit le groupe (Source, Key, Data) dont il provient. La valeur "true" doit être affectée à l'attribut IsProperty lorsque le message décrit concerne une propriété. Toutefois, si le message ne concerne pas une propriété, le groupe Key ne doit pas être présent. L'attribut Type d'un SimpleItemDescriptor doit utiliser un type simple défini dans le schéma XML (conçu dans des types simples), des schémas définis par le présent document, ou des schémas de fournisseur. De même, l'attribut Type d'un élément ElementItemDescriptor doit correspondre à une déclaration d'élément globale d'un schéma XML.

Le langage de description de message ne définit pas l'ordre des éléments dans chacune des catégories Source, Key et Data. De plus, il n'est pas exigé que les éléments décrits comme facultatifs par une définition d'événement soient présents dans un message. Cela s'applique également aux éléments facultatifs décrits dans la MessageDescription associée.

L'emplacement de tous les fichiers de schéma utilisés pour décrire des charges utiles de message figure dans le message GetEventPropertiesResponse de 10.8.

10.5.4.2 Exemple de description de message

Le code suivant est un exemple de description de message correspondant à l'exemple d'événement de 10.5.2.2:

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="InputToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="LogicalState" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

10.5.5 Filtre de contenu de message

Dans la demande d'abonnement, un client peut filtrer les notifications par TopicExpression (voir 10.7.4) et MessageContent. Pour ce dernier, le [OASIS WS-BaseNotification] propose le dialecte XPath 1.0. En raison de la structure de message spécifique exigée par le présent document, la spécification exige un sous-ensemble de la syntaxe [W3C Xpath]. Le dialecte correspondant peut être référencé avec l'URI suivant:

```
Dialect=http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter
```

Priorité et associativité:

L'opération "and" a une priorité plus élevée que l'opération "or". Ces deux opérations restent associatives.

La priorité et l'associativité des opérations "and" et "or" dans la définition grammaticale ci-dessous sont identiques aux spécifications XPath 1.0.

La structure des expressions se présente comme suit:

[1] Expression::= BoolExpr | Expression ‘and’ Expression
| Expression ‘or’ Expression | ‘(‘ Expression ‘)’ | ‘not’ ‘(‘ Expression ‘)’

[2] BoolExpr::= ‘boolean’ ‘(‘ PathExpr ‘)’

[3] PathExpr::= [‘//’Prefix?’SimpleItem‘ | ’//’Prefix?’ElementItem‘] NodeTest

[4] Prefix::= NamespacePrefix’:’ | ‘’

[5] NodeTest::= ‘[‘ AttrExpr ‘]’

[6] AttrExpr::= AttrComp | AttrExpr ‘and’ AttrExpr | AttrExpr ‘or‘ AttrExpr | ‘(‘ AttrExpr ‘)‘
| ‘not’ ‘(‘ AttrExpr ‘)‘

[7] AttrComp::= Attribute ‘=’ ‘”’ String ‘”’

[8] Attribute::= ‘@Name’ | ‘@Value’

Cette syntaxe permet de soumettre à essai la présence d'éléments SimpleItem ou ElementItem indépendamment du groupe auquel ils appartiennent (Source, Key ou Data). De plus, la valeur d’éléments SimpleItem peut être vérifiée. L’espace de nommage de préfixe d'élément SimpleItem et ElementItem doit correspondre à “http://www.onvif.org/ver10/schema.

Enfin, des combinaisons booléennes arbitraires de ces essais sont possibles. Les expressions suivantes peuvent être formulées:

- Retourner uniquement les notifications qui contiennent une référence à l'InputToken “1”
  `boolean(//tt:SimpleItem[@Name=”` InputToken`” and @Value=”1”] )`
- Retourner uniquement les notifications qui ne contiennent pas de référence à un RelayToken
  `not(boolean(//tt:SimpleItem[@Name=”RelayToken”]))`
- Retourner uniquement les notifications qui concernent l'InputToken “2” avec le LogicalState “true”

```
boolean(//tt:SimpleItem[@Name=” InputToken” and @Value=”2”])
and
boolean(//tt:SimpleItem[@Name=”LogicalState” and @Value=”true”])
```

10.6 Point de synchronisation

À noter que 10.2.6 définit des règles applicables aux dispositifs prenant en charge le stockage continu des notifications qui supplantent le comportement défini dans le présent paragraphe.

Les propriétés, présentées en 10.4, informent un client de la création, des modifications et de la suppression de propriétés de manière uniforme. Lorsqu’un client souhaite synchroniser ses propriétés avec les propriétés du dispositif, il peut demander un point de synchronisation qui répète le statut actuel de toutes les propriétés auxquelles un client s’est abonné comme décrit dans le Tableau 103. L'élément PropertyOperation de toutes les notifications générées est défini sur "Initialized" (voir 10.5). Le point de synchronisation est directement demandé au SubscriptionManager qui a été retourné dans SubscriptionResponse ou dans CreatePullPointSubscriptionResponse. La mise à jour de propriété est transmise via le transport de notification de l’interface de notification. L’opération suivante doit être générée par tous les points terminaux de gestionnaire d'abonnements:

Tableau 103 – Commande SetSynchronizationPoint

SetSynchronizationPoint	Classe d'accès: READ_MEDIA
Nom de message	**Description**
SetSynchronizationPointRequest	*Ce message est vide.*
SetSynchronizationPointResponse	*Ce message est vide.*
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

10.7 Structure de rubrique

10.7.1 Généralités

Le présent document étend le cadre de Topic défini dans la spécification [OASIS WS-Topics].

Le Paragraphe 10.7.2 décrit l'espace de nommage de rubrique ONVIF. Le Paragraphe 10.7.3 intègre le langage de description de message défini en 10.5.4 dans la structure TopicSet et 10.8 définit une interface permettant à un client d'obtenir ces informations à partir d'un dispositif. Le Paragraphe 10.7.4 définit les dialectes d'expression de rubrique qui doivent être pris en charge par un dispositif.

Les définitions d'événements concrets sont spécifiées dans les sections d'événement des spécifications de service.

10.7.2 Espace de nommage de rubrique ONVIF

La spécification [OASIS WS-Topics] distingue la définition d'une arborescence de rubriques appartenant à un certain espace de nommage de rubrique et l'ensemble de rubriques pris en charge par un certain service Web. Cette distinction permet aux fournisseurs de se référer à un espace de nommage de rubrique commun tout en utilisant seulement une partie des rubriques définies.

Si l'arborescence de rubriques d'un espace de nommage de rubrique existant couvre uniquement un sous-ensemble des rubriques disponibles par un dispositif, l'arborescence de rubriques peut être développée en définissant un nouvel espace de nommage de rubrique. Un nouvel espace de nommage de rubrique est défini en ajoutant une nouvelle rubrique à un espace de nommage de rubrique existant comme décrit dans la spécification [OASIS WS-Topics].

Toutes les notifications faisant référence aux rubriques dans l'espace de nommage de rubrique ONVIF, http://www.onvif.org/ver10/topics, doivent utiliser le format de message tel que décrit en 10.5.3.

10.7.3 Informations de type de rubrique

Un dispositif doit ajouter un élément MessageDescription, de type MessageDescriptionType défini en 10.5.4, au-dessous de tous les éléments représentant des rubriques dans l'ensemble de rubriques pris en charge par le dispositif. De plus, un dispositif doit, conformément à la spécification de notification, identifier tous les éléments représentant des rubriques dans l'ensemble de rubriques en insérant l'attribut wstop:topic avec la valeur "true".

L'exemple suivant décrit comment des rubriques d'un élément TopicSet sont ajoutées avec des descriptions de message:

```
<wstop:TopicSet xmlns="">
  <tns1:Device>
    <Trigger>
      <DigitalInput wstop:topic="true">
```

```
        <tt:MessageDescription IsProperty="true">
          <tt:Source>
            <tt:SimpleItemDescription Name="InputToken"
                                      Type="tt:ReferenceToken"/>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="LogigalState" Type="xs:boolean"/>
          </tt:Data>
        </tt:MessageDescription>
      </ DigitalInput>
      <Relay wstop:topic="true">
        <tt:MessageDescription IsProperty="true">
          <tt:Source>
            <tt:SimpleItemDescription Name="RelayToken"
                                      Type="tt:ReferenceToken"/>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="LogicalState"
                                      Type="xs:RelayLogicalState"/>
          </tt:Data>
        </tt:MessageDescription>
      </Relay>
    </Trigger>
  </tns1:Device>
</wstop:TopicSet>
```

NOTE xmlns="" est inclus dans l'exemple pour s'assurer qu'aucun espace de nommage par défaut n'est présent dans le domaine d'application pour l'un quelconque des descendants de l'élément TopicSet. Voir la spécification [WS-Topics] pour plus d'informations.

10.7.4 Filtre de rubrique

Un dispositif doit prendre en charge les expressions de rubrique concrètes définies dans la spécification [OASIS WS-Topics]. Le présent document définit l'identification d'une rubrique spécifique dans des arborescences de rubriques. Le dialecte suivant doit être spécifié lorsqu'une expression de rubrique concrète est utilisée en tant que TopicExpression d'un filtre d'abonnement:

```
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete
```

La syntaxe d'expression de rubrique suivante doit être prise en charge par un dispositif.

La syntaxe étend les expressions de rubrique concrètes par une opération "or" et une chaîne de correspondance de sous-arborescence de rubrique. Cette syntaxe étendue permet de sélectionner un TopicSet arbitraire dans un seul abonnement. La syntaxe est décrite de la même façon que les expressions de rubrique de la spécification [OASIS WS-Topics]:

[3] TopicExpression::= TopicPath ('|' TopicPath)*

[4] TopicPath::= RootTopic ChildTopicExpression* *('//.')?*

[5] RootTopic::= QName

Si un préfixe d'espace de nommage est inclus dans l'élément RootTopic, il doit correspondre à une définition d'espace de nommage de rubrique valide, et le nom local doit correspondre au nom d'une rubrique racine définie dans cet espace de nommage.

[6] ChildTopicExpression::= '/' ChildTopicName

[7] ChildTopicName::= QName | NCName

Le NCName ou la partie locale du QName doit correspondre au nom d'une rubrique dans le chemin descendant depuis l'élément RootTopic, chaque barre oblique indiquant un autre niveau d'éléments de rubrique enfants dans le chemin.

Afin de référencer ce dialecte TopicExpression, l'URI suivant doit être utilisé:

```
Dialect=http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
```

Si TopicExpression se termine par les caractères "//.", cela indique que TopicExpression correspond à une sous-arborescence de rubriques. Par exemple:

"tns1:Device/Trigger //."

Cela identifie la sous-arborescence composée de tns1:Device/Trigger et de tous ses descendants.

Les exemples suivants décrivent l'utilisation de l'élément topicExpression de ConcreteSet:

Rechercher les notifications qui ont la rubrique tns1:Monitoring en tant que rubrique parent:

```
<wsnt:TopicExpression Dialect =
```

- "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet" >
  - tns1:Monitoring//.

```
</wsnt:TopicExpression>
```

Rechercher les notifications qui ont la rubrique tns1:Monitoring ou tns1:Device en tant que rubrique parent:

```
<wsnt:TopicExpression Dialect =
```

- "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet" >
  - tns1:Monitoring/ |tns1:Device//.

```
</wsnt:TopicExpression>
```

Rechercher les notifications qui correspondent soit à la rubrique tns1:Monitoring/ProcessorUsage, soit à la rubrique tns1:Device/Trigger/DigitalInput:

```
<wsnt:TopicExpression Dialect =
```

- "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
  - tns1:Monitoring/ProcessosUsage|tns1:Device/Trigger/DigitalInputs

```
</wsnt:TopicExpression>
```

10.8 Obtention de propriétés d'événement

La spécification [OASIS WS-BaseNotification] définit un ensemble de propriétés WS-ResouceProperties facultatives. Le présent document n'exige pas la mise en œuvre de l'interface WS-ResourceProperty. Au lieu de cela, l'interface directe suivante doit être mise en œuvre par un dispositif afin de fournir des informations sur les éléments FilterDialect, les fichiers de schéma et les rubriques pris en charge par le dispositif comme décrit dans le Tableau 104.

Tableau 104 – Commande GetEventProperties

GetEventProperties	Classe d'accès: READ_MEDIA
Nom de message	**Description**
GetEventPropertiesRequest	*Ceci est un message vide.*
GetEventPropertiesResponse	*Ce message contient:* xs:anyURI **TopicNamespaceLocation** [1][non limité] xs:boolean **FixedTopicSet** [1][1] wstop:TopicSetType **TopicSet** [1][1] xs:anyURI **TopicExpressionDialect** [1][non limité] xs:anyURI **MessageContentFilterDialect** [1][non limité] xs:anyURI **ProducerPropertiesFilterDialect** [0][non limité] xs:anyURI **MessageContentSchemaLocation** [1][non limité]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

Un dispositif doit répondre et déclarer si son élément TopicSet est fixe ou non, et indiquer les rubriques fournies et les dialectes pris en charge.

Les éléments TopicExpressionDialect suivants sont obligatoires pour un dispositif (voir 10.7.4):

- `http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete`
- `http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet`

Un dispositif qui ne prend en charge aucun MessageContentFilterDialect doit retourner une seule URLvide.

Le présent document n'exige pas la prise en charge d'un élément ProducerPropertiesDialect par un dispositif.

10.9 Fonctionnalités

Le langage de description de contenu de message, présenté en 10.5.4, permet de référencer des types spécifiques au fournisseur. Pour faciliter l'intégration de ces types dans une application client, l'élément GetEventPropertiesResponse comme décrit dans le Tableau 105 doit énumérer tous les emplacements d'URI vers les fichiers de schéma dont les types sont utilisés dans la description de notifications, avec des éléments MessageContentSchemaLocation. Cette liste doit contenir au moins l'URI du fichier de schéma commun.

Les fonctionnalités reflètent les fonctions facultatives et la fonctionnalité d'un service. Les informations sont statiques et ne changent pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

WSSubscriptionPolicySupport: Indique si le dispositif prend en charge la politique WS Subscription (abonnement) selon 10.3.3.

WSPullPointSupport: Indique si le dispositif prend en charge le WS PullPoint selon 10.3.3.

WSPausableSubscriptionManagerInterfaceSupport: Indique si le dispositif prend en charge l'interface WS Pausable Subscription Manager selon 10.3.3.

MaxNotificationProducers: Nombre maximal de générateurs de notification pris en charge tel que défini par [OASIS WS-BaseNotification].

MaxPullPoints: Nombre maximal de points d'extraction de notifications pris en charge.

PersistenNotificationStorage: Indique si le dispositif prend en charge le stockage continu des notifications selon 10.2.8.

Tableau 105 – Commande GetServiceCapabilities

GetServiceCapabilities	Classe d'accès: PRE_AUTH
Nom de message	**Description**
GetServiceCapabilitiesRequest	*Ceci est un message vide.*
GetServiceCapabilitiesResponse	*Le message de réponse de fonctionnalité contient les fonctionnalités de service demandées en utilisant une structure de capacité XML hiérarchique.* tev:Capabilities **Capabilities** [1][1]
Codes de défaut	**Description**
	Pas de défauts spécifiques à la commande!

10.10 Messages de défaut SOAP

Si un dispositif rencontre un échec pendant le traitement de messages [OASIS WS-BaseNotification] provenant d'un client ou d'un gestionnaire d'abonnements, il doit générer un message de défaut SOAP 1.2.

Tous les messages de défaut SOAP 1.2 doivent être générés conformément aux spécifications [OASIS WS-BaseNotification] et [OASIS WS-Topics] à une exception près; tous les défauts doivent utiliser l'URI suivant pour la propriété d'adressage de message WS-Addressing [action]:

```
http://www.w3.org/2005/08/addressing/soap/fault
```

De plus, il convient que l'erreur soit envoyée en tant que défaut de récepteur SOAP (env:Receiver), c'est-à-dire, le code d'erreur HTTP doit être 500.

10.11 Exemple de notification

10.11.1 Généralités

L'exemple suivant est un profil de communication complet pour des notifications. Il utilise l'interface de notification Real-time Pull-Point pour recevoir des notifications.

10.11.2 GetEventPropertiesRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesReq
uest
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:GetEventProperties>
    </tet:GetEventProperties>
  </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

10.11.3 GetEventPropertiesResponse

Dans cet exemple, la réponse du dispositif utilise l'espace de nommage de rubrique ONVIF. L'ensemble de rubriques ne change pas au cours du temps et est composé de la seule rubrique `tns1:Device/Trigger/DigitalInput`. Le dispositif prend en charge deux éléments TopicExpressionDialect.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
  xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
  xmlns:tns1="http://www.onvif.org/ver10/topics"
  xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRespons
e
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:GetEventPropertiesResponse>
      <tet:TopicNamespaceLocation>
        http://www.onvif.org/onvif/ver10/topics/topicns.xml
      </tet:TopicNamespaceLocation>
      <wsnt:FixedTopicSet>
        true
      </wsnt:FixedTopicSet>
      <wstop:TopicSet xmlns="">
        <tns1:Device>
          <Trigger>
            <DigitalInput wstop:topic="true">
              <tt:MessageDescription IsProperty="true">
                <tt:Source>
                  <tt:SimpleItemDescription Name="InputToken"
                                            Type="tt:ReferenceToken"/>
                </tt:Source>
                <tt:Data>
                  <tt:SimpleItemDescription Name="LogigalState"
                                            Type="xs:boolean"/>
                </tt:Data>
              </tt:MessageDescription>
            </DigitalInput>
          </Trigger>
        </tns1:Device>
      </wstop:TopicSet>
      <wsnt:TopicExpressionDialect>
        http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
      </wsnt:TopicExpressionDialect>
      <wsnt:TopicExpressionDialect>
        http://docs.oasis-open.org/wsnt/t-1/TopicExpression/ConcreteSet
      </wsnt:TopicExpressionDialect>
      <wsnt:MessageContentFilterDialect>
        http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter
      </wsnt:MessageContentFilterDialect>
      <tt:MessageContentSchemaLocation>
        http://www.onvif.org/onvif/ver10/schema/onvif.xsd
      </tt:MessageContentSchemaLocation>
    </tet:GetEventPropertiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.4 CreatePullPointSubscription

Un client peut s'abonner à des notifications spécifiques avec les informations provenant des éléments TopicProperties. L'exemple XML suivant présente l'abonnement pour des

notifications générées sous la rubrique racine du dispositif. Le délai d'attente de l'abonnement est d'une minute. Si l'abonnement n'est pas explicitement renouvelé ou que les messages ne sont pas extraits régulièrement, ils prennent fin automatiquement à l'issue de ce délai.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
  xmlns:tns1="http://www.onvif.org/ver10/topics">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscripti
onRequest
    </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
    <tet:CreatePullPointSubscription>
      <tet:Filter>
        <wsnt:TopicExpression
         Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
          tns1:Device//.
        </wsnt:TopicExpression>
      </tet:Filter>
      <tet:InitialTerminationTime>
          PT1M
      </tet:InitialTerminationTime>
    </tet:CreatePullPointSubscription>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.5 CreatePullPointSubscriptionResponse

Si le dispositif accepte l'objet Subscription, il retourne l'URI `http://160.10.64.10/Subscription?Idx=0` qui représente le point terminal de cet abonnement. De plus, le client est informé du CurrentTime du dispositif et du TerminationTime de l'abonnement créé.

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscr
iptionResponse
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:CreatePullPointSubscriptionResponse>
      <tet:SubscriptionReference>
        <wsa:Address>
          http://160.10.64.10/Subscription?Idx=0
        </wsa:Address>
      </tet:SubscriptionReference>
      <wsnt:CurrentTime>
        2008-10-09T13:52:59
      </wsnt:CurrentTime>
      <wsnt:TerminationTime>
        2008-10-09T13:53:59
      </wsnt:TerminationTime>
    </tet:CreatePullPointSubscriptionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.6 PullMessagesRequest

Le client envoie une PullMessagesRequest au point terminal indiqué dans la CreatePullPointSubscriptionResponse pour obtenir les notifications correspondant à un certain abonnement. L'exemple de demande suivant contient un délai d'attente (Timeout) de cinq (5) secondes et limite à deux (2) le nombre total de messages dans la réponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:tet="http://www.onvif.org/ver10/events/wsdl" >
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesR
equest        </wsa:Action>
    <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:PullMessages>
      <tet:Timeout>
       PT5S
      </tet:Timeout>
      <tet:MessageLimit>
        2
      </tet:MessageLimit>
    </tet:PullMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.7 PullMessagesResponse

La PullMessageResponse suivante contient des notifications qui correspondent à l'abonnement. La réponse indique au client que l'état de DigitalInput "1" est passé à "true".

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tet="http://www.onvif.org/ver10/events/wsdl"
    xmlns:tns1="http://www.onvif.org/ver10/topics"
    xmlns:tt="http://www.onvif.org/ver10/schema">
    <SOAP-ENV:Header>
      <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRespo
nse
      </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tet:PullMessagesResponse>
        <tet:CurrentTime>
          2008-10-10T12:24:58
        </tet:CurrentTime>
        <tet:TerminationTime>
          2008-10-10T12:25:58
        </tet:TerminationTime>
        <wsnt:NotificationMessage>
          <wsnt:Topic
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
              tns1:Device/Trigger/DigitalInput
          </wsnt:Topic>
          <wsnt:Message>
            <tt:Message UtcTime="2008-10-10T12:24:57.321Z"
                        PropertyOperation="Changed">
              <tt:Source>
                <tt:SimpleItem Name="InputToken" Value="1"/>
              </tt:Source>
```

```
              <tt:Data>
                <tt:SimpleItem Name="LogicalState" Value="true"/>
                </tt:Data>
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
      </tet:PullMessagesResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.8 UnsubscribeRequest

Un client doit mettre fin explicitement à un abonnement avec une UnsubscribeRequest afin que le dispositif puisse libérer immédiatement des ressources. La demande s'adresse au point terminal d'abonnement retourné dans la CreatePullPointSubscriptionResponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2" >
     <SOAP-ENV:Header>
          <wsa:Action>
  http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/UnsubscribeRequest                </wsa:Action>
             <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
          <wsnt:Unsubscribe/>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.11.9 UnsubscribeResponse

Le point terminal d'abonnement n'est plus disponible dès lors que le dispositif renvoie une UnsubscribeResponse.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2" >
     <SOAP-ENV:Header>
          <wsa:Action>
http://docs.oasis-open.org/wsn/bw-
2/SubscriptionManager/UnsubscribeResponse               </wsa:Action>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
          <wsnt:UnsubscribeResponse/>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.12 Événement de stockage continu:BeginingOfBuffer

Le début d'un événement de mémoire tampon est un événement logique lié à chaque abonnement et signalant qu'un abonnement est en cours de lecture après le début de la mémoire tampon dans n'importe quelle direction.

Si un dispositif prend en charge la notification de stockage continu, il doit prendre en charge le début de l'événement de mémoire tampon.

Un dispositif doit signaler le début de l'événement de mémoire tampon lorsqu'un abonnement est en cours de lecture, c'est-à-dire les PullMessages, après le début de la mémoire tampon de stockage continu en marche avant ou arrière.

De plus, lorsqu'une opération Seek a été effectuée avant le début de la mémoire tampon, un dispositif doit, quel que soit le sens de lecture, revenir au début de l'événement de mémoire tampon.

Un dispositif doit, pour chaque opération Seek effectuée sur un abonnement, envoyer le début de l'événement de mémoire tampon au maximum une fois.

```
Topic: tns1:EventBuffer/Begin
<tt:MessageDescription IsProperty=”false”/>
```

10.13 Codes de défaut spécifiques au service

Le service d'événement ne définit pas de défauts spécifiques au service autres que ceux définis dans la spécification [OASIS WS-BaseNotification].

11 Sécurité

11.1 Généralités

Comme pour toutes les technologies de l'information orientées réseau, la sécurité est un aspect primordial de la communication dans un système d'alarme et de sécurité électronique. La menace de sécurité dépend de l'application. Alors que certaines applications sont particulièrement vulnérables aux attaques depuis le réseau, d'autres y sont totalement insensibles. Le coût de la mise en œuvre de contre-mesures de sécurité varie suivant le type d'attaque à prévenir. Cela implique qu'il est impossible d'énumérer les exigences de sécurité générales applicables aux systèmes d'alarme et de sécurité électroniques et à leurs composants, mais que des efforts peuvent être faits pour tenter d'identifier des exigences de niveau de sécurité raisonnable pour des dispositifs conformes au présent document et pour définir un mécanisme de sécurité de base permettant de construire un système d'alarme et de sécurité électronique fiable.

Le présent document définit des mécanismes de sécurité au niveau transport.

Le présent document adopte le mécanisme d'authentification basé sur le port, comme suit.

- IEEE 802.1X

11.2 Sécurité au niveau transport

11.2.1 Généralités

La sécurité au niveau transport protège le transfert de données entre le client et le serveur. TLS (Transport Layer Security) est considéré comme une norme mature pour des connexions de transport chiffrées permettant d'obtenir un niveau de sécurité de communication de base. Le protocole TLS permet de configurer une session de transport mutuellement authentifiée et de protéger la confidentialité et l'intégrité des données transportées.

Il convient qu'un dispositif conforme au présent document prenne en charge TLS 1.0 [RFC 2246] et les spécifications connexes. Il convient que le dispositif prenne en charge TLS 1.1 [RFC 4346]. Le dispositif peut prendre en charge TLS 1.2 [RFC 5246].

Il convient qu'un dispositif prenne en charge TLS pour la protection de tous les services qu'il fournit. Il convient qu'un dispositif prenne également en charge TLS pour la protection des flux multimédias de l'option de tunnel RTP/RTSP/HTTPS tel que défini à l'Article 11. Le présent document décrit une mise en œuvre particulière de TLS et d'autres spécifications pertinentes qui peuvent être utilisées avec TLS.

Il convient qu'un client prenne en charge TLS 1.0 [RFC 2246] et TLS 1.1 [RFC 4346]. Le client peut prendre en charge TLS 1.2 [RFC 5246].

11.2.2 Suites de chiffrement prises en charge

Un dispositif qui prend en charge TLS doit prendre en charge toutes les suites de chiffrement suivantes [RFC 2246], [RFC 3268]:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

Si un client prend en charge TLS, il doit prendre en charge les suites de chiffrement suivantes:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

11.2.3 Authentification de serveur

Un dispositif qui prend en charge TLS doit prendre en charge l'authentification de serveur à l'aide de TLS. Le dispositif doit prendre en charge le traitement de certificats de serveur X.509. La longueur de clé RSA doit être d'au moins 1 024 bits.

Il convient qu'un client prenne en charge l'authentification de serveur à l'aide de TLS.

Le présent document ne fournit pas un modèle complet de génération de certificat de serveur et d'autorité de certification (CA). Toutefois, les commandes de gestion de dispositif pour l'extraction et le téléchargement de certificat de dispositifs sont définies en 8.5.

Les détails des mécanismes d'amorçage sécurisés de clés ou clés privées de serveur ne relèvent pas du domaine d'application du présent document. Toutefois, des commandes de génération de clé intégrée sont définies en 8.5.

11.2.4 Authentification de client

Il convient qu'un dispositif qui prend en charge TLS prenne en charge l'authentification de client. L'authentification de client peut être activée/désactivée avec une commande de gestion de dispositif comme décrit en 8.5.

Un dispositif qui prend en charge TLS doit inclure le type de certificat RSA (rsa_sign, par exemple) dans la demande de certificat [RFC 2246] pour des certificats de client, et doit prendre en charge la vérification du certificat et de la signature de client RSA.

Il convient qu'un client prenne en charge l'authentification de client. Si l'authentification de client est prise en charge, le client doit prendre en charge le certificat et la signature de client RSA et doit utiliser une longueur de clé RSA d'au moins 1 024 bits.

Les mécanismes d'amorçage CA de confiance sont hors du domaine d'application du présent document. Les versions ultérieures de la norme peuvent définir des mécanismes d'amorçage normalisés.

11.3 IEEE 802.1X

IEEE 802.1X est une norme IEEE de contrôle d'accès au réseau basée sur le port permettant d'authentifier et d'autoriser des dispositifs associés à des ports LAN. Elle utilise les caractéristiques d'accès physiques des infrastructures LAN IEEE 802 pour offrir un moyen d'authentification et d'autorisation des dispositifs associés à un port LAN présentant des caractéristiques de connexion point à point, ainsi qu'un moyen d'interdiction d'accès à ce port en cas d'échec du processus d'authentification et d'autorisation.

Le présent document recommande l'adoption d'IEEE 802.1X pour l'authentification basée sur le port des réseaux sans fil. Un dispositif qui prend en charge IEEE 802.1X doit prendre en

charge le type EAP-PEAP/MSCHAPv2 comme étant une méthode EAP prise en charge. Le dispositif peut également prendre en charge d'autres méthodes EAP telles que les types EAP-MD5, EAP-TLS et EAP-TTLS.

Le présent document définit un ensemble de commandes de configuration et de gestion de la configuration IEEE 802.1X, se référer à 8.5.8.

Annexe A
(informative)

Exemple de réponse GetServices avec fonctionnalités

L'élément suivant est un exemple de réponse GetServices.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Header>
  </env:Header>
  <env:Body>
    <tds:GetServicesResponse>
      <tds:Service>
        <tds:Namespace>http://www.onvif.org/ver10/device/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif/device_service</tds:XAddr>
        <tds:Capabilities>
          <tds:Capabilities>
            <tds:Network IPFilter="false" ZeroConfiguration="true"
IPVersion6="false" DynDNS="false" Dot11Configuration="false"
HostnameFromDHCP="false" NTP="0" />
            <tds:Security TLS1.0="false" TLS1.1="false" TLS1.2="false"
OnboardKeyGeneration="false" AccessPolicyConfig="false"
DefaultAccessPolicy="false" Dot1X="false" RemoteUserHandling="false"
X.509Token="false" SAMLToken="false" KerberosToken="false"
UsernameToken="false" HttpDigest="false" RELToken="false" />
            <tds:System DiscoveryResolve="true" DiscoveryBye="true"
SystemBackup="false" SystemLogging="false" FirmwareUpgrade="true"
HttpFirmwareUpgrade="false" HttpSystemBackup="false" HttpSystemLogging="false"
HttpSupportInformation="false" />
            <tds:MiscCapabilities AuxiliaryCommands="" />
          </tds:Capabilities>
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
      <tds:Service>
        <tds:Namespace>http://www.onvif.org/ver10/events/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <tev:Capabilities xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
WSSubscriptionPolicySupport="false" WSPullPointSupport="false"
WSPausableSubscription="false" />
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
<tds:Namespace>http://www.onvif.org/ver10/deviceIO/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <tmd:Capabilities
xmlns:tmd="http://www.onvif.org/ver10/deviceIO/wsdl" RelayOutputs="2"
SerialPorts="0" DigitalInputs="4" />
        </tds:Capabilities>
        <tds:Version>
          <tt:Major>2</tt:Major>
          <tt:Minor>20</tt:Minor>
        </tds:Version>
      </tds:Service>
```

```
    </tds:GetServicesResponse>
  </env:Body>
</env:Envelope>
```

À noter que les fonctionnalités peuvent être omises si un dispositif ne prend pas en charge la fonctionnalité ou si une nouvelle fonctionnalité est définie après la mise en œuvre du dispositif.

Annexe B
(normative)

Schéma XML d'interface réseau IP de dispositif

B.1 WSDL de service de gestion de dispositif

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
targetNamespace="http://www.onvif.org/ver10/device/wsdl">
  <wsdl:types>
    <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/device/wsdl" elementFormDefault="qualified"
version="2.4.1">
      <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../../../ver10/schema/onvif.xsd"/>
      <!--================================-->
      <xs:element name="GetServices">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="IncludeCapability" type="xs:boolean"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServicesResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Service" type="tds:Service" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!--================================-->
      <xs:complexType name="Service">
        <xs:sequence>
          <xs:element name="Namespace" type="xs:anyURI"/>
          <xs:element name="XAddr" type="xs:anyURI"/>
          <xs:element name="Capabilities" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:any namespace="##any" processContents="lax"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Version" type="tt:OnvifVersion"/>
          <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <!--================================-->
      <xs:element name="GetServiceCapabilities">
        <xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
      <xs:element name="GetServiceCapabilitiesResponse">
        <xs:complexType>
```