ONVIF™
ONVIF Core Specification

Version 20.12

December, 2020

CONTENTS

Contributors

Version 1

Christian Gehrmann (Ed.)>	Axis Communications AB		Alexander Neubeck	Bosch Security Systems
Mikael Ranbro	Axis Communications AB		Susanne Kinza	Bosch Security Systems
Johan Nyström	Axis Communications AB		Markus Wierny	Bosch Security Systems
Ulf Olsson	Axis Communications AB		Rainer Bauereiss	Bosch Security Systems
Göran Haraldsson	Axis Communications AB		Masashi Tonomura	Sony Corporation
Daniel Elvin	Axis Communications AB		Norio Ishibashi	Sony Corporation
Hans Olsen	Axis Communications AB		Yoichi Kasahara	Sony Corporation
Martin Rasmusson	Axis Communications AB		Yoshiyuki Kunito	Sony Corporation
Stefan Andersson (co Ed.)	Axis Communications AB			

Version 2

Stefan Andersson	Axis Communications AB		Toshihiro Shimizu	Panasonic
Christian Gehrmann	Axis Communications AB		Manabu Nakamura	Panasonic
Willy Sagefalk	Axis Communications AB		Hasan Timucin Ozdemir	Panasonic
Mikael Ranbro	Axis Communications AB		Hiroaki Ootake	Panasonic
Ted Hartzell	Axis Communications AB		Young Hoon	ITX
Rainer Bauereiss	Bosch Security Systems		Sekrai Hong	Samsung
Hans Busch (Ed.)	Bosch Security Systems		Gero Bäse	Siemens
Susanne Kinza (co Ed.)	Bosch Security Systems		Michio Hirai	Sony Corporation
Dieu Thanh Nguyen	Bosch Security Systems		Akihiro Hokimoto	Sony Corporation
Antonie van Woerdekom	Bosch Security Systems		Masashi Tonomura	Sony Corporation
Shinichi Hatae	Canon Inc		Masashi Tonomura	Sony Corporation
Takahiro Iwasaki	Canon Inc			
Takeshi Asahi	Hitachi Ltd			
Colin Caughie	IndigoVision Ltd			
Heather Logan	IndigoVision Ltd			

INTRODUCTION

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

This is the ONVIF core specification. It is accompanied by a set of computer readable interface definitions:

- ONVIF Schema [ONVIF Schema]
- ONVIF Device Service WSDL [ONVIF DM WSDL]
- ONVIF Event Service WSDL [ONVIF Event WSDL]
- ONVIF Topic Namespace XML [ONVIF Topic Namespace]

The purpose of this document is to define the ONVIF specification framework, and is divided into the following sections:

Specification Overview: Gives an overview of the different specification 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 the ONVIF specifications.

IP Configuration: Defines the ONVIF 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.

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

Security Section: Defines the message level security requirements on ONVIF compliant implementations.

1 Scope

This specification defines procedures for communication between network clients and devices. This new set of specifications makes it possible to build e.g. network video systems with devices and receivers from different manufacturers using common and well defined interfaces. The functions defined in this specification covers discovery, device management and event framework.

Supplementary dedicated services as e.g. media configuration, real-time streaming of audio and video, Pan, Tilt and Zoom (PTZ) control, video analytics as well as control, search and replay of recordings are defined in separate documents.

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

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

2 Normative references

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

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 2616, Hypertext Transfer Protocol -- HTTP/1.1
<http://www.ietf.org/rfc/rfc2616.txt>

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

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

IETF RFC 3548, The Base16, Base32, and Base64 Data Encodings
<http://www.ietf.org/rfc/rfc3548.txt>

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

IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax
<http://www.ietf.org/rfc/rfc3986.txt>

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

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

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

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

W3C SOAP Message Transmission Optimization Mechanism,

<http://www.w3.org/TR/soap12-mtom/>

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

W3C SOAP Version 1.2 Part 2: Adjuncts (Second Edition)
<http://www.w3.org/TR/2007/REC-soap12-part2-20070427/>

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

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

OASIS Web Services Base Notification 1.3
<http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf>

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

OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)
<http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

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

OASIS Web Services Security UsernameToken Profile 1.0
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

W3C Web Services Description Language (WSDL) 1.1
<http://www.w3.org/TR/wsdl>

W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures
<http://www.w3.org/TR/xmlschema11-1/>

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
<http://www.w3.org/TR/xmlschema11-2/>

W3C XML-binary Optimized Packaging
<http://www.w3.org/TR/2005/REC-xop10-20050125/>

W3C XML Path Language (XPath) Version 1.0
<https://www.w3.org/TR/1999/REC-xpath-19991116>

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>

WGS1984, National Geospatial Intelligence Agency: DoD World Geodetic System 1984
< http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html>

3 Terms and Definitions

3.1 Definitions

Ad-hoc network	Often used as a vernacular term for an independent basic service set, as defined in [IEEE 802.11-2007].
Basic Service Set	A set of IEEE802.11 stations that have successfully joined in a common network, see [IEEE 802.11-2007].

Capability	The capability commands allows a client to ask for the services provided by a device.
GPS	Global Positioning System
PullPoint	Resource for pulling messages. By pulling messages, notifications are not blocked by firewalls.
Service Set ID	The identity of an [IEEE 802.11-2007] wireless network.
WGS	World Geodetic System 1984, the coordinate system used by the global positioning system.
Wi-Fi Protected Access	A certification program created by the Wi-Fi Alliance to indicate compliance with the security protocol covered by the program.

3.2 Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN	Abstract Syntax Notation
BSSID	Basic Service Set Identification
CDMI	Cloud Data Management Interface
CCMP	Counter mode with Cipher-block chaining Message authentication code Protocol
DAS	Direct Attached Storage
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DM	Device Management
DNS	Domain Name Server
FIPS	Federal Information Processing Standard
GW	Gateway
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
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
NAS	Network Attached Storage
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
PTZ	Pan/Tilt/Zoom

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
WPA	Wi-Fi Protected Access
WS	Web Services
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
XML	eXtensible Markup Language
XPath	XML Path Language

4 Overview

This specification originated from network video use cases covering both local and wide area network scenarios and has been extended to cover generic IP device use cases. The specification defines a core set of interface functions for configuration and operation of network devices by defining their server side interfaces.

This standard covers device discovery, device configuration as well as an event framework.

All services share a common XML schema and all data types are provided in [ONVIF Schema]. The different services are defined in the respective sections and service WSDL documents.

4.1 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 [Part 1] and WSDL1.1 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 the standard are expressed as Web Services operations and defined in WSDL with HTTP as the underlying transport mechanism.

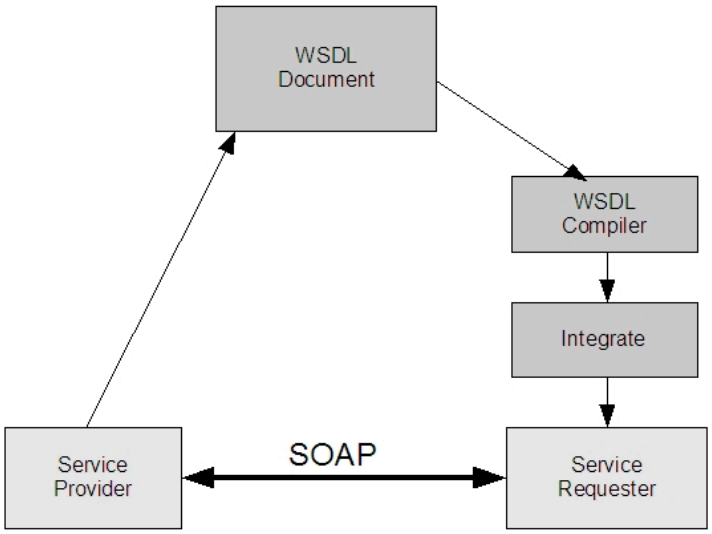

Figure 1: Web Services based development principles

Figure 1 gives an overview of the basic principles for development based on Web Services. The service provider (device) implements the ONVIF 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 platform specific code that can be used by the client side developer to integrate the Web Service into an application.

The Web Service 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 ONVIF standard defines conformant transport protocols for the SOAP messages for the described Web Services.

The Web Service overview section introduces into the general ONVIF service structure, the command definition syntax in the specification, error handling principles and the adopted Web Service security mechanisms.

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

4.2 IP configuration

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

- IP network communication capability
- Static IP configuration
- Dynamic IP configuration

4.3 Device discovery

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

This standard introduces a specific discovery behaviour suitable for e.g. video surveillance purposes. For example, a fully interoperable discovery requires a well defined service definition and a service searching criteria. The specification 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 section 4.5 below.

4.4 Profiles

Device functionality can be grouped to so called profiles. The profiles themselves are defined in separate specifications.

For each profile a number of services and functions are mandatory which are defined in the respective specifications.

4.5 Device management

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 in the Device Management WSDL file. The device management interfaces consist of these subcategories:

- Capabilities
- Network
- System
- Security

4.5.1 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 and the event service. For the other services refer to the respective service specification:

- Device
    - Network
    - System
    - Security
- Event

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

4.5.2 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

4.5.3 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.

4.5.4 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.

4.5.5 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.

4.5.6 System Restore

The System Restore 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.

4.5.7 Security

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

- Get and set access security policy.
- Handle user credentials and settings.

For further security related aspects refer to the ONVIF Advanced Security Service Specification.

4.5.8 Storage configuration

Storage configuration data contains the configuration data related to storage (DAS, NAS, CDMI). For example, CDMI client configuration data contains the server address and user credential information for a CDMI server. An ONVIF Device can connect to CDMI Server via standard CDMI protocol, for example, to store and read device configuration data, archive alarm video, export video, etc. operations.

4.6 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 specification 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 requires standardized notifications. However, the notification topics will, to a large extent, depend on the application needs. This specification defines a set of basicnotification topics.

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

4.7 Geo Location

Interface to describes the location of the device and its entities. A two level approach allows to model both outdoor and indoor situations. See Figure 2 for the orientation of the axis. The position on earth is defined via the angles lon and lat in degrees as well as the hight in meter.The model is coined ENU for (East, North, Up). The mapping of the coordinate system is defined by [WGS1984] and the base for GPS.

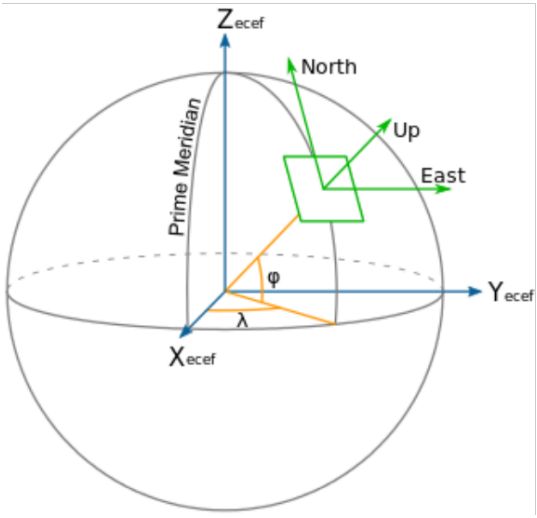

Figure 2: Location on earth[1]

[1]Source: https://en.wikipedia.org/wiki/Axes_conventions.

The range for longitude is between -180 and +180 degrees, while the range for the latitude is between -90 and +90 degrees. The range for elevation is un unbounded signed value, to deal not only with points above the sea level, but also under the world ellipsoid, such as points under water or points in depressions.

Figure 3 describes the three orientation angles called roll, pitch and yaw.

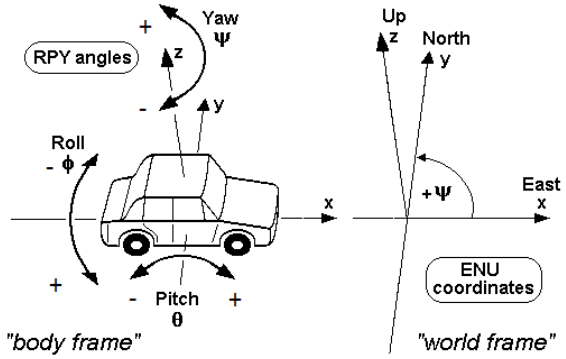

Figure 3: Orientation on earth surface[2]

The second level is defined by a threedimensional rectangular coordinate system. The vector x,y,z describes the offset in meter and the three angles roll (ϕ), pitch (θ) and yaw (ψ) describe an additional orientation.

The range for roll, pitch and yaw is between -180 and +180 degrees.

Installers and implementers can utilize this approach by either defining the six geo parameters for outdoor installations or the six local parameters for indoor systems. But also a combined approach may be applied e.g. on board of ships where the geo component defines the position of the vehicle while the local component defines the offset of the devices inside the vehicle.

It is worth noticing that orientation is only modified by invoking SetGeoLocation. Invoking any other function, such as for example SetHomePosition, will not alter the device orientation values.

5 Web Services framework

All management and configuration commands are based on Web Services.

For the purpose of this standard:

- The device is a service provider.
- The client is a service requester.

A typical ONVIF 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 specification targets devices while the UDDI model is *not* device oriented. Consequently, UDDI and service brokers are *outside the scope* of this specification.

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

[2]Source: By Qniemiec, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10893168

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], except for Requirement R2729 (ONVIF defines some shared response wrapper names) and Requirement R2801 (ONVIF references XML Schema 1.1 rather than XML Schema 1.0).

5.1 Services overview

5.1.1 General

An ONVIF compliant device shall support a number of Web Services which are defined in this 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 ONVIF defined WS-Discovery behaviour, see chapter 7.

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

`http://onvif_host/onvif/device_service`

5.1.2 Services requirements

An ONVIF compliant device shall provide the device management and event service.

If an ONVIF compliant 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 5.8.2 for the definitions of the error codes.

5.2 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” [WSDL1.1].

This specification follows the WSDL 1.1 specification 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.

Note that neither the port and service definitions are used since the ONVIF interface is not bound to a concrete server instance.

Since the release of WSDL 1.1 the underlying XML schema reference has undergone two major revisions. This specification defines that the relaxation of the Unique Particle Attribution rule of XML Schema 1.1 may be used for schema extensibility.

5.3 Namespaces

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

Table 1: Defined namespaces in this specification

Prefix	Namespace URI	Description
tt	http://www.onvif.org/ver10/schema	XML schema descriptions in this specification.
tds	http://www.onvif.org/ver10/device/wsdl	The namespace for the WSDL device 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 for ONVIF defined faults.
dn	http://www.onvif.org/ver10/network/wsdl	The namespace used for the *remote* device discovery service in this specification.
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace

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

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.
soapenc	http://www.w3.org/2003/05/soap-encoding	Encoding namespace as defined by SOAP 1.2 [SOAP 1.2, Part 2]
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XML-Schema, Part 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 [WS-Discovery].
wsadis	http://schemas.xmlsoap.org/ws/2004/08/addressing	Device addressing namespace referred in WS-Discovery [WS-Discovery].
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [WS-Addressing].

Prefix	Namespace URI	Description
wstop	http://docs.oasis-open.org/wsn/t-1	Schema namespace of the [WS-Topics] specification.
wsnt	http://docs.oasis-open.org/wsn/b-2	Schema namespace of the [WS-BaseNotification] specification.
xop	http://www.w3.org/2004/08/xop/include	XML-binary Optimized Packaging namespace as defined by [XOP]

In addition this standard 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	The ONVIF dialect for the topic expressions.
http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter	The ONVIF filter dialect used for message content filtering.

5.4 Types

Data types are defined using XML schema descriptions Part1 and Part 2. Numerous data types defined in this specification are included in [ONVIF Schema] and can be downloaded from:

- http://www.onvif.org/onvif/ver10/schema/onvif.xsd

Services should define locally used data types in the types section of the wsdl file.

5.5 Operations

Operations are defined within the WSDL portType declaration. An operation typically uses the request-response pattern. The service provider receives a request message and sends a corresponding response message.

Both request and response message may have zero or more parameters. Parameters may be marked as optional which maps to an XML minOccurs='0' attribute. Additionally parameters may have multiple occurrence.

If a request message which modifies the state of the device includes the definition of an optional element, and the optional element is not present in the request message, the device should treat the contents of that element as if it contains the current state value. Specific operations may override this treatment of optional elements, in which case the behaviour is specified in the operation's description.

For each command well defined fault codes should be listed. Additionally any command may generate generic faults as defined by 5.8.2.2.

The Access_Class_Name defines the access class of the operation. The access class characterizes the impact of the operation, see 5.9.2.3

5.6 Port Types and Messages

The operations of a service are grouped in one or more port types. A port type is a named set of abstract operations referencing the abstract messages involved.

The message definitions map messages to parameter types.elements to section contains the message content. A message defintion shall have exactly one parameter type reference as defined by WS-I basic profile [WS-I BP 2.0] named “parameters”.

5.7 Binding

A binding defines 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 an ONVIF compliant device according to this specification 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. An ONVIF compliant device shall use the style ‘document’ specified at the operation level.

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

5.8 Error handling

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

The specification classifies error handling into the following categories:

- Protocol Errors
- SOAP Errors
- Application Errors

5.8.1 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 standard, devices and clients shall use appropriate RTSP and HTTP protocol defined status codes for error reporting and when received handle accordingly.

5.8.2 SOAP errors

5.8.2.1 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 SOAP fault message structure with various components in it.

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

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

The ONVIF specifications use a separate name space for specific faults (see 5.8.2.3):

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 as specified in this specification 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 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 table.

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

5.8.2.2 Generic faults

Table 4 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 4: 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 Args	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 recov-

Fault Code	Subcode	Fault Reason	Description
			er by itself and needs reset or power cycle.

5.8.2.3 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.

5.8.2.4 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.

Table 5: 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.

5.9 Security

5.9.1 Authentication

The services defined in this standard shall be protected using digest authentication according to [RFC 2617] with the following exceptions.

- legacy devices supporting [WS-UsernameToken] and
- TLS client authorization.

If server supports both digest authentication as specified in [RFC 2617] and the user name token profile as specified in WS-Security the following behavior shall be adapted: a web service request can be authenticated on the HTTP level via digest authentication [RFC 2617] or on the web service level via the WS-Security (WSS) framework. If a client does not supply authentication credentials along with a web service request, the server shall assume that the client intends to use digest authentication [RFC 2617], if required. Hence, if a client does not provide authentication credentials when requesting a service that requires authentication, it will receive an HTTP 401 error according to [RFC 2617]. Note that this behaviour on the server's side differs from the case of supporting only username token profile, which requires for this case an HTTP 400 error on the HTTP level and a SOAP:Fault env:Sender ter:NotAuthorized error on the WS level.

A client should not simultaneously supply authentication credentials on both the HTTP level and the WS level. If a server receives a web service request that contains authentication credentials on both the HTTP level and the WS level, it shall first validate the credentials provided on the HTTP layer. If this validation was successful, the server shall finally validate the authentication credentials provided on the WS layer.

Figure 4 summarizes the authentication of a web service request by a server.

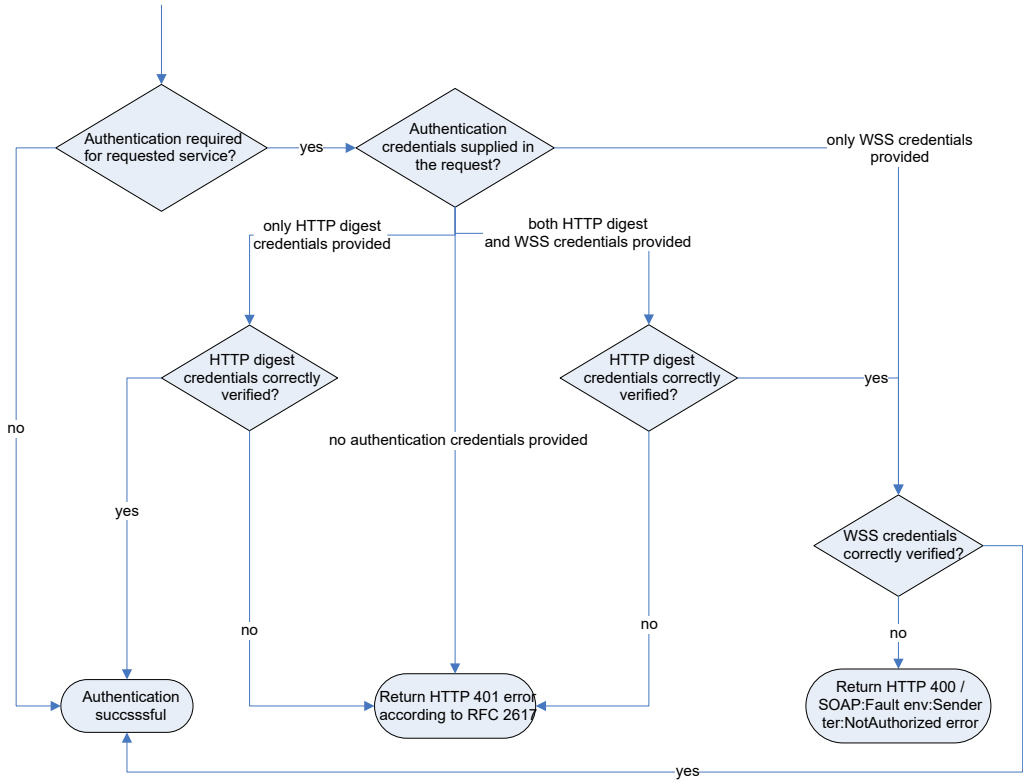

Figure 4: Authentication of a WS request by a server

Both digest authentication and the user name token profile give 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 Advanced Security Service). Digest authentication or the user name token message level security combined with TLS, with client and server authentication, protected transport level security give an acceptable level of security in many systems.

An ONVIF compliant 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.

When authenticating RTSP or HTTP methods, an ONVIF compliant device shall use digest authentication [RFC 2617]. The credentials shall be managed with the GetUsers, CreateUsers, DeleteUsers and SetUser methods. If the device also supports WS-Security, the same set of credentials shall be used.

5.9.2 User-based access control

5.9.2.1 General

The authorization framework described in Sect. 5.12 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.

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 Section 8.4.

5.9.2.2 User Levels

Each user is associated 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.

5.9.2.3 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 6 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 6: Default Access Policy Definition

	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		

5.9.2.4 Default Access Policy

By default, the device should enforce the following default access policy, which gives an acceptable level of security in many systems.

The default access policy builds upon the access classes that are associated to the services and grants access rights in the following way. A user of level c shall be granted access to a service associated to access class r if and only if an "X" is present in the cell at column c and row r in Table 6.

A device that signals support for the Default Access Policy via the respective capability shall support at least one user of each user level Administrator, Operator and User.

5.9.3 Username token profile

A client shall use both nonce and timestamps as defined in [WS-UsernameToken]. The server shall reject any Username Token not using *both* nonce *and creation* timestamps.

This specification 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 5.9.2.2.

5.10 String representation

The following sub-paragraphs are valid for all ONVIF services.

5.10.1 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.

5.10.2 Allowed characters in strings

A device shall not have any restriction regarding legal characters in string that aren't explicitly stated in this and other ONVIF specifications.

6 IP configuration

The device and client communicate over an open or closed IP network. This standard 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 4. Device IP configuration includes parameters such as IP addresses and a default gateway.

An ONVIF compliant 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.

An ONVIF compliant device should support dynamic IP configuration of link-local addresses according to [RFC3927]. A device that supports IPv6 shall support stateless IP configuration according to [RFC4862] and neighbour discovery according to RFC4861.

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 [RFC3315] 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 ONVIF device management service as specified in section 8.2 and may additionally be provided through local interfaces. The latter is outside the scope of this specification.

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 [RFC3927].

Note that the network interface should set up an explicit IPv4 route for multicast traffic to ensure that WS-Discovery is successful, whether a default route is present or not. In a linux environment, this can be done with a command line like:

/sbin/route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0

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

7 Device discovery

7.1 General

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

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

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

[WS-Discovery] describes the Universally Unique Identifier (UUID): URI format recommendation for endpoint references in Section 2.6, but this specification overrides this recommendation. Instead, the Uniform Resource Name: Universally Unique Identifier (URN:UUID) format is used [RFC4122] (see Section 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 [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 [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 8.3.19.

7.3 Discovery definitions

7.3.1 Endpoint reference

A device or an endpoint that takes the client role should use a URN:UUID [RFC4122] 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

An ONVIF compliant 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

An ONVIF compliant 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 specification defines scope attributes as follows:

The scheme attribute:onvif

The authority attribute:www.onvif.org

This implies that all ONVIF defined scope URIs have the following format:

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

A device may have other scope URIs. These URIs are not restricted to ONVIF defined scopes.

Table 7 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 Section 8.3.

Table 7: Scope parameters

Category	Defined values	Description
Profile	Any character string.	Value that indicates the profile supported by the device. The defined values are outside of the scope of this document and are defined in the profile specifications.
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 profile, 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/Profile/Streaming
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 6.

7.3.3 Probe and Probe Match

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

An ONVIF compliant device shall at least support the `http://schemas.xmlsoap.org/ws/2005/04/discovery/rfc3986` scope matching rule. This scope matching definitions differs slightly from the definition in [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 5.1.

7.3.4 Resolve and Resolve Match

This specification 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 [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 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

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 get information about the device capabilities and configurations or to set device configurations. An ONVIF compliant device shall support the device management service as specified in [ONVIF DM WSDL]. 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.

8.1 Capabilities

8.1.1 GetWsdlUrl

This method allows to provide a URL where product specific WSDL and schema definitions can be retrieved. This method has been deprecated with version 20.12.

REQUEST:

This is an empty message.

RESPONSE:

- **WsdlUrl [xs:anyURI]**
  The requested URL.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

8.1.2 Capability exchange

8.1.2.1 General

Any endpoint can ask for the capabilities of a device using the capability exchange request response operation. The capability list includes references to the addresses (XAddr) of the service implementing the interface operations in the category.

8.1.2.2 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 implement this method if any of the ONVIF compliant services implements the GetServiceCapabilities. For making sure about the structure of GetServices response with capabilities, please refer to Annex C.

The version in GetServicesResponse shall contain the specification version number of the corresponding service that is implemented by a device.

For the returned XAddr a device shall match the scheme and IP part of the one used in the GetServices request. Note that if device is behind a NAT that device may return the local adress and not the external address used by the client.

REQUEST:

- **IncludeCapability [boolean]**
  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.

RESPONSE:

- **tds:Service [1][unbounded]**
  The capability response message contains the requested information about the services.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

8.1.2.3 GetServiceCapabilities

This command returns the capabilities of the device service. The service shall implement this method if the device supports the GetServices method.

Table 8 describes how to interpret the indicated capabilities.

REQUEST:

This is an empty message.

RESPONSE:

- **Capabilities [tds:DeviceServiceCapabilities]**
  The capability response message contains the requested device capabilities using a hierarchical XML capability structure.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

Table 8: The capabilities in the GetServiceCapabilities command

Category	Capability	Description
Network	IPFilter	Indication if the device supports IP filtering control using the commands in Section 8.2.18, 8.2.19, 8.2.20 and 8.2.21.
	ZeroConfiguration	Indication if the device supports zero configuration according to the commands in Section 8.2.16 and Section 8.2.17.
	IPVersion6	Indication if the device supports IP version 6.
	DynDNS	Indication if the device supports Dynamic DNS configuration according to Section 8.2.8and Section 8.2.9 .
	Dot11Configuration	Indication if the device supports IEEE802.11 configuration as specified in Section 8.2.22
	HostnameFromDHCP	Indicates whether retrieval of hostname from DHCP is supported by the device.

Category	Capability	Description
	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 (deprecated).
	DHCPv6	Indicates support for Stateful IPv6 DHCP.
System	DiscoveryResolve	Indication if the device responses to resolve requests as described in Section 7.3.4.
	DiscoveryBye	Indication if the device sends bye messages as described in Section 7.3.5
	RemoteDiscovery	Indication if the device supports remote discovery support.
	SystemBackup	Indication if the device supports system backup and restore as specified in Section 8.3.3 and Section 8.3.5
	FirmwareUpgrade	Indication if the device supports firmware upgrade as specified in Section 8.3.9.
	SystemLogging	Indication if the device supports system log retrieval as specified in Section 8.3.11.
	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 section 8.3.2.
	HTTPSupportInformation	Indication if the device supports retrieval of support information using HTTP Get, see section 8.3.2.
	StorageConfiguration	Indication if the device supports storage configuration interfaces as specified in Section 8.7 Storage Configuration.
	GeoLocationEntities	Indicates the number of geo location entities supported. See section 8.3.20 and 8.3.21.
	AutoGeo	Indicates the support for automatic retrieval of geo location. See section 8.3.20 and 8.3.21.

Category	Capability	Description
	StorageTypesSupported	Enumerates the supported StorageTypes.
	NetworkConfigNotSupported	Indicates no support for network configuration.
	UserConfigNotSupported	Indicates no support for user configuration.
Security	AccessPolicyConfig	Indication if the device supports retrieving and loading device access control policy according to Section 8.4.1 and Section 8.4.2.
	DefaultAccessPolicy	Indicates if the device supports the default access policies as defined in 5.9.2.2.
	UsernameToken	Indication if the device supports WS-Security UsernameToken authentication as defined in [WS-UsernameToken].
	HttpDigest	Indication if the device supports the HTTP digest authentication.
	X.509Token	Indication if the device supports the WS-Security X.509 token [WS-X.509Token].
	SAMLToken	Indication if the device supports the WS-Security SAML token [WS-SAMLToken].
	KerberosToken	Indication if the device supports the WS-Security Kerberos token [WS-KerberosToken].
	RELToken	Indication if the device supports the WS-Security REL token [WS-RELToken].
	Dot1X	Indication if the device supports IEEE 802.1X port-based network authentication (deprecated)
	SupportedEAPMethod	List of supported EAP Method types. The numbers correspond to the IANA [EAP-Registry].
	RemoteUserHandling	Indication if device supports remote user handling and the corresponding methods defined in section 8.4.7 and 8.4.8.
	MaxUsers	The maximum number of users that the device supports
	MaxUserNameLength	Maximum number of characters supported for the username by CreateUsers.

Category	Capability	Description
	MaxPasswordLength	Maximum number of characters supported for the password by CreateUsers and SetUser.
Misc	AuxiliaryCommands	List of commands supported by SendAuxiliaryCommand

8.1.2.4 GetCapabilities

This method provides a backward compatible interface for the base capabilities. Refer to GetServices for a full set of capabilities.

Annex A describes how to interpret the indicated capability. Apart from the addresses, the capabilities only reflect optional functions in this specification.

REQUEST:

- **Category - optional, unbounded [tt:CapabilityCategory]**
  This message contains a request for device capabilities. The client can either ask for all capabilities or just the capabilities for a particular service category. If no Category is specified the device SHALL return all capabilities.

RESPONSE:

- **Capabilities [tt:Capabilities]**
  The capability response message contains the requested device capabilities using a hierarchical XML capability structure.

FAULTS:

- **env:Receiver - ter:ActionNotSupported - ter:NoSuchService**
  The requested WSDL service category is not supported by the device.

ACCESS CLASS:

PRE_AUTH

For the list of capabilities refer to Annex A.

8.2 Network

A device shall support the commands defined in this section unless the NetworkConfigNotSupported capability is signalled as 'True' confirming it doesn’t support network configuration.

8.2.1 GetHostname

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.

REQUEST:

This is an empty message.

RESPONSE:

- **FromDHCP [xs:boolean]**
  Signals whether the hostname is obtained via DHCP
- **Name [xs:token]**
  The host name. In case of DHCP the host name has been obtained from the DHCP server.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

8.2.2 SetHostname

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

A device shall accept strings formated according to RFC 1123 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.

REQUEST:

- **Name [xs:token]**
  The host name.If Name is an empty string hostname should be retrieved from DHCP, otherwise the specified Name shall be used.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidHostname**
  The requested hostname cannot be accepted by the device.

ACCESS CLASS:

WRITE_SYSTEM

8.2.3 SetHostnameFromDHCP

This operation 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. A device shall accept the command independent whether it is currently using DHCP to retrieve its IPv4 address or not. Note that the device is not required to retrieve its hostname via DHCP while the device is not using DHCP for retrieving its IP address. In the latter case the device may fall back to the statically set hostname.

REQUEST:

- **FromDHCP [xs:boolean]**
  This message contains: • “FromDHCP”: True if the hostname shall be obtained via DHCP.

RESPONSE:

- **RebootNeeded [xs:boolean]**
  An indication if a reboot is needed in case of changes in the hostname settings.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

WRITE_SYSTEM

8.2.4 GetDNS

This operation gets the DNS settings from a device. The device shall return its DNS configurations through the GetDNS command.

REQUEST:

This is an empty message.

RESPONSE:

- **FromDHCP [xs:boolean]**
  True if the DNS servers are obtained via DHCP.
- **SearchDomain - optional, unbounded [xs:token]**
  The domain(s) to search if the hostname is not fully qualified.
- **DNSFromDHCP - optional, unbounded [tt:IPAddress]**
  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 describes the configuration status.
- **DNSManual - optional, unbounded [tt:IPAddress]**
  A list of manually given DNS servers.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.5 SetDNS

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

It is valid to set the FromDHCP flag while the device is not using DHCP to retrieve its IPv4 address.

REQUEST:

- **FromDHCP [xs:boolean]**
  True if the DNS servers are obtained via DHCP
- **SearchDomain - optional, unbounded [xs:token]**
  The domain(s) to search if the hostname is not fully qualified.
- **DNSManual - optional, unbounded [tt:IPAddress]**
  A list of manually given DNS servers.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidIPv6Address**
  The suggested IPv6 address is invalid.

- **env:Sender - ter:InvalidArgVal - ter:InvalidIPv4Address**
  The suggested IPv4 address is invalid.

ACCESS CLASS:

WRITE_SYSTEM

8.2.6 GetNTP

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

REQUEST:

This is an empty message.

RESPONSE:

- **FromDHCP [xs:boolean]**
  True if the NTP servers are obtained via DHCP.
- **NTPFromDHCP - optional [tt:NetworkHost]**
  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 describes the current configuration status.
- **NTPManual - optional [tt:NetworkHost]**
  A list of manually given NTP servers.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.7 SetNTP

This operation sets the NTP settings on a device. 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 string shall be considered as invalid strings. It is valid to set the FromDHCP flag while the device is not using DHCP to retrieve its IPv4 address.

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

REQUEST:

- **FromDHCP [xs:boolean]**
  True if the NTP servers are obtained via DHCP.
- **NTPManual - optional, unbounded [tt:NetworkHost]**
  A list of manually given NTP servers when they not are obtained via DHCP.

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.8 GetDynamicDNS

This operation gets the dynamic DNS settings from a device. 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.

REQUEST:

This is an empty message.

RESPONSE:

- **Type [tt:DynamicDNSType]**
  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 - optional [tt:DNSName]**
  The DNS name in case of the device does the update.
- **TTL - optional [xs:duration]**
  Time to live.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.9 SetDynamicDNS

This operation sets the dynamic DNS settings on a device. 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.

REQUEST:

- **Type [tt:DynamicDNSType]**
  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 - optional [tt:DNSName] xs:duration TTL [0][1]**
  The DNS name in case of the device does the update.
- **TTL - optional [xs:duration]**
  Time to live.

RESPONSE:

This is an empty message.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

WRITE_SYSTEM

8.2.10 GetNetworkInterfaces

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

REQUEST:

This is an empty message.

RESPONSE:

- **NetworkInterfaces - optional, unbounded [tt:NetworkInterface]**
  This message contains an array of device network interfaces.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.11 SetNetworkInterfaces

This operation sets the network interface configuration on a device. 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 isn't present in the request.

REQUEST:

- **InterfaceToken [tt:ReferenceToken]**
  The token of the network interface to operate on.
- **NetworkInterface [tt:NetworkInterfaceSetConfiguration]**
  The configuration to be applied to the network interface.

RESPONSE:

- **RebootNeeded [xs:boolean]**
  An indication if a reboot is needed in case of changes in the network settings.

FAULTS:

- **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 none 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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.12 GetNetworkProtocols

This operation gets defined network protocols from a device. The device shall support the GetNetworkProtocols command returning configured network protocols.

This message returns an array of defined protocols supported by the device. There are three protocols defined, HTTP, HTTPS and RTSP. For each protocol the parameters Port and Enable/Disable can be retrieved.

REQUEST:

This is an empty message.

RESPONSE:

- **NetworkProtocols - optional, unbounded [tt:NetworkProtocol]**
  Port

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.13 SetNetworkProtocols

This operation configures defined network protocols on a device. The device shall support configuration of defined network protocols through the SetNetworkProtocols command.

This message configures one or more defined network protocols supported by the device. There are currently three protocols defined, HTTP, HTTPS and RTSP. For each protocol the parameters Port and Enable/Disable can be configured.

REQUEST:

- **NetworkProtocols - unbounded [tt:NetworkProtocol]**
  Port

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.14 GetNetworkDefaultGateway

This operation gets the default gateway settings from a device. The device shall support the GetNetworkDefaultGateway command returning manually configured default gateway *address(es)*.

REQUEST:

This is an empty message.

RESPONSE:

- **IPv4Address - optional, unbounded [tt:IPv4Address]**
  The default IPv4 gateway address(es).
- **IPv6Address - optional, unbounded [tt:IPv6Address]**
  The default IPv6 gateway address(es).

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.15 SetNetworkDefaultGateway

This operation sets the default gateway settings on a device. The device shall support configuration of default gateway through the SetNetworkDefaultGateway command.

REQUEST:

- **IPv4Address - optional, unbounded [tt:IPv4Address]**
  The default IPv4 gateway address(es).
- **IPv6Address - optional, unbounded [tt:IPv6Address]**
  The default IPv6 gateway address(es).

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.16 GetZeroConfiguration

This operation gets the zero-configuration from a device. If the device supports dynamic IP configuration according to [RFC3927], it shall support the return of IPv4 zero configuration address and status through the GetZeroConfiguration command

REQUEST:

This is an empty message.

RESPONSE:

- **InterfaceToken [tt:ReferenceToken]**
  The token of the network interface
- **Enabled [xs:boolean]**
  If zero configuration is enabled or not.
- **Address - optional, unbounded [tt:IPv4Addresses]**
  The IPv4 zero configuration address(es).

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.17 SetZeroConfiguration

This operation sets the zero-configuration on the device. 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.

REQUEST:

- **InterfaceToken [tt:ReferenceToken]**
  The token of the network interface to operate on.
- **Enabled [xs:boolean]**
  If zero configuration is enabled or not.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidNetworkInterface**
  The supplied network interface token does not exists

ACCESS CLASS:

WRITE_SYSTEM

8.2.18 GetIPAddressFilter

This operation gets the IP address filter settings from a device. 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.

REQUEST:

This is an empty message.

RESPONSE:

- **Type [tt:IPAddressFilterType]**
  Sets if the filter should deny or allow access.
- **IPv4Address - optional, unbounded [tt:PrefixedIPv4Address]**
  The IPv4 filter address(es)
- **IPv6Address - optional, unbounded [tt:PrefixedIPv6Address]**
  The IPv6 filter address(es)

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.2.19 SetIPAddressFilter

This operation sets the IP address filter settings on a device. 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.

REQUEST:

- **Type [tt:IPAddressFilterType]**
  Sets if the filter should deny or allow access.
- **IPv4Address - optional, unbounded [tt:PrefixedIPv4Address]**
  The IPv4 filter address(es)

- **IPv6Address - optional, unbounded [tt:PrefixedIPv6Address]**
  The IPv6 filter address(es)

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidIPv6Address**
  The suggested IPv6 address is invalid.
- **env:Sender - ter:InvalidArgVal - ter:InvalidIPv4Address**
  The suggested IPv4 address is invalid.

ACCESS CLASS:

WRITE_SYSTEM

8.2.20 AddIPAddressFilter

This operation adds an IP filter address to a device. 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.

REQUEST:

- **Type [tt:IPAddressFilterType]**
  Type": Sets if the filter should deny or allow access.
- **IPv4Address - optional, unbounded [tt:PrefixedIPv4Address]**
  The IPv4 filter address(es)
- **IPv6Address - optional, unbounded [tt:PrefixedIPv6Address]**
  The IPv6 filter address(es)

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.21 RemoveIPAddressFilter

This operation deletes an IP filter address from a device. 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.

REQUEST:

- **Type [tt:IPAddressFilterType]**
  Value of this field is ignored in this command.
- **IPv4Address - optional, unbounded [tt:PrefixedIPv4Address]**
  The IPv4 filter address(es)
- **IPv6Address - optional, unbounded [tt:PrefixedIPv6Address]**
  The IPv6 filter address(es)

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.2.22 IEEE 802.11 configuration

Requirements in this section and subsections are only valid for a device that signals IEEE 802.11 support via its Network Dot11Configuration capability. in this section and subsections 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

8.2.22.1 SSID

The device shall support configuration of the SSID.

8.2.22.2 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 specification; But to allow for devices that support ad-hoc network station modes, the specification allows for selecting (and reporting) this mode.

8.2.22.3 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's 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 specification.

8.2.22.4 Security configuration

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-Enterpise)

Configuration of WEP security mode is outside the scope of this speciation but to allow for devices that support WEP security mode this specification 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.

8.2.22.4.1 None mode

The device shall support the “None” security mode.

8.2.22.4.2 PSK mode

The device shall support the PSK security mode.

To minimize the risk for compromising 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 applies:

- A client shall include a PSK value in the SetNetworkInterfaces request
- The device shall check so 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 applies:

- 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. In ONVIF the security policy shall make sure that the PSK is

8.2.22.5 GetDot11Capabilities

This operation returns the IEEE802.11 capabilities, see Table 9. The device shall support this operation.

REQUEST:

This is an empty message.

RESPONSE:

- **Capabilities [tt:Dot11Capabilities]**

FAULTS:

- **env:Receiver - ter:ActionNotSupported - ter:InvalidDot11**
  IEEE 802.11 configuration is not supported.

ACCESS CLASS:

READ_SYSTEM

Table 9: IEEE802.11 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.

8.2.22.6 GetDot11Status

This operation returns the status of a wireless network interface. The device shall support this command. 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)

REQUEST:

- **InterfaceToken [tt:ReferenceToken]**

RESPONSE:

- **Status [tt:Dot11Status]**

FAULTS:

- **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.

ACCESS CLASS:

READ_SYSTEM

8.2.22.7 ScanAvailableDot11Networks

This operation returns a lists of the wireless networks in range of the device. A device should support this operation. 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)

REQUEST:

- **InterfaceToken [tt:ReferenceToken]**

RESPONSE:

- **Networks - optional, unbounded [tt:Dot11AvailableNetworks]**

FAULTS:

- **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.

ACCESS CLASS:

READ_SYSTEM

8.3 System

8.3.1 GetDeviceInformation

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.

REQUEST:

This is an empty message.

RESPONSE:

- **Manufacturer [xs:string]**
- **Model [xs:string]**
- **FirmwareVersion [xs:string]**
- **SerialNumber [xs:string]**
- **HardwareId [xs:string]**

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.3.2 GetSystemUris

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 specification.

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 specification.

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 specification.

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.

REQUEST:

This is an empty message.

RESPONSE:

- **SystemLogUris - optional [tt:SystemLogUriList]**
  This message contains the URIs from which the various system information components may be downloaded.
- **SupportInfoUri - optional [xs:anyURI]**
- **SystemBackupUri - optional [xs:anyURI]**

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.3.3 GetSystemBackup

This interface has been deprecated. A device shall implement this command if the capability SystemBackup is signaled. For a replacement method see section 8.3.2 and 8.3.5.

This operation retrieves system backup configuration file(s) from a device. The backup is returned with reference to a name and mime-type together with binary data. The format of the backup configuration data is vendor specific. It is expected that after completion of the restore operation the device is working on the same configuration as that of the time the configuration was backed up. Note that the configuration of static IP addresses may differ.

Device vendors may put restrictions on the functionality to be restored. The detailed behavior is outside the scope of this specification.

The backup configuration file(s) are transmitted through MTOM [MTOM].

REQUEST:

This is an empty message.

RESPONSE:

- **BackupFiles - unbounded [tt:BackupFile]**
  The get system backup response message contains the system backup configuration files(s).

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM_SECRET

8.3.4 RestoreSystem

This interface has been deprecated. A device shall implement this command if the capability SystemBackup is signaled. For a replacement method see section 8.3.2 and 8.3.5.

This operation restores the system backup configuration files(s) previously retrieved from a device. The exact format of the backup configuration file(s) is *outside the scope* of this standard. If the command is supported, it shall accept backup files returned by the GetSystemBackup command.

The back up configuration file(s) are transmitted through MTOM [MTOM].

REQUEST:

- **BackupFiles - unbounded [tt:BackupFile]**
  This message contains the system backup file(s).

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidBackupFile**
  The backup file(s) are invalid.

ACCESS CLASS:

UNRECOVERABLE

8.3.5 StartSystemRestore

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 specification.

System restore over HTTP may be achieved 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”.

REQUEST:

This is an empty message.

RESPONSE:

- **UploadUri [xs:anyURI]**
  A URL to which the system configuration file may be uploaded.

- **ExpectedDownTime - optional [xs:duration]**
  An optional duration that indicates how long the device expects to be unavailable after the upload is complete.

FAULTS:

-
  No command-specific faults.

ACCESS CLASS:

UNRECOVERABLE

8.3.6 GetSystemDateAndTime

This operation gets the device system date and time. 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.

REQUEST:

This is an empty message.

RESPONSE:

- **DateTimeType [tt:SetDateTimeType]**
  If the system time and date are set manually or by NTP
- **DayLightSavings [xs:boolean]**
  “DaylightSavings”: Daylight savings on or off
- **TimeZone - optional [tt:TimeZone]**
  The time zone as it is defined in POSIX 1003.1 section 8.3
- **UTCDateTime - optional [tt:DateTime]**
  The time and date in UTC.
- **LocalDateTime - optional [tt:DateTime]**
  The local time and date of the device

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

8.3.7 SetSystemDateAndTime

This operation sets the device system date and time. 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] section 8.3 as invalid.

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.

REQUEST:

- **DateTimeType [tt:SetDateTimeType]**
  If the system time and date are set manually or by NTP

- **DayLightSavings [xs:boolean]**
  "DaylightSavings": Automatically adjust Daylight savings if defined in TimeZone.
- **TimeZone - optional [tt:TimeZone]**
  The time zone is defined in POSIX 1003.1 section 8.3
- **UTCDateTime - optional [tt:DateTime]**
  The time and date in UTC. If DateTimeType is NTP, UTCDateTime has no meaning.

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.3.8 SetSystemFactoryDefault

This operation reloads parameters of a device to their factory default values. The device shall support hard and soft factory default through the SetSystemFactoryDefault command.

Hard All parameters are set to their factory default value.

Soft 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.

REQUEST:

- **FactoryDefault [tt:FactoryDefaultType]**
  Mode Hard or Soft.

RESPONSE:

This is an empty message.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

UNRECOVERABLE

8.3.9 UpgradeSystemFirmware

This interface has been deprecated. A device shall implement this command if the capability FirmwareUpgrade is signaled. For a replacement method see the next section.

This operation upgrades a device firmware version. After a successful upgrade the response message is sent before the device reboots. The exact format of the firmware data is *outside the scope* of this standard.

After applying a firmware upgrade the device shall keep the basic network configuration like IP address, subnet mask and gateway or DHCP settings unchanged. Additionally a firmware upgrade shall not change user credentials.

The firmware is transmitted through MTOM [MTOM].

REQUEST:

- **Firmware [tt:AttachmentData]**
  This message contains the firmware used for the upgrade. The firmware upgrade is “soft” meaning that all parameters keep their current value.

RESPONSE:

- **Message [Xs:string]**
  This message contains a “Message” string allowing the device to report back a message to the client as for an example “Upgrade successful, rebooting in x seconds.”

FAULTS:

- **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.

ACCESS CLASS:

UNRECOVERABLE

8.3.10 StartFirmwareUpgrade

This operation initiates a firmware upgrade using the HTTP POST mechanism. 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 specification.

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”.

After applying a firmware upgrade the device shall keep the basic network configuration like IP address, subnet mask and gateway or DHCP settings unchanged. Additionally a firmware upgrade shall not change user credentials.

REQUEST:

This is an empty message.

RESPONSE:

- **UploadUri [xs:anyURI]**
  A URL to which the firmware file may be uploaded.
- **UploadDelay [xs:duration]**
  An optional delay; the client shall wait for this amount of time before initiating the firmware upload.
- **ExpectedDownTime [xs:duration]**
  A duration that indicates how long the device expects to be unavailable after the firmware upload is complete.

FAULTS:

-
  No command-specific faults.

ACCESS CLASS:

UNRECOVERABLE

In case it is not possible to provide exact figures for either UploadDelay or ExpectedDownTime, the device shall provide best-effort estimates.

8.3.11 GetSystemLog

This operation gets a system log from a device. 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 standard.

The system log information is transmitted through MTOM [MTOM] or as a string.

REQUEST:

- **LogType [tt:SystemLogType]**
  System (the system log) or Access (the client access log)

RESPONSE:

- **Binary - optional [tt:AttachmentData]**
  Binary encoded response.
- **String - optional [xs:string]**
  UTF-8 encoded information.

FAULTS:

- **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

ACCESS CLASS:

READ_SYSTEM_SECRET

8.3.12 GetSystemSupportInformation

This operation gets arbitrary device diagnostics information from a device. 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 standard.

The diagnostics information is transmitted as an attachment through MTOM [MTOM] or as string.

REQUEST:

This is an empty message.

RESPONSE:

- **BinaryFormat - optional [tt:AttachmentData]**
  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.
- **StringFormat - optional [xs:string]**

FAULTS:

- **env:Sender - ter:InvalidArgs - ter:SupportInformation- - Unavailable**
  There is no support information available.

ACCESS CLASS:

READ_SYSTEM_SECRET

8.3.13 SystemReboot

This operation reboots a device. Before the device reboots the response message shall be sent. The device shall support reboot through the SystemReboot command.

REQUEST:

This is an empty message.

RESPONSE:

- **Message xs:string**

  This message contains a “Message” string allowing the device to report back a message to the client as for an example “Rebooting in x seconds.”

FAULTS:

No command specific faults!

ACCESS CLASS:

UNRECOVERABLE

8.3.14 GetScopes

This operation *requests* the scope parameters of a device. The scope parameters are used in the device discovery to match a probe message, see Section 7. The Scope parameters are of two different types:

- Fixed
- Configurable

Fixed scope parameters are permanent device characteristics and cannot be removed through the device management interface. The scope type is indicated in the scope list returned in the get scope parameters response. A device shall support retrieval of discovery scope parameters through the GetScopes command. As some scope parameters are mandatory, the device shall return a non-empty scope list in the response.

REQUEST:

This is an empty message.

RESPONSE:

- **Scopes - unbounded [tt:Scope]**
  The scope response message contains a list of URIs defining the device scopes. See also Section for the ONVIF scope definitions.

FAULTS:

- **env:Receiver - ter:Action - ter:EmptyScope**
  Scope list is empty.

ACCESS CLASS:

READ_SYSTEM

8.3.15 SetScopes

This operation *sets* the scope parameters of a device. The scope parameters are used in the device discovery to match a probe message, see Section 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.

REQUEST:

- **Scopes - unbounded [Xs:anyURI]**
  The set scope contains a list of URIs defining the device scope. See also Section .

RESPONSE:

This is an empty message.

FAULTS:

- **env:Receiver - ter:Action - ter:TooManyScopes**
  The requested scope list exceeds the supported number of scopes.

ACCESS CLASS:

WRITE_SYSTEM

8.3.16 AddScopes

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

REQUEST:

- **xs:anyURI:ScopeItem [1][unbounded]**
  The add scope contains a list of URIs to be added to the existing configurable scope list. See also Section ..

RESPONSE:

This is an empty message.

FAULTS:

- **env:Receiver - ter:Action - ter:TooManyScopes**
  The requested scope list exceeds the supported number of scopes.

ACCESS CLASS:

WRITE_SYSTEM

8.3.17 RemoveScopes

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

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

REQUEST:

- **ScopeItem - unbounded [xs:anyURI]**
  The remove scope contains a list of URIs that should be removed from the device scope.

RESPONSE:

- **ScopeItem - optional, unbounded [xs:anyURI]**
  The scope response message contains a list of URIs that has been Removed from the device scope.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.3.18 GetDiscoveryMode

This operation gets the discovery mode of a device. See Section 7.2 for the definition of the different device discovery modes. The device shall support retrieval of the discovery mode setting through the GetDiscovery-Mode command.

REQUEST:

This is an empty message.

RESPONSE:

- **DiscoveryMode [tt:DiscoveryMode]**
  This message contains the current discovery mode setting, i.e. discoverable or non-discoverable.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.3.19 SetDiscoveryMode

This operation sets the discovery mode operation of a device. See Section 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.

REQUEST:

- **DiscoveryMode [tt:DiscoveryMode]**
  This message contains the requested discovery mode setting, i.e. discoverable or non-discoverable.

RESPONSE:

This is an empty message.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

WRITE_SYSTEM

8.3.20 GetGeoLocation

This operation gets the geo location information of a device. A device that signals support for GeoLocation via the capability GeoLocationEntities shall support the retrieval of geo location information via this command.

The command shall return all location information. Each location entity consists of the following set of optional fields:

GeoPosition	lon,lat according to [WGS1984] and altitude in meters.
GeoOrientation	roll, pitch and yaw angles in degree.
LocalOffset	indoor position in meters
LocalOrientation	indoor orientation agles for pan, tilt and roll in degree.
Entity	Attribute specifying whether the values above define the location of the device, a VideoSource or other entity.
Token	Optional attribute referencing the individual entity.
Fixed	Attribute signaling that the entity cannot be deleted.
GeoSource	Optional external reference that provides the geo location.
AutoGeo	The Geo location information is retrieved internally e.g. by a GPS receiver.

REQUEST:

This is an empty message.

RESPONSE:

- **Location - optional, unbounded [tt:LocationEntity]**
  This message contains all geo location information stored in the device.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM

8.3.21 SetGeoLocation

This operation allows to modify one or more geo location entries. A device that signals support for GeoLocation via the GeoLocationEntities capabiliy shall support modifying geo location information via this command.

The method allows to update one or more entries at once. The method shall modify only those entries that are referenced by the request arguments. A device shall create a new entry in case the combination of type and token does not yet exist. A device shall remove any of the location and orientations components in case they are not passed in the request.

REQUEST:

- **Location - unbounded [tt:LocationEntity]**
  This message contains one or more geo location entries to be stored.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Receiver - ter:Action - ter:TooManyEntries**
  The requested geo location list exceeds the supported number of entries.
- **env:Sender - ter:InvalidArgVal - ter:NoAutoGeo**
  The device does not support automatic retrieval of geo information.

ACCESS CLASS:

WRITE_SYSTEM

If the AutoGeo attribute is set to true and the device is signaling support for **Location** in the AutoGeo capability then the device shall ignore the GeoLocation information provided in the request. If the AutoGeo attribute is set to true and the device is signaling support for **Heading** in the AutoGeo capability then the device shall ignore the GeoOrientation.yaw information provided in the request. If the AutoGeo attribute is set to true and the device is signaling support for **Leveling** in the AutoGeo capability then the device shall ignore the GeoOrientation.roll and GeoOrientation.pitch information provided in the request. Beyond this a device shall return in a subsequent GetGeoLocation command all entity elements as passed to SetGeoLocation.

A device signaling support AutoGeo via the AutoGeo capability shall support the attribute.

8.3.22 DeleteGeoLocation

This operation allows to remove one or more geo location entries. A device that signals support for GeoLocation via its capabilities shall support the remove of geo location information via this command.

A device shall delete an enitity based on the passed fields type and token.

REQUEST:

- **Entity - unbounded [tt:LocationEntity]**
  This message contains one or more geo location entries to be removed.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested entry does not exist.
- **env:Sender - ter:InvalidArgVal - ter:Fixed**
  Cannot delete a fixed entity.

ACCESS CLASS:

WRITE_SYSTEM

8.4 Security

This section 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.

8.4.1 Get access policy

Access to different services and sub-sets of services should be subject to access control. Section 5.9 gives the prerequisite for end-point authentication. Authorization decisions can then be taken using an *access security policy.* This standard does not mandate any particular policy description format or security policy but this is up to the device manufacturer or system provider to choose policy and policy description format of choice. 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.

REQUEST:

This is an empty message.

RESPONSE:

- **PolicyFile [tt:BinaryData]**
  This message contains the requested policy file.

FAULTS:

- **env:Receiver - ter:Action - ter:EmptyPolicy**
  The device policy file does not exist or it is empty.

ACCESS CLASS:

READ_SYSTEM_SECRET

8.4.2 Set access policy

This command setsthe device access security policy (for more details on the access security policy see the Get command, Section 8.4.1). If the device supports access policy settings based on WS-Security authentication, then the device shall support this command.

REQUEST:

- **PolicyFile [tt:BinaryData]**
  This message contains the policy file to set.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgs - ter:PolicyFormat**
  The requested policy cannot be set due to unknown policy format.

ACCESS CLASS:

WRITE_SYSTEM

8.4.3 Get users

This operation lists the registered users and along with their user levels. A device shall support this command unless support signalled via the UserConfigNotSupported capability is ‘True’. The device shall support retrieval of registered device users through the GetUsers command.

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

REQUEST:

This is an empty message.

RESPONSE:

- **User - optional, unbounded [tt:User]**
  List of users and corresponding credentials. Each entry includes user name and level. Note, that the password shall not be included.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_SYSTEM_SECRET

8.4.4 Create users

This operation creates new device users and corresponding credentials on a device for authentication, see Section 5.9 for details. A device shall support this command unless support signalled via the UserConfig-NotSupported capability is 'True'. The device shall support creation of device users and their credentials for authentication through the CreateUsers command 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.

ONVIF compliant devices are recommended to support password length of at least 28 bytes.

REQUEST:

- **User - unbounded [tt:User]**
  User information for users to be created including name, level and password.

RESPONSE:

This is an empty message.

FAULTS:

- **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

ACCESS CLASS:

WRITE_SYSTEM

8.4.5 Delete users

This operation deletes users on a device. A device shall support this command unless support signalled via the UserConfigNotSupported capability is ‘True’. 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 be deleted.

REQUEST:

- **Username - unbounded [xs:string]**
  This message contains the name of the user or users to be deleted.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:UsernameMissing**
  Username not recognized.
- **env:Sender - ter:InvalidArgVal - ter:FixedUser**
  Username may not be deleted

ACCESS CLASS:

WRITE_SYSTEM

8.4.6 Set users settings

This operation updates the settings for one or several users on a device for authentication, see Sect. 5.9 for details. The device shall support update of device users and their credentials through the SetUser command. A device shall support this command unless support signalled via the UserConfigNotSupported capability is ‘True’. Either all change requests are processed successfully or a fault message shall be returned and no change requests be processed.

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

REQUEST:

- **User - unbounded [tt:User]**
  User information for users to be updated including name, level and password.

RESPONSE:

This is an empty message.

FAULTS:

- **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.

ACCESS CLASS:

WRITE_SYSTEM

8.4.7 GetRemoteUser

This operation returns the configured remote user (if any). A device that signals support for remote user handling via the Security Capability RemoteUserHandling shall support this operation. The user is only valid for the WS-UserToken profile or as a HTTP / RTSP user.

Password derivation is outside of the scope of this specification.

REQUEST:

This is an empty message.

RESPONSE:

- **RemoteUser - optional [tt:RemoteUser]**
  Optional information regarding remote user.

FAULTS:

- **env:Receiver - ter:ActionNotSupported - ter:NotRemoteUser**
  Remote User handling is not supported

ACCESS CLASS:

READ_SYSTEM

8.4.8 SetRemoteUser

This operation sets the remote user. A device that signals support for remote user handling via the Security Capability RemoteUserHandling shall support this operation. Password derivation is outside of the scope of this specification.

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

REQUEST:

- **RemoteUser - optional [tt:RemoteUser]**
  Name, password information.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Receiver - ter:ActionNotSupported - ter:NotRemoteUser**
  Remote User handling not supported

ACCESS CLASS:

WRITE_SYSTEM

8.4.9 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 returning the address property of the device service endpoint reference.

REQUEST:

This is an empty message.

RESPONSE:

- **GUID [xs:string]**
  The requested URL.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

8.5 Input/Output (I/O)

The commands in ths section are kept for backward compatibility purposes. For a more extensive IO interface please refer to the ONVIF Device IO Specification.

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.

8.5.1 GetRelayOutputs

This operation gets a list of all available relay outputs and their settings.

REQUEST:

This is an empty message.

RESPONSE:

- **RelayOutputs - optional, unbounded [tt:RelayOutput]**

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_MEDIA

8.5.2 SetRelayOutputSettings

This operation sets the settings of a relay output.

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 Section 8.5.3) 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 Section 8.5.3) 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.

REQUEST:

- **RelayOutputToken [tt:ReferenceToken]**
  Token reference to the requested relay output.
- **RelayOutputSettings [tt:RelayOutputSettings]**
  The settings of the relay.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:RelayToken**
  Unknown relay token reference.
- **env:Sender - ter:InvalidArgVal - ter:ModeError**
  Monostable delay time not valid

ACCESS CLASS:

ACTUATE

8.5.3 SetRelayOutputState

This operation triggers a relay output[1].

REQUEST:

- **RelayOutputToken [tt:ReferenceToken]**
  Token reference to the requested relay output.
- **LogicalState [RelayLogicalState]**
  Trigger request, i.e., active or inactive.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:RelayToken**
  Unknown relay token reference.

ACCESS CLASS:

ACTUATE

8.6 Auxiliary operation

This section describes operations to manage auxiliary commands supported by a device, 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 is reported in the AuxiliaryCommands attribute returned by the capabilites commands, see section 8.1.2. The command transmitted by using this command should match one of the commands supported by the device. If for example the capability command response lists only irlampon

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

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.
- tt:IRLamp|On – Request to turn ON an IR illuminator attached to the unit.
- tt:IRLamp|Off – Request to turn OFF an IR illuminator attached to the unit.
- tt:IRLamp|Auto – Request to configure an IR illuminator attached to the unit so that it automatically turns ON and OFF.

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

REQUEST:

- **AuxiliaryCommand [tt:AuxiliaryData]**
  This message contains the auxiliary command.

RESPONSE:

- **AuxiliaryCommandResponse - optional [tt:AuxiliaryData]**
  The response contains the auxiliary response.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:AuxiliaryDataNotSupported**
  The requested AuxiliaryCommand is not supported.

ACCESS CLASS:

ACTUATE

8.7 Storage Configuration

The following operations allow client to configure the storage configuration data of device. The storage configuration can refer to DAS, NAS, and CDMI Server.

8.7.1 GetStorageConfigurations

This operation lists all existing storage configurations. A device indicating storage configuration capability shall support the listing of existing storage configurations through the GetStorageConfigurations command.

REQUEST:

This is an empty message.

RESPONSE:

- **StorageConfigurations - optional, unbounded [tt:StorageConfiguration]**
  This message contains a list of existing storage configurations. If a device has no storage configuration, then the message shall return an empty list.

FAULTS:

- **No specific fault codes.**

ACCESS CLASS:

READ_MEDIA

8.7.2 CreateStorageConfiguration

This operation creates a new storage configuration. The configuration data shall be created in the device and shall be persistent (remains after a device reboots). A device indicating storage configuration capability shall support the creation of storage configurations as long as the number of existing storage configurations does not exceed the value of MaxStorageConfigurations capability.

REQUEST:

- **StorageConfigurationData [tt:StorageConfigurationData]**
  The request message specifies which configuration data shall be created

RESPONSE:

- **StorageConfigurationToken [tt:ReferenceToken]**
  This message contains unique token for the newly created Storage Configuration data

FAULTS:

- **env:Receiver - ter:OperationProhibited - ter:MaxStorageConfigurations**
  The maximum number of supported storage configurations has been reached.

ACCESS CLASS:

ACTUATE

8.7.3 GetStorageConfiguration

This operation retrieves the Storage configuration when the storage configuration token is known. A device indicating storage configuration capability shall support retrieval of specific storage configuration through the GetStorageConfiguration command.

REQUEST:

- **StorageConfigurationToken [tt:ReferenceToken]**
  This message contains the token of the requested storage configuration.

RESPONSE:

- **StorageConfiguration [tt:StorageConfiguration]**
  The message contains the requested storage configuration matching with the given token.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested storage configuration does not exist.

ACCESS CLASS:

READ_MEDIA

8.7.4 SetStorageConfiguration

This operation modifies an existing storage configuration. A device indicating storage configuration capability shall support the modification of storage configuration through the SetStorageConfiguration command.

REQUEST:

- **StorageConfiguration [tt:StorageConfiguration]**
  This message contains the modified storage configuration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested storage configuration does not exist.
- **env:Sender - ter:InvalidArgVal - ter:ConfigModify**
  The configuration parameters are not possible to set.

ACCESS CLASS:

ACTUATE

8.7.5 DeleteStorageConfiguration

This operation deletes a storage configuration. This change shall always be persistent. A device indicating storage configuration capability shall support the deletion of a storage configuration through the DeleteStorageConfiguration command.

REQUEST:

- **StorageConfigurationToken [tt:ReferenceToken]**
  This message contains an storage configuration token that indicates which storage configuration shall be deleted

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoConfig**
  The requested storage configuration does not exist.

ACCESS CLASS:

ACTUATE

8.8 MonitoringEvents

8.8.1 Processor Usage

If a device supports monitoring of processing unit usage, it should provide the processing unit usage monitoring event to inform a client about its current processing unit usage in percent. The value shall be the usage average over a time interval. It is recommended to use a time interval of five seconds to avoid flooding the event queue with excessive processing unit usage events.

```
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>
```

8.8.2 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>
```

8.8.3 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 Upload Status event

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

8.8.4 Operating Time

The set of events defined in this section relates to operating time. A device supporting operation time events should provide the following events. A device shall report times specified in the following events as UTC using the 'Z' indicator.

The following event should be generated with 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 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 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 was last booted.

```
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 the 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 true value when the area of operation the device is certified for is not adhered to caused by 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 true value when the area of operation the device is configured for is not adhered to caused by outside influences.

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

A device shall report the time of 1970-01-01T00:00:00Z when notifying the Initialized state of a property event that has never occurred. This applies e.g. to LastReset, LastClockSynchronization and Backup/Last.

8.8.5 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. It is recommended to use a time interval of sixty seconds or a 1% change to avoid flooding the event queue with excessive relative humidity events. An event shall be sent if either the interval or percent change occurs.

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. It is recommended to use a time interval of sixty seconds or a 5% change to avoid flooding the event queue with excessive temperature change events. An event shall be sent if either the interval or percent change occurs.

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

8.8.6 Battery capacity

If measurements of the battery level are supported a device should provide the data using the BatteryCapacity event. It is recommended to use a 2% change to avoid flooding the event queue with excessive battery capacity change events.

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

8.8.7 Asynchronous Operation Status

An asynchronous operation can emit its progress with the following event.

```
Topic: tns1:Monitoring/AsynchronousOperationStatus
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name=“Token" Type=“tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name=“OperationName" Type=“xs:string"/>
    <tt:SimpleItemDescription Name=“ServiceName" Type=“xs:string“/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name=“Progress" Type="xs:float"/> <!– [0.0,1.0] -- >
    <tt:ElementItemDescription Name=“FileProgressStatus" Type=“tt:ArrayOfFileProgress"/> <!– optiona
    <tt:ElementItemDescription Name=“Error" Type=“soapenv:Fault"/> <!– optional ->
  </tt:Data>
</tt:MessageDescription>
```

The Token field refers to the operation unique token value that is returned by an asynchronous operation in its response message. An asynchronous operation using this event shall generate a unique token for each invocation. The OperationName field indicates the name of asynchronous operation. The ServiceName field indicates the name of service in which the asynchronous operation is defined. The combination of service name and operation name uniquely identifies the particular operation. The Progress field reports the completion percentage of an asynchronous operation. The value range of Progress field is [0.0,1.0] where 1.0 indicates the completion of an asynchronous operation. The Error field reports errors during the execution of an asynchronous operation.

8.8.8 Device Management

The following topics signal important device status information:

```
tns1:Device/OperationMode/ShutdownInitiated
tns1:Device/OperationMode/UploadInitiated
```

8.8.9 Liquid level

If measurements of the level of the liquid inside a washer tank are supported, a device should signal whether the level is under the warning threshold using the LowLiquid event.

```
Topic: tns1:Monitoring/Washer/LiquidLow
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="IsLow" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

8.8.10 Mechanial failure

The following event should be generated with true value when a cooling fan fails.

```
Topic: tns1:Device/HardwareFailure/FanFailure
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken "/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="Failed" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with true value when a power supply fails.

```
Topic: tns1:Device/HardwareFailure/PowerSupplyFailure
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken "/>
```

```
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="Failed" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with true value when a mass storage device fails.

```
Topic: tns1:Device/HardwareFailure/StorageFailure
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken "/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="Failed" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

The following event should be generated with true value when the device reaches a temperature outside the normal range of operation, and should be generated with false value when the device returns to normal temperature range.

```
Topic: tns1:Device/HardwareFailure/TemperatureCritical
<tt:MessageDescription IsProperty="true">
  <tt:Data>
    <tt:SimpleItemDescription Name="Critical" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

8.8.11 Geo Location

If a device supports monitoring of geo location, it should provide the geo location monitoring event to inform a client about its current location in geo-referenced coordinates. It is recommended to use reasonable time interval to avoid flooding the event queue with excessive processing unit usage events.

```
Topic: tns1:Monitoring/GeoLocation
Event description:
<tt:MessageDescription IsProperty="true“>
  <tt:source>
    <tt:SimpleItemDescription Name="Token" Type="tt:ReferenceToken"/>
  </tt:source>
  <tt:Data>
    <tt:ElementItemDescription Name="CurrentLocation" Type=“tt:GeoLocation"/>
  </tt:Data>
</tt:MessageDescription>
```

9 Event handling

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 specification defines event handling based on the [WS-BaseNotification] and [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 section 9.4.2.

The description of event payload and their filtering within subscriptions is discussed in section 9.4. Section 9.5 describes how a synchronization point can be requested by clients using one of the three notification interfaces. Section 9.6 describes the integration of Topics and section 9.9 discusses the handling of faults.

Section 9.10 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 [WS-BaseNotification] specification.

Both device and client shall support [WS-Addressing] for event services.

9.1 Real-time Pull-Point Notification Interface

This section 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. Additioanlly it provides the Renew and Unsubscribe operations of the Base Subscription Manager Interface. The sequence diagram of the interaction is shown in Figure 5.

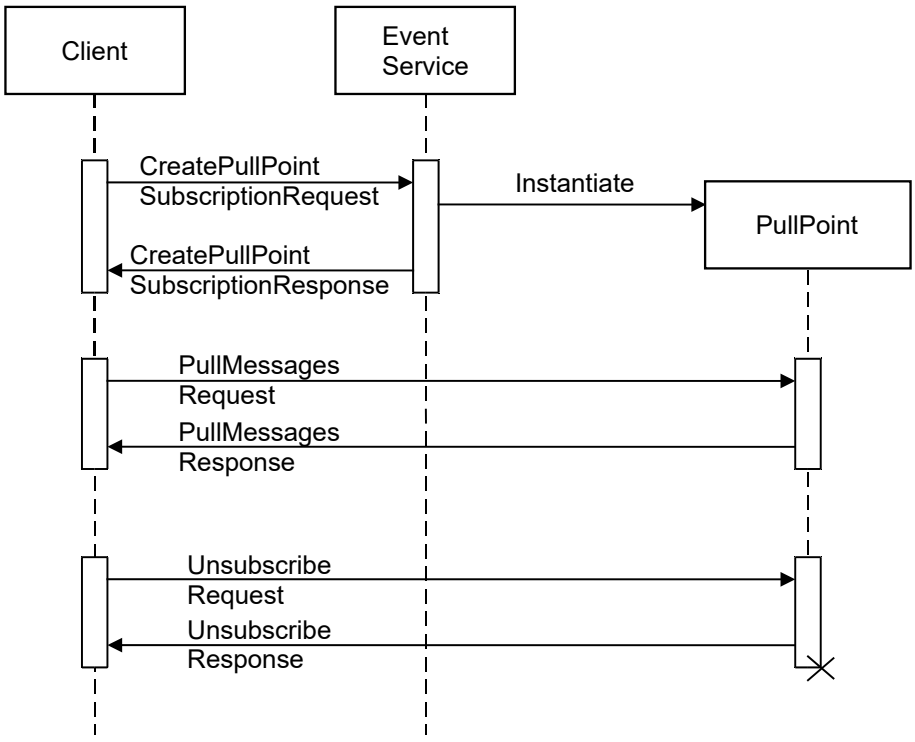

Figure 5: Sequence diagram for the Real-time Pull-Point Notification Interface

4. The device immediately responds 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 pullpoints per client) in order to allow precise event generation. If a device would 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 by e.g. activating multiple motion detectors and similar without need. Additionally the traffic produced by all these events may cause a substantial network load.

If the device supports persistent notification storage, see 9.1.7, 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 revserse the pull direction. There is also a BeginOfBuffer event, as defined in 9.11.1, that signals the start of the buffer.

An ONVIF compliant device shall implement the Real Time Pull-Point Notification Interface.

9.1.1 Create pull point subscription

An ONVIF compliant device shall provide the CreatePullPointSubscription command. If no Filter element is specified the pullpoint 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.

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 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 pullpoint should not provide Initialized nor Deleted events for Properties.

Both request and response message contain the same elements as the SubscriptionRequest and Response of [WS-BaseNotification] without the ConsumerReference.

REQUEST:

- **Filter - optional [wsnt:FilterType]**
  Optional filtering for e.g. topics.
- **InitialTerminationTime - optional [wsnt:AbsoluteOrRelativeTimeType]**
  Initial termination time.
- **SubscriptionPolicy - optional [xs:any]**

RESPONSE:

- **SubscriptionReference [wsa:EndpointReferenceType]**
  Endpoint reference of the subscription to be used for pulling the messages.
- **CurrentTime [xs:dateTime]**
  Current time of the server for synchronization purposes.
- **TerminationTime [xs:dateTime]**
  Date time when the PullPoint will be shut down without further pull requests.

FAULTS:

- The same faults as for Subscription Request of the [WS-BaseNotification] are used.

ACCESS CLASS:

READ_MEDIA

9.1.2 Pull messages

The device shall provide the following PullMessages command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command.

The device shall support a Timeout of at least one minute. The device shall not respond with a PullMessagesFaultResponse when the MessageLimit is greater than the device supports. Instead, the device shall return up to the supported messages in the response.

The response behavior shall be one of three types:

- If there are one or more messages waiting (i.e., aggregated) when the request arrives, the device shall immediately respond with the waiting messages, up to the MessageLimit. The device shall not discard unsent messages, but shall await the next PullMessages request to send remaining messages.
- If there are no messages waiting, and the device generates a message (or multiple simultaneous messages) prior to reaching the Timeout, the device shall immediately respond with the generated messages, up to the MessageLimit. The device shall not wait for additional messages before returning the response.
- If there are no messages waiting, and the device does not generate any message prior to reaching the Timeout, the device shall respond with zero messages. The device shall not return a response with zero messages prior to reaching the Timeout.

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.

REQUEST:

- **Timeout [xs:duration]**
  Maximum time to block until this method returns.
- **MessageLimit [xs:int]**
  Upper limit for the number of messages to return at once. A server implementation may decide to return less messages.

RESPONSE:

- **CurrentTime [xs:dateTime]**
  The date and time when the messages have been delivered by the web server to the client.
- **TerminationTime [xs:dateTime]**
  Date time when the PullPoint will be shut down without further pull requests.
- **NotificationMessage - optional, unbounded [wsnt:NotificationMessageHolderType]**
  List of messages. This list shall be empty in case of a timeout.

PULLMESSAGESFAULTRESPONSE:

- **MaxTimeout [xs:duration]**
  Only when the Timeout exceeds the upper limit supported by the device. Not sent when the Message-Limit is exceeded. The Fault Message shall contain the upper limits for both parameters.
- **MaxMessageLimit [xs:int]**

FAULTS:

-
  No specific fault codes.

ACCESS CLASS:

READ_MEDIA

9.1.3 Renew

An ONVIF compliant device shall support this command if it signals support for [WS-Base Notification] via the MaxNotificationProducers capability.

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.

REQUEST:

- **TerminationTime [wsnt:AbsoluteOrRelativeTimeType]**
  The new relative or absolute termination time.

RESPONSE:

- **CurrentTime [xs:dateTime]**
  The current server time.
- **TerminationTime [xs:dateTime]**
  The updated TerminationTime for the SubscriptionManager.

RESOURCEUNKNOWNFAULTRESPONSE:

- **Timestamp [xs:dateTime]**
  The pull point reference is invalid
- **Originator - optional [wsa:EndpointReferenceType]**
- **ErrorCode - optional [xs:any]**

UNACCEPTABLETERMINATIONTIMEFAULTRESPONSE:

- **Timestamp [xs:dateTime]**
  The Timeout exceeds the upper limit supported by the device.
- **Originator - optional [wsa:EndpointReferenceType]**
- **ErrorCode - optional [xs:any]**

FAULTS:

-
  No specific fault codes.

ACCESS CLASS:

READ_MEDIA

9.1.4 Unsubscribe

The device shall provide the following Unsubscribe command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command. The command is defined in section 6.1.2 of [OASIS Web Services Base Notification 1.3 [http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf]].

This command shall terminate the lifetime of a pull point.

REQUEST:

This is an empty message.

RESPONSE:

This is an empty message.

RESOURCEUNKNOWNFAULTRESPONSE:

- **Timestamp [xs:dateTime]**
  The pull point reference is invalid

- **Originator - optional [wsa:EndpointReferenceType]**
- **ErrorCode - optional [xs:any]**

FAULTS:

- No specific fault codes.

ACCESS CLASS:

READ_MEDIA

9.1.5 Seek

A device supporting persistent notification storage as defined in section 9.1.7 shall provide the following Seek command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command.

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

After a Seek request a pullpoint shall ignore the behavior described in section 9.6 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 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 generate property state as response to a call to the Seek method.

REQUEST:

- **UtcTime [xs:datetime]**
  This message shall be addressed to a PullPoint in order to readjust the pull position:
- **Reverse - optional [xs:bool]**

RESPONSE:

This is an empty message.

FAULTS:

- No specific fault codes.

ACCESS CLASS:

READ_MEDIA

9.1.6 Pull Point Lifecycle

Figure 5 depicts the basic operation of a pull point. This chapter 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 a power cycle of a device.

9.1.7 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 its support persistent notification storage with the PersistentNotificationStorage capability. See section 9.8.

This specification defines 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 e.g. computational dealy.

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

9.2 Notification Streaming Interface

This section defines the transmission of events via RTP streaming packets. For details regarding the configuration see section “Metadata Configuration“ of the ONVIF Media Service Specification.

The following requirements apply if a devices supports transmission of events via RTP streaming packets:

- The events shall be encoded as wsnt:NotificationMessage as defined in [WS-BaseNotification] to transport the Message Payload, the Topic and the ProducerReference.
- Multiple instances of the wsnt:NotificationMessage elements can be placed within a metadata document.
- Since there is no explicit SubscriptionReference with streaming notifications, the wsnt:NotificationMessage shall not contain the SubscriptionReference element.

9.3 Basic Notification Interface

Section 9.3.1 briefly introduces the Basic Notification Interface of the [WS-BaseNotification] specification. Section 9.3.2 summarizes the mandatory and the optional interfaces of the [WS-BaseNotification] specification. Please refer for a full documentation of the Basic Notification Interface to the [WS-BaseNotification] specification.

9.3.1 Introduction

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 6. First, the client establishes a connection to the Event Service. The client can then subscribe for 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 [WS-BaseNotification] and is up to the implementation of the device.

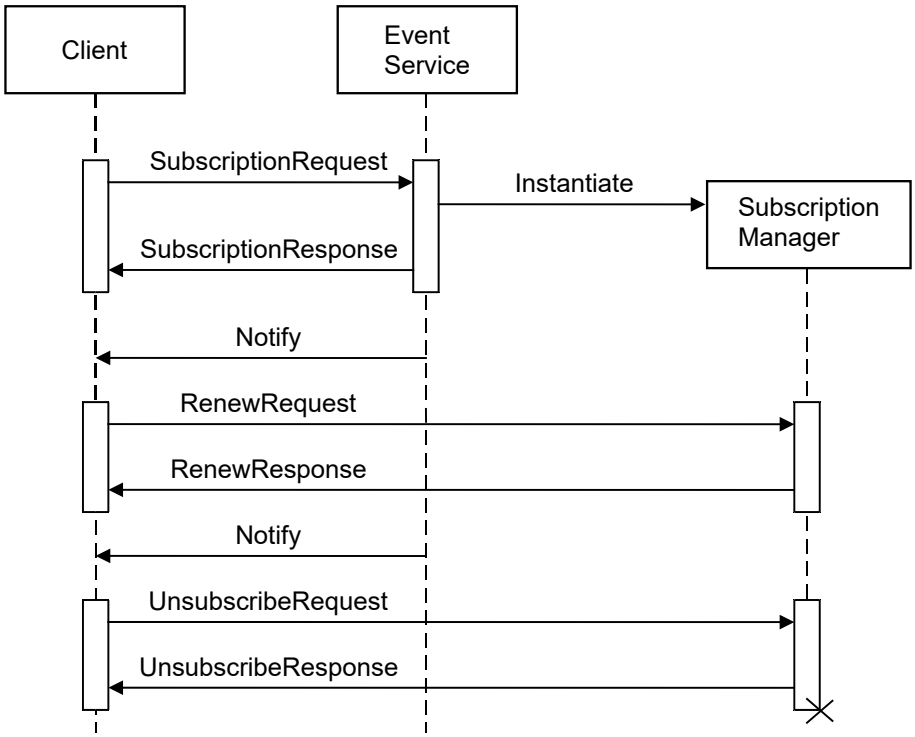

Figure 6: Sequence diagram for the Base Notification Interface

9.3.2 Requirements

This section details those interfaces of the [WS-BaseNotification] that a device shall provide.

An ONVIF compliant device shall support the NotificationProducer Interface of the [WS-BaseNotification] if the capability MaxNotificationProducers is non-zero. The device shall support TopicExpression filters with the dialects described in 9.6.3. 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 [WS-BaseNotification] specification. The SubscriptionPolicy `wsnt:UseRaw` is optional for the device. Although the [WS-BaseNotification] has CurrentTime and TerminationTime as optional elements in a SubscribeResponse and RenewResponse, an ONVIF compliant 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 [WS-BaseNotification] on a device is optional.

An ONVIF compliant device shall implement the Base Subscription Manager Interface of the [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.

An ONVIF compliant 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.

9.4 Event Notifications

A notification answers 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. An ONVIF compliant 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, either a single parameter, multiple parameters, or none may be needed to uniquely identify the component. 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.

ONVIF uses the NotificationMessage type from [WS-BaseNotification] to hold one or more notification messages of type tt:Message:

```
<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>
          <tt:element name="Message" type="tt:Message" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Section 9.4.1 gives a detailed formatting of the Message payload, and Section 9.4.3 introduces a description language for the Message payload. Section 9.4.4 defines the grammar used in a subscription to filter notifications by their Message content.

9.4.1 Notification Message Format

The main information elements of a notification message are:

- UtcTime represents the time when the event described by the message occurred.
- Topic and Source items uniquely identify the source of an event
- Data contains one or more values describing the event
- An optional Key item may extend the Source identifier

Source, Data and Key are structured as item lists. Each can hold an arbitrary number of items of type SimpleItem or ElementItem. 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.

ElementItem should not be used in the Source and Key elements.

Vendor specific extensions shall express the SimpleItem and ElementItem Name attribute as QName. This avoids potential name clashes between Vendor specific extensions and future ONVIF extensions.

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 section 9.6), where each topic can be augmented by a description of the message payload.

The subsequent example demonstrates the different parts of the notification:

```
<wsnt:NotificationMessage>
  ...
  <wsnt:Topic Dialect="...Concrete">
    tns1:PTZController/PTZPreset/Reached
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="...">
      <tt:Source>
        <tt:SimpleItem Name="PTZConfigurationToken" Value="PTZConfig1"/>
      </tt:Source>
      <tt:Data>
        <tt:SimpleItem Name="PresetToken" Value="Preset5"/>
        <tt:SimpleItem Name="PresetName"  Value="ParkingLot"/>
      </tt:Data>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

The Item “PTZConfigurationToken” uniquely identifies the component, which is responsible for the detection of the Event. In this example, the component is a PTZ Node referenced by the PTZ Configuration “PTZConfig1”. The event `tns1:PTZController/PTZPreset/Reached` indicates that the PTZ unit has arrived at a preset. The data block contains information about the preset that has been reached. The Preset is identified by a PresetToken “Preset5” which is named “PresetName”.

9.4.2 Property Events

This specification introduces the notion of a propery event which allows observation of state changes of properties. As with other events a property is uniquely identified by its Topic and Source. The state of a property is reflected by the values of its Data items.

Each property has an individual lifecycle in a subscription as shown in

Figure 7.

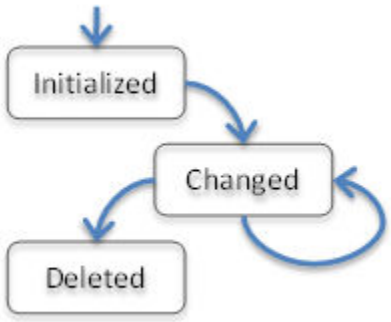

Figure 7: Property event state chart

When a client subscribes to a topic representing certain properties, the device shall provide notifications informing the client of all objects with the requested property, which are alive at the time of the subscription. After all existing objects have been reported the device shall send notifications when a property has changed, is deleted or a new one created. A client may also request the values of all currently alive properties the client has subscribed to at any time by asking for a synchronization point (see section 9.5).

A notification message of a property event shall include the PropertyOperation attribute. The operation mode “Initialized” shall be used to inform a client about the creation of a property. The operation mode “Initialized” shall also be used when a synchronization point has been requested.

The property interface is defined in this standard 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. Section 9.4 explains the structure of events and properties in detail.

9.4.2.1 Property Example

The following video analytics example demonstrates the dynamic behaviour of properties: The rule engine interface of the video analytics detector can define fields. Such a detector field is described by a polygon in the image plane. For each object in the scene, the rule engine determines which objects are within the polygon. A client can access this information by subscribing to the corresponding ObjectsInside property of the detector field. Each time an object appears in the scene, a new ObjectsInside property is created. The client is informed by a corresponding “property created” notification indicating if the object appeared inside or outside the polygon. Each time an object enters or leaves the polygon, a “property changed” notification is produced indicating that the ObjectsInside property for this object has changed. When an object leaves the scene, the corresponding ObjectsInside property is deleted and the client is informed via a “property deleted” notification.

The example in this section demonstrates the application of Key Items. The rule engine can contain FieldDetector rules. These rules define an ObjectsInside property for each object in the scene. When a new object appears outside of such a Field, the following notification is produced:

```
<wsnt:NotificationMessage>
  ...
  <wsnt:Topic Dialect="...Concrete">
    tns1:RuleEngine/FieldDetector/ObjectsInside
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="..." PropertyOperation="Initialized">
      <tt:Source>
        <tt:SimpleItem Name="VideoSourceConfigurationToken"    Value="1"/>
        <tt:SimpleItem Name="VideoAnalyticsConfigurationToken" Value="1"/>
        <tt:SimpleItem Name="Rule"             Value="myImportantField"/>
      </tt:Source>
      <tt:Key>
        <tt:SimpleItem Name="ObjectId"                         Value="5"/>
      </tt:Key>
      <tt:Data>
        <tt:SimpleItem Name="IsInside"                    Value="false"/>
      </tt:Data>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

The Source Items describe the Rule which produced the notification. When multiple objects are in the scene, each of these objects has its own ObjectsInside property. Therefore, the Object ID is used as an additional Key Item in order to make the property unique. The IsInside Item is a Boolean value indicating whether the object is inside or outside of the Field.

When the object enters the Field, the rule produces a “property changed” message and resembles the following:

```
<wsnt:NotificationMessage>
  ...
```

```
  <wsnt:Topic Dialect="...Concrete">
    tns1:RuleEngine/FieldDetector/ObjectsInside
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="..." PropertyOperation="Changed">
      <tt:Source>
        <tt:SimpleItem Name="VideoSourceConfigurationToken"    Value="1"/>
        <tt:SimpleItem Name="VideoAnalyticsConfigurationToken" Value="1"/>
        <tt:SimpleItem Name="Rule"             Value="myImportantField"/>
      </tt:Source>
      <tt:Key>
        <tt:SimpleItem Name="ObjectId"                         Value="5"/>
      </tt:Key>
      <tt:Data>
        <tt:SimpleItem Name="IsInside"                     Value="true"/>
      </tt:Data>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

Finally, when the object leaves the scene, a “property deleted” message is produced:

```
<wsnt:NotificationMessage>
  ...
  <wsnt:Topic Dialect="...Concrete">
    tns1:RuleEngine/FieldDetector/ObjectsInside
  </wsnt:Topic>
  <wsnt:Message>
    <tt:Message UtcTime="..." PropertyOperation="Deleted">
      <tt:Source>
        <tt:SimpleItem Name="VideoSourceConfigurationToken"    Value="1"/>
        <tt:SimpleItem Name="VideoAnalyticsConfigurationToken" Value="1"/>
        <tt:SimpleItem Name="Rule"             Value="myImportantField"/>
      </tt:Source>
      <tt:Key>
        <tt:SimpleItem Name="ObjectId"                         Value="5"/>
      </tt:Key>
    </tt:Message>
  </wsnt:Message>
</wsnt:NotificationMessage>
```

In this case, the Data item can be omitted because the object and its corresponding property no longer exists.

9.4.3 Message Description Language

The structure of the Message payload was introduced in the previous section. 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[1]:

```
<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>
```

[1]Please note that the schema is included here for *information only*. [ONVIF Schema] contains the normative schema definition.

```
    <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), ONVIF schemas, 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 ONVIF Event definition are not required to be present to 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 Section 9.7.

9.4.3.1 Message Description Example

The following code is an example of a Message Description corresponding to the Property example of Section 9.4.2.1:

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
      Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescriptionD Name="VideoAnalyticsConfigurationToken"
      Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Rule"
      Type="xs:string"/>
  </tt:Source>
  <tt:Key>
    <tt:SimpleItemDescription Name="ObjectId"
      Type="xs:integer"/>
  </tt:Key>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsInside"
      Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

9.4.4 Message Content Filter

In the Subscription request, a client can filter notifications by TopicExpression (see Section 9.6.3) and by MessageContent. For the latter, the [WS-BaseNotification] proposes the XPath 1.0 dialect. Due to the specific Message structure required by this specification, the specification requires a subset of the XPath 1.0 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 independent 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 VideoSourceConfiguration “1”

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

Return only notifications which do not contain a reference to a VideoAnalyticsConfiguration

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

Return only notifications which do relate to VideoAnalyticsConfiguration “2” running on VideoSourceConfiguration “1”

```
boolean(//tt:SimpleItem[@Name=”VideoAnalyticsConfigurationToken” and @Value=”2”] )
  and boolean(//tt:SimpleItem[@Name=”VideoSourceConfigurationToken” and @Value=”1”] )
```

Return only notifications which are related to VideoSourceConfiguration “1” but are not related to VideoAnalyticsConfigurations

```
boolean(//tt:SimpleItem[@Name=”VideoSourceConfigurationToken” and @Value=”1”] )
  and not(boolean(//tt:SimpleItem[@Name=”VideoAnalyticsConfigurationToken”] ))
```

Return only notifications when objects enter or appear in “myImportantField”

```
boolean(//tt:SimpleItem[@Name=”IsInside” and @Value=”true”] ) and
  boolean(//tt:SimpleItem[@Name=”Rule” and @Value=”myImportantField”] )
```

9.5 Synchronization Point

Note that section 9.1.5 defines rules for devices supporting persistent notification storage that override the behavior defined in this section.

Properties, introduced in section 9.2, 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. The PropertyOperation of all produced notifications is set to “Initialized” (see Section 9.4). 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:

REQUEST:

This is an empty message.

RESPONSE:

This is an empty message.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_MEDIA

When a client uses the notification streaming interface, the client should use the SetSynchronizationPoint operation defined in the ONVIF Media Service Specification.

9.6 Topic Structure

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

Section 9.6.1 describes the ONVIF Topic Namespace. Section 9.6.2 incorporates the Message Description Language defined in section 9.4.3 into the TopicSet structure, furthermore section 9.7 defines an interface that allows a client to get this information. A Topic Expression Dialects to be supported by a device is defined in section 9.6.3.

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

9.6.1 ONVIF Topic Namespace

The [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 [WS-Topics] specification.

All notifications referring to topics in the ONVIF topic namespace shall use the Message Format as described in Section 9.4.2 [85].

9.6.2 Topic Type Information

A device shall add a MessageDescription element, of type MessageDescriptionType defined in Section 9.4.3, 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 element 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:RuleEngine>
    <LineDetector>
      <Crossed wstop:topic="true">
        <tt:MessageDescription>
          <tt:Source>
            <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
              Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
              Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
          </tt:Data>
        </tt:MessageDescription>
      </ Crossed>
    </LineDetector>
    <FieldDetector>
      <ObjectsInside wstop:topic="true">
        <tt:MessageDescription IsProperty="true">
          <tt:Source>
            <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
              Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
              Type="tt:ReferenceToken"/>
            <tt:SimpleItemDescription Name="Rule" Type="xs:string"/>
          </tt:Source>
          <tt:Key>
            <tt:SimpleItemDescription Name="ObjectId" Type="xs:integer"/>
          </tt;Key>
          <tt:Data>
            <tt:SimpleItemDescription Name="IsInside" Type="xs:boolean"/>
          </tt:Data>
        </tt:MessageDescription>
      </ObjectsInside>
    </FieldDetector>
  </tns1:RuleEngine>
</wstop:TopicSet>
```

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

9.6.3 Topic Filter

An ONVIF compliant device shall support the Concrete Topic Expressions defined in the [WS-Topics] specification. This specification 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 subtree 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 [WS-Topics 1.3] 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:RuleEngine/FieldDetector//."
```

This identifies the sub-tree consisting of tns1:RuleEngine/FieldDetector and all its descendents.

The following examples demonstrate the usage of the ConcreteSet topicExpression:

Look for notifications which have the VideoAnalytics topic as parent topic:

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

Look for notifications which have the VideoAnalytics topic or the RuleEngine as parent topic:

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

Look for notifications produced by either a LineDetector or a FieldDetector:

```
<wsnt:TopicExpression Dialect ="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
tns1:RuleEngine/FieldDetector//.|tns1:RuleEngine/LineDetector//.
</wsnt:TopicExpression>
```

9.7 Get event properties

The [WS-BaseNotification] specification defines a set of optional WS-ResouceProperties. This specification does not require the implementation of the WS-ResourceProperty interface. Instead, the subsequent direct interface shall be implemented by an ONVIF compliant device in order to provide information about the Filter-Dialects, Schema files and topics supported by the device.

REQUEST:

This is an empty message.

RESPONSE:

- **TopicNamespaceLocation - unbounded [xs:anyURI]**
  List of topic namespaces supported.
- **FixedTopicSet [xs:boolean]**
  True when topicset is fixed for all times.
- **TopicSet [wstop:TopicSetType]**
  Set of topics supported.

- **TopicExpressionDialect - unbounded [xs:anyURI]**
  Defines the XPath expression syntax supported for matching topic expressions.
- **MessageContentFilterDialect - unbounded [xs:anyURI]**
  Defines the XPath function set supported for message content filtering.
- **ProducerPropertiesFilterDialect - optional, unbounded [xs:anyURI]**
  Optional ProducerPropertiesDialects. Refer to Web Services Base Notification 1.3 (WS-BaseNotification) for advanced filtering.
- **MessageContentSchemaLocation - unbounded [xs:anyURI]**
  The Message Content Description Language allows referencing of vendor-specific types.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

READ_MEDIA

An ONVIF compliant 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 an ONVIF compliant device (see Section 9.6.3):

```
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 specification does not require the support of any ProducerPropertiesDialect by a device.

The Message Content Description Language, introduced in Section 9.4.3, allows referencing of vendor-specific types. In order to ease the integration of such types into a client application, the GetEventPropertiesResponse 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 ONVIF schema file.

9.8 Capabilities

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 Section 9.3.2
WSPullPointSupport	Indication if the device supports the WS Pull Point according to Section 9.3.2
WSPausableSubscriptionManagerInterfaceSupport	Indication if the device supports the WS Pausable Subscription Manager Interface according to Section 9.3.2
MaxNotificationProducers	Maximum number of supported notification producers as defined by WS-BaseNotification.
MaxPullPoints	Maximum supported number of notification pull points
PersistenNotificationStorage	Indication if the device supports persistent notification storage according to Section 9.1.7.
EventBrokerProtocols	A space separated list of supported event broker protocols as defined by the datatype tev:EventBrokerProtocol.

MaxEventBrokers Maxiumum number of event broker configurations that can be added to the device.

REQUEST:

This is an empty message.

RESPONSE:

- **Capabilities [tev:Capabilities]**
  The capability response message contains the requested service capabilities using a hierarchical XML capability structure.

FAULTS:

No command specific faults defined.

ACCESS CLASS:

PRE_AUTH

9.9 SOAP Fault Messages

If a device encounters a failure while processing [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 [WS-BaseNotification] and [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.

9.10 Notification example

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

9.10.1 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/GetEventPropertiesRequest
      </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tet:GetEventProperties>
      </tet:GetEventProperties>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

9.10.2 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:RuleEngine/LineDetector/Crossed`. The Message associated with this

topic contains information about the VideoSourceConfigurationToken, the VideoAnalyticsConfigurationToken and the object which has crossed the line. 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/GetEventPropertiesResponse
      </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:RuleEngine>
            <LineDetector>
              <Crossed wstop:topic="true">
                <tt:MessageDescription>
                  <tt:Source>
                    <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                      Type="tt:ReferenceToken"/>
                    <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                      Type="tt:ReferenceToken"/>
                  </tt:Source>
                  <tt:Data>
                    <tt:SimpleItemDescription Name="ObjectId"
                      Type="xs:integer"/>
                  </tt:Data>
                </tt:MessageDescription>
              </Crossed>
            </LineDetector>
          </tns1:RuleEngine>
        </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>
```

9.10.3 CreatePullPointSubscription

A client can subscribe to specific notifications with the information from the TopicProperties. The following XML example shows the subscription for notifications produced by the Rule Engine of the device. The client

reacts only to notifications that reference VideoAnalyticsConfiguration “2” and VideoSourceConfiguration “1”. 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/CreatePullPointSubscriptionRequest
      </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tet:CreatePullPointSubscription>
        <tet:Filter>
          <wsnt:TopicExpression
            Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
            tns1:RuleEngine//.
          </wsnt:TopicExpression>
          <wsnt:MessageContent Dialect="http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilt
            boolean(//tt:SimpleItem[@Name="VideoAnalyticsConfigurationToken"
            and @Value="2"] ) and
            boolean(//tt:SimpleItem[@Name="VideoSourceConfigurationToken"
            and @Value="1"] )
          </wsnt:MessageContent>
        </tet:Filter>
        <tet:InitialTerminationTime>
          PT1M
        </tet:InitialTerminationTime>
      </tet:CreatePullPointSubscription>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

9.10.4 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/CreatePullPointSubscriptionResponse
      </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>
```

9.10.5 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/PullMessagesRequest        </ws
        <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>
```

9.10.6 PullMessagesResponse

The following PullMessageResponse contains two notifications which match the subscription. The Response informs the client that two objects have crossed lines corresponding to rules “MyImportantFence1” and “MyImportantFence2”.

```
<?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/PullMessagesResponse
      </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:RuleEngine/LineDetector/Crossed
          </wsnt:Topic>
          <wsnt:Message>
            <tt:Message UtcTime="2008-10-10T12:24:57.321Z">
              <tt:Source>
                <tt:SimpleItem Name="VideoSourceConfigurationToken"
                  Value="1"/>
                <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                  Value="2"/>
                <tt:SimpleItem Value="MyImportantFence1" Name="Rule"/>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItem Name="ObjectId" Value="15" />
              </tt:Data>
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
        <wsnt:NotificationMessage>
          <wsnt:Topic Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
            tns1:RuleEngine/LineDetector/Crossed
          </wsnt:Topic>
          <wsnt:Message>
            <tt:Message UtcTime="2008-10-10T12:24:57.789Z">
              <tt:Source>
                <tt:SimpleItem Name="VideoSourceConfigurationToken"
                  Value="1"/>
                <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                  Value="2"/>
                <tt:SimpleItem Value="MyImportantFence2" Name="Rule"/>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItem Name="ObjectId" Value="19"/>
              </tt:Data>
            </tt:Message>
          </wsnt:Message>
        </wsnt:NotificationMessage>
      </tet:PullMessagesResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

9.10.7 UnsubscribeRequest

A client has to terminate a subscription explicitly with an UnsubscribeRequest 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>
```

9.10.8 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>
```

9.11 Persistent storage event

9.11.1 BeginOfBuffer

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 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 at most send the beginning of buffer event one time.

```
Topic: tns1:EventBuffer/Begin
<tt:MessageDescription IsProperty=”false”/>
```

9.12 Event Broker

Events can be bridged from an ONVIF device to an MQTT broker. This section describes the event broker configuration interface, how ONVIF topics are mapped to MQTT topics and how the payload is conveyed.

A device that signals the capability MaxEventBrokers > 0 shall support the AddEventBroker, DeleteEventBroker and GetEventBroker commands.

9.12.1 Data structures

9.12.1.1 EventBrokerConfig

- **Address**

  Event broker address in the format scheme://host:port, where scheme can be "mqtt", "mqtts", "ws" or "wss". The supported schemes shall be returned by the EventBrokerProtocols capability. The Address must be unique.

- **TopicPrefix**

  Prefix that will be prepended to all topics before they are published. This is used to make published topics unique for each device. TopicPrefix is not allowed to be empty.

- **UserName**

  User name for the event broker.

- **Password**

Password for the event broker. Password shall not be included when returned with GetEventBrokers.

- **CertificateID**

  Optional certificate ID in the key store pointing to a client certificate to be used for authenticating the device at the message broker.

- **PublishFilter**

  Concrete Topic Expression to select specific topics to publish, see section 9.6.3.

- **QoS**

  Quality of service level to use when publishing. This defines the guarantee of delivery for a specific message: 0 = At most once, 1 = At least once, 2 = Exactly once.

- **Status**

  Current connection status (see tev:ConnectionStatus for possible values).

9.12.2 AddEventBroker

The AddEventBroker command allows an ONVIF client to add an event broker configuration to device to enable ONVIF events to be transferred to an event broker. If an existing event broker configuration already exists with the same Address, the existing configuration shall be modified.

REQUEST:

- **EventBroker [tev:EventBrokerConfig]**
  The event broker definition to be added or modified.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Receiver - ter:Action - ter:TooManyEventBrokers**
  The device doesn’t support adding more event broker configurations.
- **env:Sender - ter:InvalidArgVal - ter:InvalidAddress**
  The event broker address is not supported.
- **env:Sender - ter:InvalidArgVal - ter:InvalidProtocol**
  The event broker protocol is not supported.
- **env:Sender - ter:InvalidArgVal - ter:InvalidFilter**
  The topic filter was not understood.

ACCESS CLASS:

WRITE_SYSTEM

9.12.3 DeleteEventBroker

The DeleteEventBroker allows an ONVIF client to delete an event broker configuration from an ONVIF device.

REQUEST:

- **Address [xs:anyURI]**
  The uri of the event broker to be removed.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidAddress**
  The event broker address was not found.

ACCESS CLASS:

WRITE_SYSTEM

9.12.4 GetEventBrokers

The GetEventBrokers command lets a client retrieve event broker configurations from the device. If Address is specified one event broker configuration shall be returned, if the Address is found, otherwise a fault is returned. If no Address is specified all event broker configurations shall be returned.

REQUEST:

- **Address - optional [xs:anyURI]**
  Optional address filter.

RESPONSE:

- **EventBroker - optional, unbounded [tev:EventBrokerConfig]**
  List of configured event broker definitions.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:InvalidAddress**
  The event broker address was not found.

ACCESS CLASS:

READ_MEDIA

9.12.5 Topic Structure

Topics are published according to the following structure:

<TopicPrefix>/<PayloadPrefix>/<LocalTopic>[/&<Source>[/<Key>]]

ONVIF topics are are only locally unique within the device so they must be prefixed with a TopicPrefix to become globally unique. The TopicPrefix is configurable through the AddEventBroker command. The TopicPrefix shall not be empty. The PayloadPrefix signals what kind of data that is published. The following prefixes are defined.

Table 10: Prefix definitions

Prefix	Description
onvif-ej	ONVIF event with JSON payload

The LocalTopic is the same as the ONVIF topic, but because MQTT does not support namespaces, the namespace prefix for ONVIF topics shall be dropped, so that for example, “tns1:Device/HardwareFailure/StorageFailure” becomes “Device/HardwareFailure/StorageFailure”. This means that the default topic namespace is ONVIF, i.e. “http://www.onvif.org/ver10/topics”. If an event uses another topic namespace this should be signalled using the syntax: "tns:{namespace-alias}/<topic>". Vendor specific extensions should choose a suitable namespace alias to avoid name clashes. As an example, consider the ONVIF topic "tns1:Device/HardwareFailure/acme:LensFailure" where tns1="http://www.onvif.org/ver10/topics" and acme="http://www.acme.com/topics". This should be translated to the MQTT topic "Device/HardwareFailure/tns:acme/LensFailure".

The Source and Key are taken from the payload, preceded by a '&' character. For property events, they are added to the topic to make it unique so that they can be cached by the broker individually. For each SimpleItem name/value pair in the Source/Key a '/' character plus the value shall be appended to the topic. The values shall be added in the same order as they are listed in the response to GetEventProperties. Note that special characters like, '/', '#' and '+' shall be omitted from the topic. For an example, see section 9.12.6.1.

9.12.6 JSON Event Payload

This section describes how an ONVIF event is mapped to the JSON data format when published using the "onvif-ej" payload definition. Mapping definition from XML to JSON according to RFC 5234:

MESSAGE ::= "{" TIMEINFO *("," ITEM) "}"

TIMEINFO ::= DQUOTE "UtcTime" DQUOTE ":" TIMESTAMP

ITEM ::= SIMPLEITEM / ELEMENTITEM

SIMPLEITEM ::= DQUOTE SNAME DQUOTE ":" DQUOTE SVALUE DQUOTE

ELEMENTITEM ::= DQUOTE ENAME DQUOTE ": {" [TYPE *("," TYPE)] "}"

TYPE ::= ATTRIBUTE / SIMPLETYPE / COMPLEXTYPE

ATTRIBUTE ::= DQUOTE ANAME DQUOTE ": " DQUOTE AVALUE DQUOTE

SIMPLETYPE ::= DQUOTE NAME DQUOTE ": " DQUOTE VALUE DQUOTE

COMPLEXTYPE ::= DQUOTE NAME DQUOTE ": {" [TYPE *("," TYPE)] "}"

with

- TIMESTAMP being the double quoted UTC timestamp of the XML Message UtcTime element
- SNAME being the value of the "Name" attribute of the SimpleItem
- SVALUE being the value of the "Value" attribute of the SimpleItem
- ENAME being the value of the "Name" attribute of the ElementItem
- ANAME being the name of the attribute of the corresponding XML attribute
- AVALUE being the value of the attribute of the corresponding XML attribute
- SIMPLETYPE being the XML attribute currently parsed
- NAME being the name of the corresponding XML element
- VALUE being the value of the corresponding XML element

The parser shall parse the elements recursively in the order they are present in the XML representation.

The output shall be generated according above described rules.

9.12.6.1 Example

The XML example response from PullMessages listed in Section 9.10.6 contains two messages which are mapped to the following corresponding MQTT topics and JSON payload:

```
Topic:
  MyDevice/onvif-ej/RuleEngine/LineDetector/Crossed/&1/2/MyImportantFence1
Payload:
```

```
{
  "UtcTime": "2008-10-10T12:24:57.321Z",
  "Source": {
    "VideoSourceConfigurationToken": "1",
    "VideoAnalyticsConfigurationToken": "2",
    "Rule": "MyImportantFence1"
    },
  "Data": {
    "ObjectId": "15"  }
}}
Topic:
  MyDevice/onvif-ej/RuleEngine/LineDetector/Crossed/&1/2/MyImportantFence2Payload:
Payload:
{
  "UtcTime": "2008-10-10T12:24:57.789Z
  "Source": {
    "VideoSourceConfigurationToken": "1",
    "VideoAnalyticsConfigurationToken": "2",
    "Rule": "MyImportantFence2"
    },
  "Data": {
    "ObjectId": "19"
} }
```

9.12.7 Property events

For property events, the PropertyOperation shall not be included in the message payload.

9.12.7.1 Retained flag

For property events, the device shall set the retained flag when sending the messages to the broker.

9.12.7.2 Payload for deleted properties

For property events, ONVIF compatible devices shall send a notification with PropertyOperation=Deleted when a property is deleted as explained in section 9.4.2. To achieve same behavior, the device shall send a zero byte payload to the MQTT event broker. This will delete any retained messages in the broker for that specific topic.

9.12.8 SetSynchronizationPoint behavior

If a client sends a SetSynchronizationPoint request, property events shall not be republished to the event broker.

Annex A.
Capability List of GetCapabilities (normative)

This annex describes a legacy interface to signal capabilities for a certain service or function using the GetCapabilities method.

Table A.1:

Category	Capability	Description
Analytics	XAddr	The address to the analytics service. If this field is empty the device supports analytics but not the rules or module interfaces.
	RuleSupport	Indication if the device supports rules interface and rules syntax as specified in the Video Analytics Service Specification.
	AnalyticsModuleSupport	Indication if the device supports the scene analytics module interface as specified in the Video Analytics Service Specification.
Device	XAddr	The address to the device service.
Device – Network	IPFilter	Indication if the device supports IP filtering control using the commands in Section 8.2.18, 8.2.19, 8.2.20 and 8.2.21.
	ZeroConfiguration	Indication if the device supports zero configuration according to the commands in Section 8.2.16 and Section 8.2.17.
	IPVersion6	Indication if the device supports IP version 6.
	DynDNS	Indication if the device supports Dynamic DNS configuration according to Section 8.2.8and Section 8.2.9 .
	Dot11Configuration	Indication if the device supports IEEE802.11 configuration as specified in Section 8.2.22
Device – System	DiscoveryResolve	Indication if the device responses to resolve requests as described in Section 7.3.4.
	DiscoveryBye	Indication if the device sends bye messages as described in Section 7.3.5
	RemoteDiscovery	Indication if the device supports remote discovery support.
	SupportedVersions	List of the device supported ONVIF specification versions.

Category	Capability	Description
	SystemBackup	Indication if the device supports system backup and restore as specified in Section 8.3.3 and Section 8.3.5
	FirmwareUpgrade	Indication if the device supports firmware upgrade as specified in Section 8.3.9.
	SystemLogging	Indication if the device supports system log retrieval as specified in Section 8.3.10 [57].
	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 section 8.3.2.
	HTTPSupportInformation	Indication if the device supports retrieval of support information using HTTP Get, see section 8.3.2.
Device – IO	InputConnectors	The number of input connectors.
	RelayOutputs	The number of relay outputs.
	Auxiliary	Indication of support for auxiliary service along with list of supported auxiliary commands
Device – Security	TLS1.0	Support of TLS 1.0.
	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 (deprecated).
	AccessPolicyConfig	Indication if the device supports retrieving and loading device access control policy according to Section 8.4.1 and Section 8.4.2.
	X.509Token	Indication if the device supports the WS-Security X.509 token [WS-X.509Token].
	SAMLToken	Indication if the device supports the WS-Security SAML token [WS-SAMLToken].
	KerberosToken	Indication if the device supports the WS-Security Kerberos token [WS-KerberosToken].

Category	Capability	Description
	RELToken	Indication if the device supports the WS-Security REL token [WS-REL-Token].
	Dot1X	Indication if the device supports IEEE 802.1X port-based network authentication (deprecated).
	SupportedEAPMethod	List of supported EAP Method types. The numbers correspond to the IANA [EAP-Registry].
	RemoteUserHandling	Indication if device supports remote user handling and the corresponding methods defined in section 8.4.7 and 8.4.8.
Event	XAddr	The address to the event service
	WSSubscriptionPolicySupport	Indication if the device supports the WS Subscription policy according to Section9.3.2
	WSPullPointSupport	Indication if the device supports the WS Pull Point according to Section 9.3.2
	WSPausableSubscription-ManagerInterfaceSupport	Indication if the device supports the WS Pausable Subscription Manager Interface according to Section9.3.2
Imaging	XAddr	The address to the imaging service
Media	XAddr	The address to the media service.
Media – streaming	RTPMulticast	Indication of support of UDP multicasting as described in the ONVIF Streaming Specification.
	RTP_TCP	Indication if the device supports RTP over TCP, see ONVIF Streaming Specification.
	RTP_RTSP_TCP	Indication if the device supports RTP/RTSP/TCP transport, see ONVIF Streaming Specification.
Media - profile	MaximumNumberOfProfiles	The maximum Number of MediaProfiles the device supports.
PTZ	XAddr	The address to the PTZ service.
Receiver	XAddr	The address to the receiver service.
	RTP_Multicast	Indication if the device supports receiving of RTP Multicast.
	RTP_TCP	Indication if the device supports receiving of RTP over TCP.
	RTP_RTSP_TCP	Indication if the device supports receiving of RTP over RTSP over TCP

Category	Capability	Description
	SupportedReceivers	The maximum number of receivers the device supports.
	MaximumRTSPURILength	The maximum length allowed for RTSP URIs.
Recording	XAddr	The address to the recording control service.
	DynamicRecordings	Indication if the device supports dynamic creation and deletion of recordings, see ONVIF Recording Configuration Specification.
	DynamicTracks	Indication if the device supports dynamic creation and deletion of tracks, see ONVIF Recording Configuration Specification.
	DeleteData	Indication if the device supports explicit deletion of data, see ONVIF Recording Configuration Specification.
Search	XAddr	The address to the recording search service.
	MetadataSearch	Indication if the device supports generic search of recorded metadata as defined in the ONVIF Recording Search Specification..
Replay	XAddr	The address to the replay service.
Analytics Device	XAddr	The address to the analytics device service of the device.
Display	XAddr	The address to the display service.
Display - layout	FixedLayout	Indication that the SetLayout command supports only predefined layouts..
Device IO	XAddr	The address to the device IO service.
	VideoSources	The number of video inputs
	VideoOutputs	The number of video outputs
	AudioSources	The number of audio inputs
	AudioOutputs	The number of audio outputs
	RelayOutputs	The number of relay outputs.

Annex B.
Bibliography

[EAP-Registry] Extensible Authentication Protocol (EAP) Registry

[http://www.iana.org/assignments/eap-numbers/eap-numbers.xml]

ONVIF Security Recommendations White Paper

[http://www.onvif.org/portals/3/documents/whitepapers/ONVIF_Security_Recommendations_ver10.pdf]
[http://www.onvif.org/portals/3/documents/whitepapers/ONVIF_Security_Recommendations_ver10.pdf%5d]

ONVIF PTZ Coordinate Spaces White Paper

[http://www.onvif.org/Portals/0/documents/whitepapers/ONVIF_PTZ_coordinate_spaces.pdf]

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee et al., August 1998* [http://www.ietf.org/rfc/rfc2396.txt]

[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`

[WS-KerberosToken] “Web Services Security Kerberos Token Profile 1.1”, OASIS Standard, ,1 February 2006.

`http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf`

[WS-SAMLToken] “Web Services Security: SAML Token Profile 1.1”, OASIS Standard, 1 February 2006.

`http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf`

[WS-X.509Token] “Web Services Security X.509 Certificate Token Profile 1.1”, OASIS Standard,1 February 2006.

`http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf`

[WS-RELToken] “Web Services Security Rights Expression Language (REL) Token Profile 1.1”, OASIS Standard, 1 February 2006

`http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf`

[X.680] ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information

Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic

Notation.

[X.681] ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, Information

Technology - Abstract Syntax Notation One (ASN.1): Information Object

Specification.

[X.682] ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, Information

Technology - Abstract Syntax Notation One (ASN.1): Constraint Specification.

[X.683] ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, Information

Technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 Specifications.

[X.690] ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

[ONVIF DM WSDL] ONVIF Device Management Service WSDL, ver 2.1, 2011.

http://www.onvif.org/ver10/device/wsdl/devicemgmt.wsdl

[ONVIF Event WSDL] ONVIF Event Service WSDL, ver 2.1, 2011.

http://www.onvif.org/ver10/event/wsdl/event.wsdl

[ONVIF DP WSDL] ONVIF Remote Discovery Proxy Services WSDL, ver 2.0, 2010.

http://www.onvif.org/ver10/network/wsdl/remotediscovery.wsdl

[ONVIF Schema] ONVIF Schema, ver 2.0, 2010.

http://www.onvif.org/onvif/ver10/schema/onvif.xsd

[ONVIF Topic Namespace] ONVIF Topic Namespace XML, ver 2.0, 2010.

http://www.onvif.org/ver10/topics/topicns.xml

[ONVIF Security] ONVIF Advanced Security Specification

WS-I, Basic Profile Version 2.0 – Working Group Draft, C. Ferris (Ed), A. Karmarkar (Ed) and P. Yendluri (Ed), October 2007.

<http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html>

Annex C.
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:enc="http://www.w3.org/2003/05/soap-encoding"
  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"
              X.509Token="false" SAMLToken="false" KerberosToken="false"
              HttpDigest="false" RELToken="false" />
            <tds:System DiscoveryResolve="true" DiscoveryBye="true"
              HttpFirmwareUpgrade="true"HttpSystemLogging="false" />
            <tds:Misc 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/media/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <trt:Capabilities xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
            SnapshotUri="true" Rotation="false">
            <trt:ProfileCapabilities MaximumNumberOfProfiles="10" />
            <trt:StreamingCapabilities RTPMulticast="true" RTP_TCP="false"
              RTP_RTSP_TCP="true" NonAggregateControl="true" />
          </trt: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/ver20/ptz/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <tptz:Capabilities xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl"
            EFlip="false" Reverse="false" />
```

```
        </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" />
        </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/ver20/imaging/wsdl</tds:Namespace>
        <tds:XAddr>http://192.168.0.10/onvif</tds:XAddr>
        <tds:Capabilities>
          <timg:Capabilities xmlns:timg="http://www.onvif.org/ver20/imaging/wsdl"
            ImageStabilization="false" />
        </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/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"
            VideoSources="1" VideoOutputs="0" AudioSources="1" AudioOutputs="1"
            RelayOutputs="0" SerialPorts="0" DigitalInputs="0" />
        </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 new capability is defined after the device implementation.

Annex D.
Deprecated Interfaces

D.1 D.1 Remote Discovery Proxy

The definition and interfaces for the Remote Discovery Proxy have been deprecated with release 2.6.1. The following interfaces have been removed from the specification:

- Get remote discovery mode
- Set remote discovery mode
- Get remote DP addresses
- Set remote DP addresses

The definitions are available via the link http://www.onvif.org/specs/core/ONVIF-Core-Specification-v260.pdf.

D.2 D.2 Security

The definition and interfaces for the Security have been deprecated with release 16.12. The following interfaces have been removed from the specification: The Security part was handed over to Advanced Security Service.

- Create IEEE 802.1X configuration
- Set IEEE 802.1X configuration
- Get IEEE 802.1X configuration
- Get IEEE 802.1X configurations
- Delete IEEE 802.1X configuration
- Create self-signed certificate
- Get certificates
- Get CA certificates
- Get certificate status
- Set certificate status
- Get certificate request
- Get client certificate status
- Set client certificate status
- Load device certificate
- Load device certificates in conjunction with its private key
- Get certificate information
- Load CA certificates

- Delete certificate

The following GetServiceCapabilities have been deprecated:

- TLS1.0
- TLS1.1
- TLS1.2
- OnboardKeyGeneration

The definitions are available via the link http://www.onvif.org/specs/core/ONVIF-Core-Specification-v1606.pdf.

Annex E.
Revision History

Rev.	Date	Editor	Changes
2.1	Jul-2011	Hans Busch	Separated non Core services. Change Request 52, 56, 57, 58, 61, 64, 69, 88, 154, 200, 224, 235, 243, 244, 245, 246, 248, 253
2.1.1	Jan-2012	Hans Busch	Change Request 242, 263, 280, 286, 329, 335, 362, 433, 501, 512, 535, 536, 540, 555 - 562, 564, 569, 581, 587
2.2	April-2012	Hans Busch	Add Device and Service Monitoring Events. Change Request 620
2.2.1	Dec-2012	Hans Busch, Michio Hirai	Change Request 693, 707, 746, 751, 783, 785, 798, 824, 854, 843, 860, 720, 756, 874
2.3	May-2013	Michio Hirai	Event service extension (Seek, Persistent notification storage) Change Request 898, 693, 844, 793, 928, 859, 876, 887, 950, 989, 1025, 1032, 873
2.4	Aug-2013	Michio Hirai	Change Request 1055, 1056, 1057, 1089, 1143
2.4.1	Dec-2013	Michio Hirai	Change Request 1185, 1213, 1216
2.4.2	May-2014	Hans Busch, Michio Hirai	Change Request 1325, 1238, 1244, 1409, 1290,1300, 1303, 1241, 1308, 1309, 1359, 1372, 1375, 1374, 1299, 1301, 1302, 1371, 1373
2.5	Jul-2014	Hasan Timucin Ozdemir	Added 4.5.8 Storage configuration Added 8.7 Storage Configuration (includes additional configuration interface operations) Added Table 98: Device service specific fault codes (additional fault codes due to new storage configuration interfaces) Added 8.8.7 Asynchronous Operation Status (an event for reporting the progress of an asynchronous operation to the monitoring clients) Added optional FileProgressStatus element to event description Added StorageConfiguration capability in GetServiceCapabilities Change Request 1502, 1430, 1467, 1471, 1478, 1482, 1483, 1488, 1490, 1491, 1515, 1539 Change Request 1502, 1430, 1467, 1471, 1478, 1482, 1483, 1488, 1490, 1491, 1515, 1539
2.6	Jun-2015	Michio Hirai	Change Request 1581, 1582, 1585, 1586, 1587, 1588, 1662, 1661, 1663, 1666
2.6.1	Dec-2015	Michio Hirai, Hiroyuki Sano	Change Request 1672, 1677, 1721, 1729
16.06	Jun-2016	Hiroyuki Sano	Change Request 1743, 1760, 1761, 1762, 1773, 1782, 1841, 1842
16.12	Dec-2016	Ottavio Campana, Hiroyuki Sano	Added Geo Location in , - Change Request 1641, 1777, 1876, 1883, 1952
17.06	Jun-2017	Hiroyuki Sano, Enrico Campana	Change Request 2025, 2026, 2033, 2056, 2065, 2102, 2108, 2142 Add GeoLocation event
17.12	Dec-2017	Hiroyuki Sano	Change Request 2024
18.06	Jun-2018	Hiroyuki Sano	Change Request 2205, 2226, 2237, 2251, 2267, 2312

Rev.	Date	Editor	Changes
18.12	Dec-2018	Hiroyuki Sano	Change Request 2382
19.06	Jun-2019	Hiroyuki Sano	Change Request 2453
19.12	Dec-2019	Hiroyuki Sano	Change Request 2581, 2629
20.06	Jun-2020	Michio Hirai	Change Request 2558
20.06	Jun-2020	Fredrik Svensson	Add section on Event Broker
20.12	Dec-2020	Hans Busch	Update layout and consolidate framework section. Add options for network configuration and user management support.