ONVIF™
Core Specification

Version 2.0
November, 2010

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 (co Ed.)**	**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 OK**	**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**	**Kazunori Sakaki**	**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 standard is to realize a fully interoperable network video implementation comprised of products from different network video vendors. This standard describes the network video model, interfaces, data types and data exchange patterns. The standard reuses existing relevant standards where available, and introduces new specifications only where necessary to support the specific requirements for network video surveillance.

This is the ONVIF core specification. In addition, ONVIF has released the following related specifications:

- ONVIF Schema [ONVIF Schema]
- ONVIF Analytics Service WSDL [ONVIF Analytics WSDL]
- ONVIF Analytics Device Service [ONVIF AnalyticsDevice WSDL]
- ONVIF Device Service WSDL [ONVIF DM WSDL]
- ONVIF DeviceIO Service WSDL [ONVIF DeviceIO WSDL]
- ONVIF Display Service WSDL [ONVIF Display WSDL]
- ONVIF Event Service WSDL [ONVIF Event WSDL]
- ONVIF Imaging Service WSDL [ONVIF Imaging WSDL]
- ONVIF Media Service WSDL [ONVIF Media WSDL]
- ONVIF PTZ Service WSDL [ONVIF PTZ WSDL]
- ONVIF Recording Service WSDL [ONVIF Recording WSDL]
- ONVIF Remote Discovery WSDL [ONVIF DP WSDL]
- ONVIF Replay Service WSDL [ONVIF Replay WSDL]
- ONVIF Search Service WSDL [ONVIF Search 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 video IP configuration requirements.

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

Device Management: Defines the network video transmitter management commands.

DeviceIO: Defines commands to handle physical inputs and outputs

Display: Defines commands to deal with display devices

Imaging and Media: Defines the configuration commands related to imaging and media settings.

Real Time Streaming: Provides requirements for interoperable video, audio and metadata streaming.

Event Handling: Defines how to subscribe to and receive data from network video events (notifications).

PTZ Control: Provides commands for pan, tilt and zoom control.

Video Analytics: Defines the ONVIF analytics model, analytics object description and analytics rules configurations.

Video Analytics Device: Defines commands to deal with an Video Analytics Device.

Recording Control:: Defines mechanism for the configuring of recordings.

Recording Search and Replay Control: Provides commands for retrieval of recorded media including metadata.

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

1 Scope

This standard defines procedures for communication between network video clients and video transmitter devices. This new set of specifications makes it possible to build network video systems with devices and receivers from different manufacturers using common and well defined interfaces. These interfaces cover functions such as device management, real-time streaming of audio and video, event handling, Pan, Tilt and Zoom (PTZ) control, video analytics as well as control, search and replay of recordings.

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 for the introduced network video services.

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. These extensions have been introduced in order to cover the specific network video discovery needs.

This standard is not limited to discovery, configuration and control functions, but defines precise formats for media and metadata streaming in IP networks using suitable profiling of IETF standards. Furthermore, appropriate protocol extensions have been introduced in order to make it possible for network video manufacturers to offer a fully standardized network video transfer solution to its customers and integrators.

2 Normative references

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

ISO/IEC 14496-3:2005, *Information technology -- Coding of audio-visual objects -- Part 3: Audio*

ISO/IEC 14496-10:2008, *Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding*

ITU-T G.711, *Pulse code modulation (PCM) of voice frequencies*
< http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.711-198811-I!!PDF-E&type=items>

ITU-T G.726, *40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*
<http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.726-199012-I!!PDF-E&type=items>

RSA Laboratories, PKCS #10 v1.7: *Certification Request Syntax Standard, RSA Laboratories*
<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf>

FIPS 180-2, SECURE HASH STANDARD
<http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>

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

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

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

IETF RFC 2326, Real Time Streaming Protocol (RTSP)
<http://www.ietf.org/rfc/rfc2326.txt>

IETF RFC 2435, RFC2435 - RTP Payload Format for JPEG-compressed Video
<http://www.ietf.org/rfc/rfc2435.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 2782, A DNS RR for specifying the location of services (DNS SRV)
<http://www.ietf.org/rfc/rfc2782.txt>

IETF RFC 2818, HTTP over TLS
<http://www.ietf.org/rfc/rfc2818.txt>

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

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

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

IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications
<http://www.ietf.org/rfc/rfc3550.txt>

IETF RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control
<http://www.ietf.org/rfc/rfc3551.txt>

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

IETF RFC 3984, RTP Payload Format for H.264 Video
<http://www.ietf.org/rfc/rfc3984>

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 RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.1
<http://www.ietf.org/rfc/rfc4346.txt>
IETF RFC 4566, SDP: Session Description Protocol
<http://www.ietf.org/rfc/rfc4566.txt>
IETF RFC 4571, Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport
<http://www.ietf.org/rfc/rfc4571.txt>
IETF RFC 4585, Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)
<http://www.ietf.org/rfc/rfc4585.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>
IETF 5104, Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)
<http://www.ietf.org/rfc/rfc5104.txt>
IETF 5246, The Transport Layer Security (TLS) Protocol Version 1.2
<http://www.ietf.org/rfc/rfc5246.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/>
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>
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 Part 1: Structures Second Edition

<http://www.w3.org/TR/xmlschema-1/>

W3C XML Schema Part 2: Datatypes Second Edition

<http://www.w3.org/TR/xmlschema-2/>

W3C XML-binary Optimized Packaging

<http://www.w3.org/TR/2005/REC-xop10-20050125/>

IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

<http://standards.ieee.org/getieee802/download/802.11-2007.pdf>

IEEE 802.1X, Port-Based Network Access Control

<http://standards.ieee.org/getieee802/download/802.1X-2004.pdf>

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.
Configuration Entity	A network video device media abstract component that is used to produce a media stream on the network, i.e. video and/or audio stream.
Control Plane	Consists of Media control functions, such as device control, media configuration and PTZ commands.
Digital PTZ	Function that diminishes or crops an image to adjust the image position and ratio.
Imaging Service	Services for exposure time, gain and white balance parameters among others.
Infrastructure network	An IEEE 802.11 network that includes an access point, as defined in [IEEE 802.11-2007].
Input/Output (I/O)	Currently relay ports and Video/Audio Inputs/Outputs are handled.
Layout	Defines the arrangement of display areas (panes) on a monitor
Media Entity	Media configuration entity such as video source, encoder, audio source, PTZ, and analytics, for example.
Media Plane	Consists of media stream, such as video, audio and metadata.
Media Profile	Maps a video or an audio source or an audio output to a video or an audio encoder, a audio decoder configuration and PTZ and analytics configurations.
Metadata	All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications).
Network Video Transmitter (NVT)	Network video server (an IP network camera or an encoder device, for example) that sends media data over an IP network to a client.
Network Video Display (NVD)	Network video receiver (an IP network video monitor, for example) that receives media data over an IP network from e.g. an NVT.
Network Video Storage (NVS)	A device that records media and metadata received from a streaming device, such as an NVT, over an IP network to a permanent storage medium. The NVS also enables clients to review the stored data
Network Video Analytics (NVA)	A device that performs analysis on data received from a streaming device, such as an NVT, or a storage device, such as an NVS.
Optical Zoom	Changes the focal length (angle of view) for the NVT by moving the zoom lens in the camera's optics.
Pane	Defines an area on a physical display.
PKCS	Refers to a group of Public Key Cryptography Standards devised and published by RSA Security.

Pre Shared Key	A static key that is distributed to the device.
PTZ Node	Low-level PTZ entity that maps to the PTZ device and its capabilities.
PullPoint	Resource for pulling messages. By pulling messages, notifications are not blocked by firewalls.
Recording	Represents the currently stored media (if any) and metadata on the NVS from a single data source. A recording comprises one or more tracks. A recording can have more than one track of the same type e.g. two different video tracks recorded in parallel with different settings
Recording Event	An event associated with a Recording, represented by a notification message in the APIs
Recording Job	A job performs the transfer of data from a data source to a particular recording using a particular configuration
Remote Discovery Proxy (Remote DP)	The remote DP allows a NVT to register at the remote DP and at the NVC to find registered NVTs through the remote DP even if the NVC and NVT resides in different administrative network domains.
Scene Description	Metadata output by video analytics describing object location and behaviour.
Service Set ID	The identity of an [IEEE 802.11-2007] wireless network.
Track	An individual data channel consisting of video, audio, or metadata. This definition is consistent with the definition of track in [RFC 2326]
Video Analytics	Algorithms or programs used to analyze video data and to generate data describing object location and behaviour.
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

AAC	Advanced Audio Coding
ASN	Abstract Syntax Notation
AVP	Audio/Video Profile
AVPF	Audio/Video Profile for rtcp Feedback
BLC	Back Light Compensation
BSSID	Basic Service Set Identification
CA	Certificate Authority
CBC	Cipher-Block Chaining
CCMP	Counter mode with Cipher-block chaining Message authentication code Protocol
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DHT	Define Huffman Table
DM	Device Management
DNS	Domain Name Server
DQT	Define Quantization Table
DP	Discovery Proxy
DRI	Define Restart Interval
EOI	End Of Image
FOV	Field Of View
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
Ir	Infrared
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
MPEG-4	Moving Picture Experts Group - 4
MTOM	Message Transmission Optimization Mechanism
NAT	Network Address Translation

NFC	Near Field Communication
NTP	Network Time Protocol
NVA	Network Video Analytics Device
NVC	Network Video Client
NVD	Network Video Display
NVT	Network Video Transmitter
NVS	Network Video Storage Device
OASIS	Organization for the Advancement of Structured Information Standards
ONVIF	Open Network Video Interface Forum
POSIX	Portable Operating System Interface
PKCS	Public Key Cryptography Standards
PSK	Pre Shared Key
PTZ	Pan/Tilt/Zoom
REL	Rights Expression Language
RSA	Rivest ,Sharmir and Adleman
RTCP	RTP Control Protocol
RTP	Realtime Transport Protocol
RTSP	Real Time Streaming Protocol
SAML	Security Assertion Markup Language
SDP	Session Description Protocol
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
SOI	Start Of Image
SOF	Start Of Frame
SOS	Start Of Scan
SR	Sender Report
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

4 Overview

This standard is based on network video use cases covering both local and wide area network scenarios. The specification starts from a core set of interface functions for configuration and operation of network video devices by defining their server side interfaces. The set of network video devices includes Network Video Transmitter (NVT), Network Video Display (NVD), Network Video Storage (NVS) and Network Video Analytics (NVA). The framework is designed to be extended and enhanced in future versions.

The framework covers procedures from the network video device deployment and the configuration phase to the real time streaming phase for different network video scenarios.

This standard covers device discovery, device configuration, events, PTZ control, video analytics and real time streaming functionality for live video, as well as search, replay and recording management functionality for recorded video.

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.

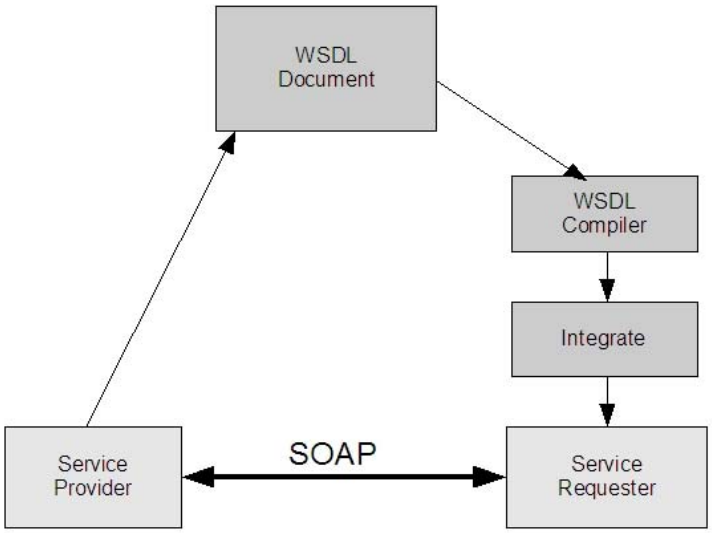


Figure 1: Web Services based development principles

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 defines the different ONVIF services, the command definition syntax in the specification, error handling principles and the adopted Web Service security mechanisms.

To ensure interoperability, all defined 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 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.

In addition to the standard web services discovery protocol this specification supports remote discovery proxies to find registered devices through the remote discovery proxy even if the client and the device reside in different administrative network domains.

4.4 Device Types

The device type signals the primary function of a device. This specification specifies the following set of device types:

- Network Video Transmitter (NVT)
- Network Video Display (NVD)
- Network Video Storage (NVS)
- Network Video Analytics (NVA)

For each device type a number of services are mandatory which are defined in section 5.1.1. A device may support other optional services and device signals availability of optional services via the device discovery.

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 as the different device services and are further divided into subcategories (when applicable) as follows:

- Analytics
- Device
    - Capabilities
    - Network
    - System
    - I/O
    - Security
- Event

- Imaging
- Media
- PTZ
- Device IO
- Display
- Recording
- Search
- Replay
- Analytics Device

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.

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.
- Handle HTTPS server certificates.
- Enable/disable HTTPS client authentication.

- Key generation and certificate download functions.
- Handle IEEE 802.1X supplicant certificate
- Handle IEEE 802.1X CA certificate
- IEEE 802.1X configuration

4.6 Device IO

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

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

- VideoOutputs
- VideoSources
- AudioOutputs
- AudioSources
- RelayOutputs

The following commands list existing interfaces:

- GetVideoOutputs – Gets all existing video outputs of the device.
- GetVideoSources – Gets all existing video sources of the device.
- GetAudioOutputs – Gets all existing audio outputs of the device.
- GetAudioSources – Gets all existing audio sources of the device
- GetRelayOutputs– Gets all existing relay outputs of the device

For VideoOutputs, VideoSources, AudioOutputs and AudioSources the following commands are supported:

- *Set<device name>Configuration* – Modifies the configuration of a specific interface.
- *Get< device name >Configuration* – Gets the configuration of a specific interface.
- *Get< device name >ConfigurationOptions* – Gets the supported property values for a specific interface.

RelayOutputs supports following commands:

- SetRelayOutputSettings – Modifies the configuration of a relay output
- SetRelayOutputState – Sets the logical state

WSDL for the DeviceIO service is specified in [DeviceIOService.wsdl].

4.7 Imaging configuration

The imaging service provides configuration and control data for imaging specific properties. WSDL is part of the framework and provided in the Imaging WSDL file.

The service includes the following operations:

- Get and set imaging configurations (exposure time, gain and white balance, for example).
- Get imaging configuration options (valid ranges for imaging parameters).
- Move focus lens.
- Stop ongoing focus movement.
- Get current position and move status for focus.

4.8 Media configuration

Media configurations are handled through the media service. Media configurations are used to determine the streaming properties of requested media streams as defined in this specification. The device provides media configuration through the media service. WSDL for the media service is provided in the Media WSDL file.

4.8.1 Media profiles

Real-time video and audio streaming configurations are controlled using media profiles. A media profile maps a video and/or audio source to a video and/or an audio encoder, PTZ and analytics configurations. The NVT presents different available profiles depending on its capabilities (the set of available profiles might change dynamically though).

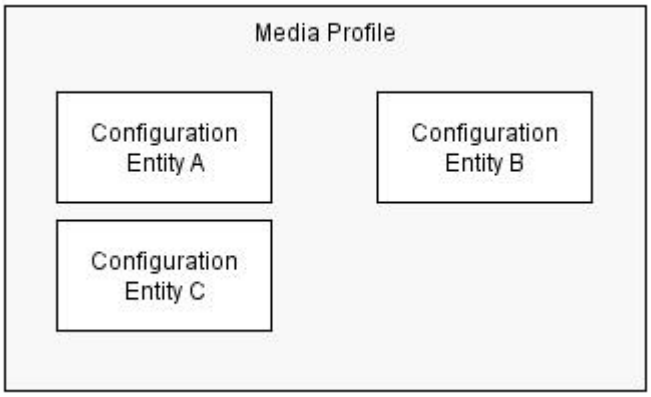


Figure 2: A media profile

A device having Media configuration service provides at least one media profile at boot. A device may provide “ready to use” profiles for the most common media configurations that the device offers.

The Profile contains a “fixed” attribute that indicates if a profile can be deleted or not. If a profile is fixed or not is defined by the NVT.

A profile consists of a set of interconnected *configuration entities.* Configurations are provided by the NVT and can be either static or created dynamically by the NVT. For example, the dynamic configurations can be created by the NVT depending on current available encoding resources. A configuration entity is one of the following:

- Video source configuration
- Audio source configuration
- Video encoder configuration
- Audio encoder configuration
- PTZ configuration
- Video analytics configuration
- Metadata configuration
- Audio output configuration
- Audio decoder configuration

A profile consists of all or a subset of these configuration entities. Depending on the capabilities of the NVT, a particular configuration entity can be part of a profile or not. For example, a profile with an audio source and an audio encoder configuration can exist only in a device with audio support.

An example of a complete profile configuration is illustrated in Figure 3.

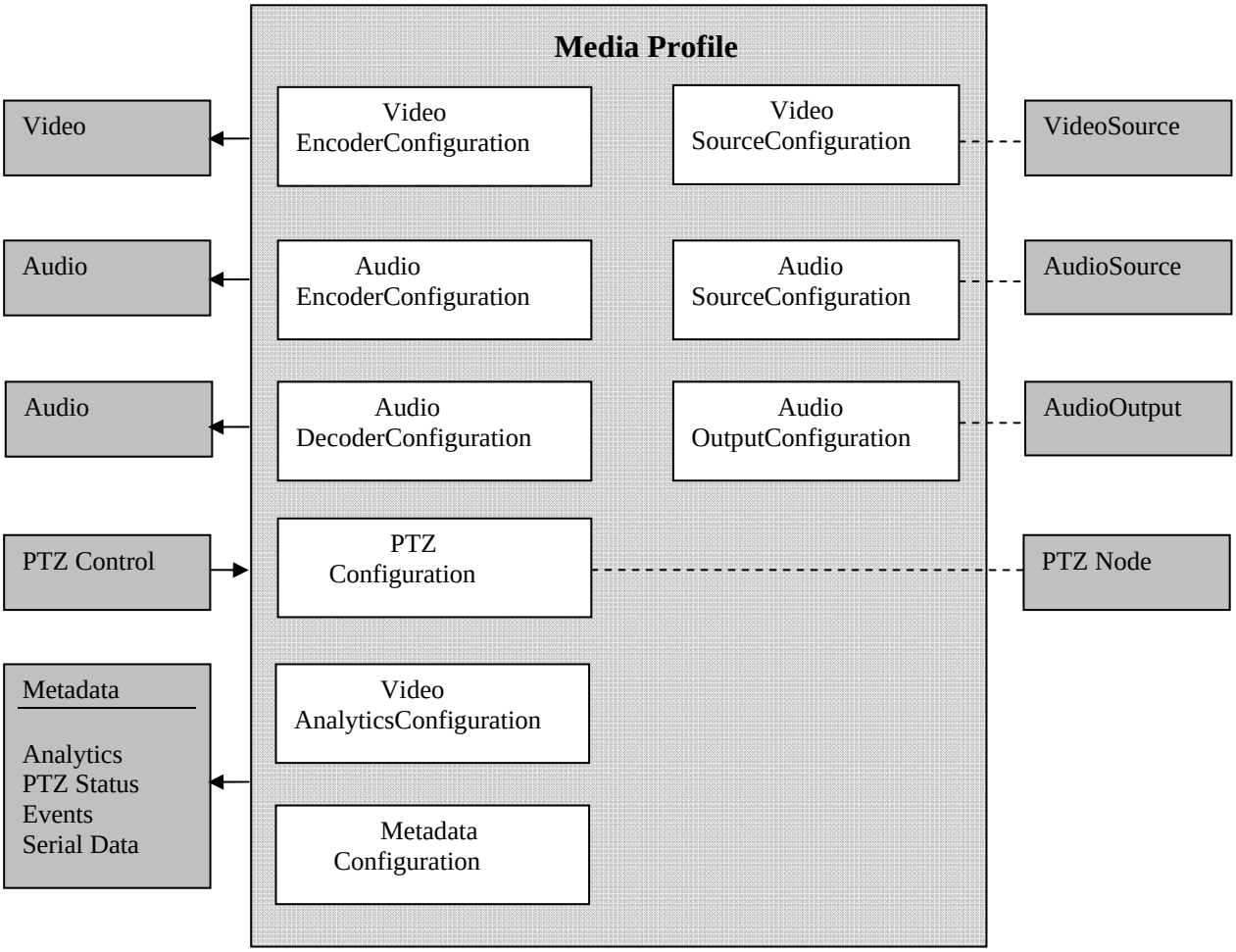


Figure 3: Complete profile configuration

A media profile describes how and what to present to the client in a media stream as well as how to handle PTZ input and Analytics.

The following commands list existing sources:

- *GetVideoSources* – Gets all existing video sources in the device.
- *GetAudioSources* – Gets all existing audio sources in the device.
- *GetAudioOutputs* – Gets all existing audio outputs in the device

The following commands manage Media Profiles:

- *CreateProfile* – Creates a new media profile.
- *GetProfiles* – Gets all existing media profiles.
- *GetProfile* – Gets a specific media profile.
- *DeleteProfile* – Deletes a specific media profile.

- *Add<configuration entity>* – Adds a specific configuration entity to the media profile.

- *Remove<configuration entity>* – Removes a specific configuration entity from a media profile.

The following commands manage Configuration Entities:

- *Get<configuration entity>Options* – Gets the valid property values for a specific configuration entity.

- *Set<configuration entity>* – Sets a configuration entity configuration.

- *Get<configuration entity>s* – Gets all existing configuration entities of the type.

- *Get<configuration entity>* – Gets a specific configuration entity.

- *GetCompatible<configuration entity>s* – Gets all configuration entities compatible with a specific media profile.

Where *<configuration entity>* is the type of configuration entity. For example, the complete command to get a video encoder configuration is:

GetVideoEncoderConfiguration

The following commands initiate and manipulate a video/audio stream:

- *GetStreamUri* – Requests a valid RTSP or HTTP stream URI for a specific media profile and protocol.

- *StartMulticastStreaming* – Starts multicast streaming using a specified media profile.

- *StopMulticastStreaming* – Stops a multicast stream.

- *SetSynchronizationPoint* – Inserts a synchronization point (I-frame etc) in active streams.

- *GetSnapshotUri* – Requests a valid HTTP URI for a specific media profile that can be used to obtain a JPEG snapshot.

Refer to 5 for examples of how the profiles are used in a client implementation.

4.9 Real-time streaming

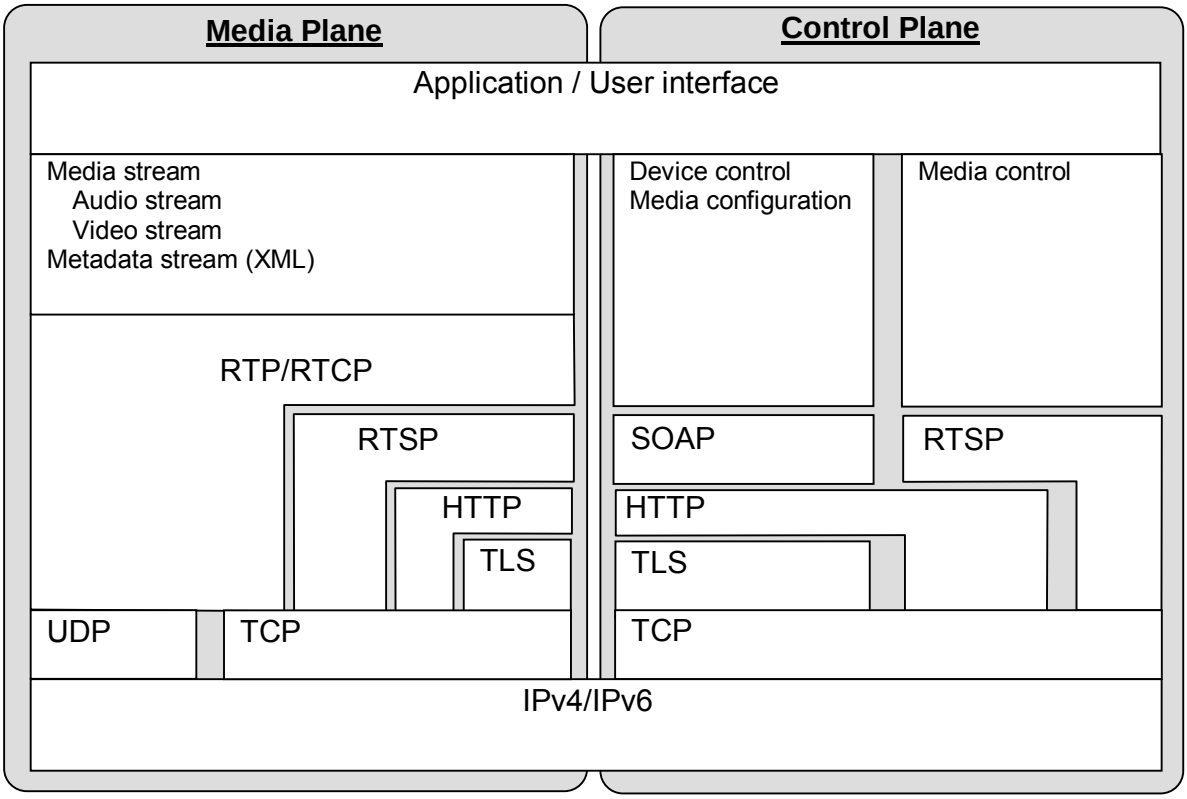


Figure 4: Layer structure

This standard defines media streaming options and formats. A distinction is made between *media plane* and *control plane,* as illustrated in Figure 4. A set of media streaming (audio, video and meta data) options, all based on RTP [RFC 3550], are described in order to provide interoperable media streaming services.

The metadata streaming container format allows well-defined, real-time streaming of analytics, PTZ status and notification data.

Media configuration is done over SOAP/HTTP and is covered by the media configuration service as discussed in Section 4.6.

Media control is accomplished over RTSP as defined in RFC 2326. This standard utilizes RTP, RTCP and RTSP profiling, as well as JPEG over RTP extensions and multicast control mechanisms.

The standard introduces extensions to the RTSP standard to allow bi-directional streaming connections.

Streaming configurations for the following video codecs are provided:

- JPEG (over RTP), see 12.1.3.

- MPEG-4, Simple Profile (SP) [ISO 14496-2]
- MPEG-4, Advanced Simple Profile (ASP) [ISO 14496-2]
- H.264, baseline [ISO 14496-10]
- H.264, main [ISO 14496-10]
- H.264, extended [ISO 14496-10]
- H.264, high [ISO 14496-10]

and for the following audio codecs:

- G.711 [ITU-T G.711]
- G.726 [ITU-T G.726]
- AAC [ISO 14496-3]

4.10 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 basic notification topics that a device is recommended to support, see Appendix A. In addition, for some services, this specification extends the basic notification topics with mandatory events.

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

4.11 PTZ control

The PTZ service is used to control a Pan Tilt and Zoom (PTZ) video encoder device. WSDL for the PTZ service is provided in the PTZ WSDL file.

The PTZ control principle follows the *MediaProfile* model (see 4.8) and consists of three major blocks:

- PTZ Node – Low-level PTZ entity that maps to the PTZ device and it’s capabilities.
- PTZ Configuration – Holds the PTZ configuration for a specific PTZ Node.
- PTZ Control Operation – PTZ, preset and status operations.

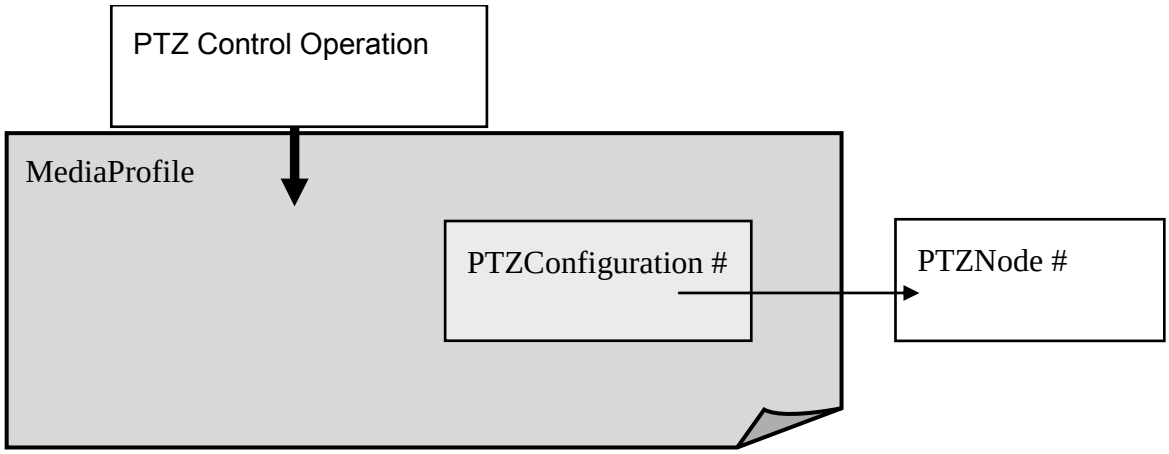


Figure 5: PTZ control model

A PTZ capable NVT may have one or many PTZ nodes. The PTZ node may be a mechanical PTZ driver, an uploaded PTZ driver on a video encoder or a digital PTZ driver. The PTZ node is the lowest level entity of the PTZ Control and it specifies the supported PTZ capabilities.

PTZ configurations are set *per media profile* and are handled through these configuration commands:

- Get and set configurations for pan, tilt and zoom.
- Get configuration options for pan, tilt and zoom.

This standard defines the following PTZ control operations:

- PTZ absolute, relative and continuous move operations.
- Stop operation.
- Get PTZ status information (position, error and move status, for example).
- Get, set, remove and move to preset position.
- Get, set and move to home position.

4.12 Video analytics

Video analytic applications are divided into image analysis and application-specific parts. The interface between these two parts produces an abstraction that describes the scene based on the objects present. Video analytic applications are reduced to a comparison of the scene descriptions and of the scene rules (such as virtual lines that are prohibited to cross, or polygons that define a protected area). Other rules may represent intra-object behaviour such as objects following other objects (to form a tailgating detection). Such rules can also be used to describe prohibited object motion, which may be used to establish a speed limit.

These two separate parts, referred to as the video analytics engine and as the rule engine, together with the events and actions, form the video analytics architecture according to this specification as illustrated in Figure 6.

The video analytics architecture consists of elements and interfaces. Each element provides a functionality corresponding to a semantically unique entity of the complete video analytics

solution. Interfaces are unidirectional and define an information entity with a unique content. Only the Interfaces are subject to this specification. Central to this architecture is the ability to distribute any elements or sets of adjacent elements to any device in the network.

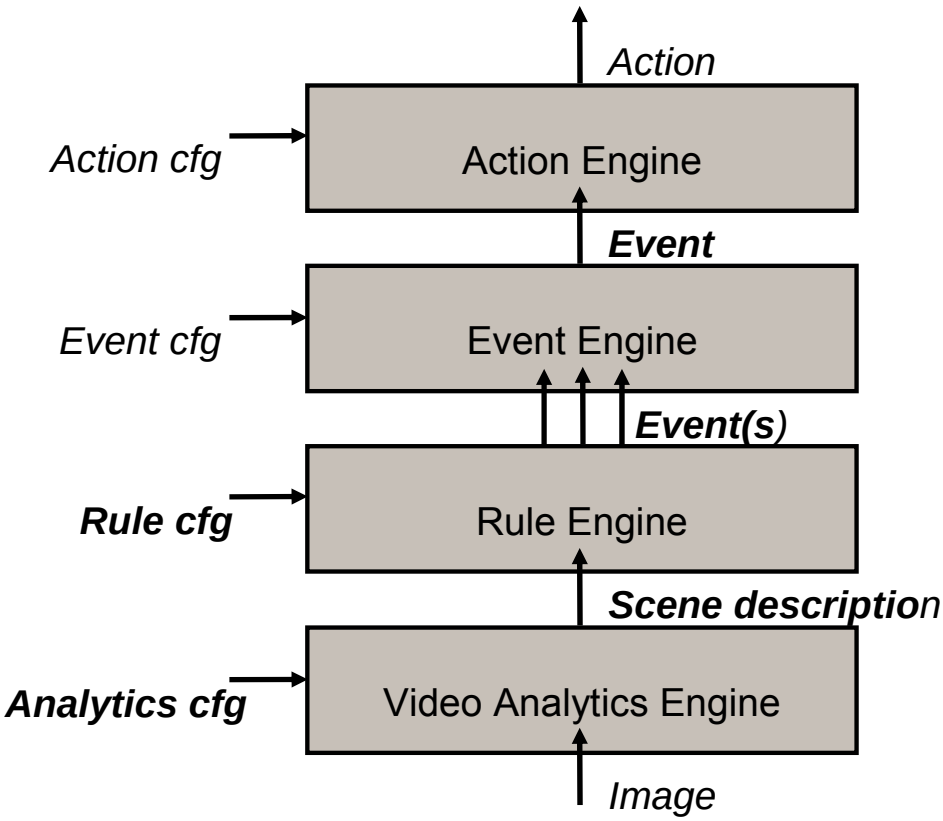


Figure 6: Video analytics architecture

The following interfaces are defined in this standard:

- Analytics Configuration Interface
- Scene Description
- Rule Configuration Interface
- Event Interface

The standard defines a configuration framework for the Video Analytics Engine. This framework enables a client to ask the device for supported analytics modules responsible for their configurations. Configurations of such modules can be dynamically added, removed or modified by a client, allowing a client to run multiple Video Analytics Modules in parallel if supported by the device.

The output from the Video Analytics Engine is called a *Scene Description*. The Scene Description represents the abstraction of the scene in terms of the objects, either static or dynamic, that are part of the scene. This specification defines an XML-based Scene Description Interface including data types and data transport mechanisms.

Rules describe how the scene description is interpreted and how to react on that information. The specification defines standard rule syntax and methods to communicate these rules from the application to the device.

An event signals the state of the analysis of the scene description and the associated rules. The event interface is both the input and the output of the event engine element. The event interface is handled through the general notification and topics framework (see 4.10).

WSDL for the video analytics service is part of the framework and provided in the Analytics WSDL file.

4.13 Analytics Device

The Analytics Device Service has to be used for stand alone analytics devices which perform evaluation processes on media streams or metadata enhanced media streams. Evaluations may involve more than one media stream or metadata enhanced media stream at a time.

The Analytics Device Service receives media streams or metadata enhanced media streams from live-generating or storing devices. It could comprise decoder capabilities if analysis is being performed on uncompressed data.

The Analytics Device Service is being used by a Client to configure properties and functionality of a stand alone analytics device. Backchannel capabilities are not provided by stand alone analytics devices.

The output of the Analytics Device Service can be obtained using the Event Service, additionally the GetStreamUri command is supported.

4.14 Display

The display service provides functions to enable a client to control and configure display devices. The service introduces panes, each of which occupies an area of the physical display. The configuration of the pane maps audio inputs and outputs to a video output. The configuration also references a Receiver Object which receives the data to be displayed. Functions to retrieve and configure the configuration of a pane are provided.

A layout defines how these panes are visible on the display (e.g single view or quad view). The service introduces commands to retrieve the current layout of a display and change the layout.

The service also introduces commands to request the encoding and decoding capabilities of a video output as well as the layout options.

4.15 Receiver

A receiver is an object that acts as an RTSP client endpoint. Receivers are used by other services that consume media streams, such as the Display, Recording and Analytics Device services. A receiver has a configuration that determines the RTSP endpoint to which it should connect and the connection parameters it should use.

A receiver can operate in three distinct modes:

Always Connect. The receiver attempts to maintain a persistent connection to the configured endpoint.

Never Connect. The receiver does not attempt to connect.

Auto Connect. The receiver connects on demand, as required by consumers of the media streams.

A single receiver may be used by more than one consumer. For example, in order to record a stream and also perform analytics on it, both a recording job and an analytics engine could be attached to the same receiver. If the receiver uses the “Auto Connect” mode, it will connect whenever either the recording job or the analytics engine is active, and disconnect when neither of them are active.

Receivers may be created and deleted either manually, by calling the CreateReceiver and DeleteReceiver operations in the Receiver Service, or automatically by other services. For example, if a recording job is created with the “AutoCreateReceiver” option, it will automatically create and attach to a Receiver. Deleting the recording job will also delete the receiver.

4.15.1 Synchronization Points

Because receivers use RTSP addresses to specify the source of the stream, they do not necessarily have access to the web services interface of the transmitter. This means that they cannot use the SetSynchronizationPoint command described in Section 11.18.1.

Instead, receivers should use the PLI message described in [RFC 4585] to request a synchronization point.

4.16 Storage

This standard provides a set of interfaces that enable the support of interoperable network storage devices, such as network video recorders (NVR), digital video recorders (DVR) and cameras with embedded storage.

The following functions are supported:

- Recording Control
- Search
- Replay

These functions are provided by three interrelated services:

Recording service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes creation and deletion of recordings and tracks.

Search service enables a client to find information about the recordings on the storage device, for example to construct a “timeline” view, and to find data of interest within a set of recordings. The latter is achieved by searching for events that are included in the metadata track recording,

Replay service enables a client to play back recorded data, including video, audio and metadata. Functions are provided to start and stop playback and to change speed and direction of the replayed stream. It also enables a client to download data from the storage device so that export functionality can be provided.

4.16.1 Storage Model

The storage interfaces in this standard present a logical view of the data on the storage device. This view is completely independent of the way data might be physically stored on disk.

The key concept in the storage model is that of a *recording*. The term *recording* is used in this specification to denote a container for a set of related audio, video and metadata *tracks*, typically from the same data source e.g. a camera. A *recording* could hold any number of tracks. A *track* is viewed as an infinite timeline that holds data at certain times.

At a minimum, a recording is capable of holding three tracks, one for audio, one for video and one for metadata. Some implementations of the recording service may support multiple tracks of each type. For example the same recording could hold two video tracks, one containing a low resolution or low frame rate stream and one containing a high resolution or high frame rate stream.

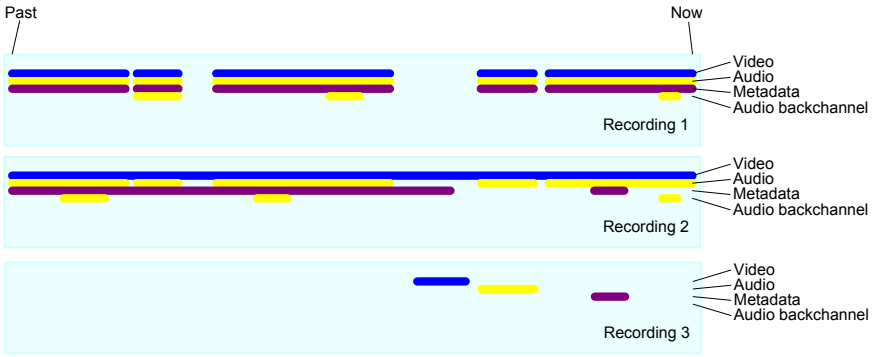


Figure 7: Storage Model with Tracks

It is important to note that the storage interfaces do not expose the internal storage structures on the device. In particular, a recording is not intended to represent a single file on disk although in many storage device implementations a recording is physically stored in a series of files. For instance, some camera implementations realise alarm recording by creating a distinct file for each alarm that occurs. Although each file could be represented as a different *recording*, the intent of the model in this standard is that all these files are aggregated into a single recording.

Within a recording the regions where data is actually recorded are represented by pairs of events, where each pair comprises an event when recording started and an event when recording stopped. A client can construct the logical view of the recordings by using the FindRecordings and FindEvents methods of the search service.

If metadata is recorded, the metadata track can hold all the events generated by the data source (see the chapter on event handling and the MetadataConfiguration object). In addition, a device also conceptually records ONVIF defined historical events (see Recording Event Descriptions in the search service), this includes information like start and end of a recorded data range. A device may also conceptually record vendor specific historical events. Events generated by the device are not inserted in existing metadata tracks of recordings. The FindEvents method in the search service can find all the recorded events.

4.16.2 Recording

The recording service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes creation and deletion of recordings and tracks.

Recording jobs transfer data from a recording source to a recording. A recording source can be a receiver object created with the receiver service, or it can be a media profile that encodes data on a local device. The media profile could be used as a source on a camera with embedded storage.

To save data to a recording, a client first creates a recording and ensures that the recording has the necessary tracks. Then the client creates a recording job that pulls data from one or more sources and stores the data to the tracks in the recording.

Clients may set up multiple recording jobs that all record into the same recording. If multiple recording jobs are active, the device uses a priority scheme to select between the tracks defined in the recording jobs. Clients may change the mode of recording jobs at any time, thereby providing means to implement features like alarm recording or manual recording.

The recording job relies on the receiver service for receiving the data from other devices through receiver objects identified by ReceiverTokens

4.16.3 Search

The search service enables a client to find information about the recordings on the storage device, for example to construct a “timeline” view, and to find data of interest within a set of recordings. The latter is achieved by searching for events and other information that is included in the metadata track recording.

The search service provides the following functionality:

- Find recordings and information about each recording
- Find events in the metadata and among the historical events
- Find PTZ positions in the metadata
- Find other information in the metadata e.g. text from EPOS (electronic point-of-sale) systems

The actual searching is done by coupled find and result operations and is asynchronous. Each find operation initiates a search session. The client can then acquire the results from the search session in increments, or all at once, depending on implementation and the scale of the search. There are four pairs of search operations for recordings, recording events, PTZ positions and metadata.

FindRecordings and GetRecordingSearchResults

FindEvents and GetEventSearchResults

FindPTZPosition and GetPTZPositionSearchResults

FindMetadata and GetMetadataSearchResults

4.16.4 Replay

The replay service provides a mechanism for replay of stored video, audio and metadata. This mechanism may also be used to download data from the storage device so that export functionality can be provided.

The replay protocol is based on RTSP [RFC 2326]. However because RTSP does not directly support all of the requirements for replay, several extensions have been added to the protocol. In particular, an RTP header extension is defined to allow an absolute timestamp to be associated with each access unit (e.g. video frame), and to convey information about stream continuity.

The GetReplayUri command in the replay service returns the RTSP URL of a recording to allow it to be replayed using RTSP.

4.17 Security

This clause describes network video security requirements. This specification defines security mechanism on two different communication levels:

- Transport-level security
- Message-level security

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

- IEEE 802.1X

The general security requirements, definitions and transport security requirements are specified in 22. Message level security requirements are specified in 5.12. IEEE 802.1X requirements are specified in Section 8.4.7 Security management is handled through the device management service as listed above in 4.5.7.

5 Web Services framework

All management and configuration commands are based on Web Services.

For the purpose of this standard:

- The device (NVT, NVD, NVS, NVA) is a service provider.
- The client is a service requester.

A typical network video system does not have a single client that handles all device configuration and device management operations for one device. Instead, a distinction between *control* and network video *receiver* functionality may exist. A device providing services may also act as a client. Future versions of the specification may introduce additional entities and interfaces in the system.

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 and guidelines to create interoperable Web Services. The devices and the clients shall follow the guidelines in the WS-I Basic Profile 2.0 [WS-I BP 2.0]. The service descriptions in this specification follow the WS-I Basic Profile 2.0 recommendations.

5.1 Services overview

An ONVIF compliant device shall support a number of Web Services which are defined in this specification. Examples for web services defined by this specification are:

- device service
- media service
- event service

The device service is the *target service* of an ONVIF compliant device and the *entry point* for all other services of the device.

The entry point for the device management is fixed to:

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

5.1.1 Services requirements

A device shall provide the device management and event service. A device MAY support any of the other services depending on the capabilities of the device. Depending on the device type (NVT, NVD, NVS, NVA) additional services are required. The exact compliance requirements are defined as part of the different service definitions in this specification.

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

env:Receiver,

ter:ActionNotSupported,

see 5.11.2 for the definitions of the error codes.

Table 1: Service requirements for the device types

	NVT	NVS	NVD	NVA
Device	M	M	M	M
Event	M	M	M	M
Media	M			
PTZ	C			
Imaging				
Analytics				M
Recording Control		C		
Recording Search		M		
Replay Control		M		
Device IO	M		M	
Receiver		C	M	M
Display			M	
Analytics Device				M

Table 1 shows which services are required for the different device types. Mandatory services are marked with ‘M’ and services that are mandatory if a related feature is supported by the device are marked with ‘C’.

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.
- port – Specifies an address for a binding.
- service – Used to group a set of related ports.

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 2: 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.
timg	http://www.onvif.org/ver20/imaging/wsdl	The namespace for the WSDL imaging service.
tev	http://www.onvif.org/ver10/events/wsdl	The namespace for the WSDL event service.
tptz	http://www.onvif.org/ver20/ptz/wsdl	The namespace for the ptz control service.
tan	http://www.onvif.org/ver20/analytics/wsdl	The namespace for the analytics 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
tad	http://www.onvif.org/ ver10/analyticsdevice/wsdl	The namespace for the WSDL analytics device service.
tmd	http://www.onvif.org/ ver10/deviceIO/wsdl	The namespace for the WSDL deviceIO service.
tls	http://www.onvif.org/ ver10/display/wsdl	The namespace for the WSDL display service

trv	http://www.onvif.org/ ver10/receiver/wsdl	The namespace for the WSDL receiver service.
trc	http://www.onvif.org/ ver10/recording/wsdl	The namespace for the WSDL recording service.
trp	http://www.onvif.org/ ver10/replay/wsdl	The namespace for the WSDL replay service.
tse	http://www.onvif.org/ ver10/search/wsdl	The namespace for the WSDL search service

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

Table 3: 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].
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 4: 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.
http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace	The ONVIF standard zoom position space for PTZ control.
http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace	The ONVIF standard pan/tilt

	position space for PTZ control.
http://www.onvif.org/ver10/tptz/ZoomSpaces/TranslationGenericSpace	The ONVIF standard zoom translation space for PTZ control.
http://www.onvif.org/ver10/tptz/PanTiltSpaces/TranslationGenericSpace	The ONVIF standard pan/tilt translation space for PTZ control.
http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace	The ONVIF standard zoom velocity space for PTZ control.
http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace	The ONVIF standard pan/tilt velocity space for PTZ control.
http://www.onvif.org/ver10/tptz/ZoomSpaces/SpeedGenericSpace	The ONVIF standard zoom speed space for PTZ control.
http://www.onvif.org/ver10/tptz/PanTiltSpaces/SpeedGenericSpace	The ONVIF standard pan/tilt speed space for PTZ control.

5.4 Types

Data types are defined using XML schema descriptions Part1 and Part 2. All 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`

5.5 Messages

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

A message in this specification contains two main elements:

- message name
- message parts

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

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

The following WSDL notation is used for the messages in this standard:

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

respective,

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

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

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

5.6 Operations

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

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

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

The operation name defines the name of the operation.

Operations in the specification are defined using the following table format outlined in Table 5.

Table 5: Operation description outline used in this specification

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

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

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

The fault codes listed in the tables are the *specific fault* codes that can be expected from the command, see 5.11.2.2. *Any command can return a generic fault,* see 5.11.2.2.

5.6.1 One-way operation type

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

This operation type is defined by a single input message.

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

Operation_Name	One-way
Message name	**Description**
'Operation_Name'Request	*Description of the request message.* *Type*$_1$ **Name$_1$** [a$_1$][b$_1$] *Type*$_2$ **Name$_2$** [a$_2$][b$_2$] : *Type*$_n$ **Name$_n$** [a$_n$][b$_n$]

This table corresponds to the following WSDL notation in this specification:

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

5.6.2 Request-response operation type

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

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

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

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

Fault codes	Description
Code *Subcode* *Subcode*	*Description of the operation specific fault.*

This table corresponds to the following WSDL notation:

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

5.7 Port Types

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

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

5.8 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 a 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. A device shall use the style 'document' specified at the operation level.

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

5.9 Ports

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

This specification does not mandate any port naming principles.

5.10 Services

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

5.11 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

5.11.1 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 the clients shall use appropriate RTSP and HTTP protocol defined status codes for error reporting and when received handle accordingly.

5.11.2 SOAP errors

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

This standard uses a separate name space for specific faults (see 5.11.2.2):

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.

5.11.2.1 Generic faults

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

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

Table 6: 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 recover by itself and needs reset or power cycle.

5.11.2.2 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 in this standard.

5.11.2.3 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 7: 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.

5.12 Security

The services defined in this standard shall be protected using the WS-Security framework.The WS-Security specification defines a standard set of SOAP extensions that can be used to

provide Web Services message integrity and confidentiality. The framework allows several different security models using tokens. The following tokens are currently defined:

- User name token profile [WS-UsernameToken]
- X.509 security token profile [WS-X.509Token]
- SAML token profile [WS-SAMLToken]
- Kerberos token profile [WS-KerberosToken]
- Rights Expression Language (REL) Token Profile [WS-RELToken]

Server and client shall support the user name token profile as specified in WS-Security and 5.12.2 and MAY support any of the other WS-security defined profiles.

The user name token profile *gives only a rudimentary* level of security. In a system where security is important, it is recommended to always configure the device for TLS-based access (see 22.1). The user name token message level security combined with TLS, with client and server authentication, protected transport level security gives an acceptable level of security in many systems.

A ONVIF compliant device shall when authenticating in RTSP and HTTP use credentials from the same set of credentials that are defined for the web service part. For user defined with the user name token profile, digest authentication [RFC 2617] shall be used for RTSP and HTTP.

An ONVIF compliant device should authenticate a WS request at the WS level, HTTP should only be used as a transport protocol and the device shall not authenticate a WS request at this level.

An ONVIF compliant device should authenticate a RTSP request at the RTSP level, if HTTP is used to tunnel the RTSP request the device shall not authenticate at this level.

An ONVIF compliant device shall when authentication RTSP and HTTP methods use user / credentials from the same set of users / credentials that are used for the WS part. For user defined with the user name token profile, digest authentication [RFC 2617] shall be used for RTSP and HTTP.

5.12.1 User-based access control

The WS-Security framework allows protection and authentication on the SOAP message level. These authentication mechanisms are used to build an access security policy for an ONVIF service. This specification allows security policy configuration based on four different user levels:

1. Administrator
2. Operator
3. Media user
4. Anonymous

A detailed access policy for different user classes can be defined using these categories. Unauthenticated users are placed into the anonymous category and a device shall not allow users to be added to the anonymous user level category.

It shall be possible to define the *exact access security policy* by the device user or by a system administrator. The exact format of the policy configuration file is *outside the scope* of this specification.

Commands to get and set an access security policy in arbitrary format are defined in 8.4.

5.12.2 User token profile

The only mandatory WS-security token profile is the user token profile [WS-UsernameToken].

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 standard defines a set of command for managing the user name token profile credentials, see 8.4. These commands allow associating users with the different user levels defined in 5.12.1.

5.12.2.1 Password derivation

The use of the same credentials on several devices introduces a certain security risk. To require the user to supply a unique credential for each device is not feasible, instead a client should implement the following password derivation algorithm.

Denote by UA an arbitrary user. Denote by P_UA the password value used by user UA to access the devices in the system. Furthermore, denote, by NEP, the end device service point reference value for a particular device in the system. Finally, denote by PE_UA the password equivalent used by the client to access a particular device in the system. The client should calculate the PE_UA as follows:

PE_UA = base64(HMAC_SHA-1(UA + P_UA,NEP+”ONVIF password”)),

where “+” denotes concatenation and where the “ONVIF password” is an ASCII string. It should be included in the exact form it is given without a length byte or trailing null character, i.e., the following hexadecimal value: 4F 4E 56 49 46 20 70 61 73 73 77 6F 72 64.

HMAC_SHA-1 is the algorithm specified in [RFC 2104] using SHA-1 [FIPS 180-2] as the underlying algorithm. The key value to use for the HMAC function is the user password, P_UA, directly mapped to its binary equivalent. Similar, the value PE_UA should be mapped to its ASCII equivalent before transmitting it to the device.

base64 is described in [RFC 3548], note that the result of the base64 operation is the actual password equivalent and shall be used as it is.

5.12.2.1.1 Example

Assume the following password and password is used by the client (ASCII): “user” and “VRxuNzpqrX”, i.e.,

UA = 75 73 65 72

P_UA = 56 52 78 75 4E 7A 70 71 72 58

Next, assume the device has the following device service end point reference value:

Urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6.

Then the password equivalent to be used will be then calculated as:

PE_UA = base64(HMAC_SHA-1(P_UA,NEP+”ONVIF password”)) =

base64(HMAC_SHA-1(75736572565278754E7A70717258,

F81D4fAE7DEC11D0A76500A0C91E6BF6+4F4E5649462070617373776F7264)) =

base64(16 E5 C5 A9 4D DE 8A 97 6D D7 2F 55 78 5F C2 D0 6B DA 53 4A)=

FuXFqU3eipdt1y9VeF/C0GvaU0o=

The resulting password equivalence “FuXFqU3eipdt1y9VeF/C0GvaU0o=” is the password that shall be used by a client both for configuring the user credential on the particular device and then also for accessing the device.

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.

The 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.

The 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.

The 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 according to [RFC3315].

The device MAY support any additional IP configuration mechanism.

Network configuration of a device is accomplished through an IP network interface or through a local interface such as USB, serial port, Bluetooth or NFC. IP configuration through a local interface is *outside* the scope of this specification. It shall be possible to make device IP configurations through the parameter configuration interface specified in section 8.2. A device user can enable or disable any of the IP address configuration options according to this specification through a network configuration interface. The default device configuration shall be to 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].

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].

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

The Discovery Proxy role *as described in* [WS-Discovery] shall NOT be supported by a device or a client (an alternative Discovery Proxy role is introduced in this specification, see Section 7.4). A device that implements the client role ignores the interaction scheme with the Discovery Proxy as described in Section 3 in [WS-Discovery]. Instead, this specification defines a new Discovery Proxy role that allows remote discovery. The remote discovery relies on the presence of a Discovery Proxy and a system provider that would like to offer remote discovery in the system should implement the Discovery Proxy role as specified in Section 7.4

[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 Service addresses

The <d:XAddrs> element to be included into the Hello message shall be the device service address or addresses.

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

7.3.3 Hello

7.3.3.1 Types

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

For backward compatibility reason an ONVIF compliant device shall also include dn:NetworkVideoTransmitter 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.

7.3.3.2 Scopes

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

The device scope is set by using [RFC 3986] URIs. This 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>`

The device MAY have other scope URIs. These URIs are not restricted of ONVIF defined scopes.

Table 8 defines the basic capabilities and other properties of the device. 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. Future versions of the specification might introduce additional standardized scope parameters.

A device MAY have other scope URIs. These URIs are not restricted of ONVIF defined scopes.

Table 8: Scope parameters

Category	Defined values	Description
type	video_encoder	A video_encoder indicates that this device is a network video encoder device. A device with network video support, shall include the video_encoder type in its scope list.
	Ptz	A ptz scope indicates that the device is a ptz device. A device with PTZ support shall include a scope entry with this value in its scope list.
	audio_encoder	The audio_encoder scope indicates that this device is an audio encoder and a device with audio encoder support shall include a scope entry with this value in its scope list.
	video_analytics	The video analytics scope indicates that this device supports video analytics as defined in Section17. A device with video analytics support shall include a scope entry with this value in its scope list.
	Network_Video_Transmitter	The network video transmitter scope indicates if the device is an NVT compliant device. An NVT shall include a scope entry with this value in its scope list.
	Network_Video_Decoder	The network video display scope indicates if the device is an NVD compliant device. An NVD shall include a scope entry with this value in its scope list.
	Network_Video_Storage	The network video storage scope indicates if the device is an NVS compliant device. An NVS shall include a scope entry with this value in its scope list.
	Network_Video_Analytic	The network video analytic scope indicates if the device is an NVA compliant device. An NVA shall include a scope entry with this value in its scope list
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. A device shall include at least one location entry into its scope list.
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 entry of the type, location, 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.

7.3.3.2.1 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 an NVT configuration. In this example we assume that the NVT is configured with the following scopes:

```
onvif://www.onvif.org/type/Network_Video_Transmitter
onvif://www.onvif.org/type/video_encoder
onvif://www.onvif.org/type/ptz
onvif://www.onvif.org/type/audio_encoder
```

```
onvif://www.onvif.org/type/video_analytics
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.

7.3.3.3 Addresses

A device MAY include the <d:XAddrs> element with the address(es) for the device service in the Hello message.

The IP addressing configuration principles for a device are defined in 5.12.2.1.1.

7.3.4 Probe and Probe Match

For the device probe match types, scopes and addresses definitions, see 7.3.3 Hello.

The 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 management and configuration interfaces as defined in 5.1.

7.3.5 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.

7.3.6 Bye

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

7.3.7 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>
```

All faults arising in an extension or from the application should be generated according to SOAP 1.2 fault message protocols. After transmission of a SOAP fault message to the Sender, the fault should be notified to the application that a fault has been generated.

7.4 Remote discovery extensions

This section describes discovery extensions needed to cover more complex network scenarios. These extensions *are not* required by an ONVIF-compliant endpoints. A device that supports remote service discovery shall support the discovery extensions defined in this section.

The remote discovery extensions defined in this section can be used *together* with the ordinary multicast base WS-Discovery scheme as defined in this specification. For example, the remote discovery extensions can work in parallel with the ordinary “local” discovery.

7.4.1 Network scenarios

If the client and the device *do not* reside in the same administrative domain, it is not possible for the client to find *and* connect to the device using multicast probe. For example, if the device or the client resides in a network behind a firewall or NAT (Gateway GW) it could not connect to a multicast probe. Other methods, then, are needed and the specification uses four different scenarios:

1. The device resides in one administrative domain (private) and the client resides in a public network, see Figure 8.

2. The device resides in a public network and the client resides in one administrative domain (private), see Figure 9.

3. The device resides in one administrative domain (private) and the client resides in *another* administrative domain (private), see Figure 10.

4. Both the device and the client reside in a public network, see Figure 11.

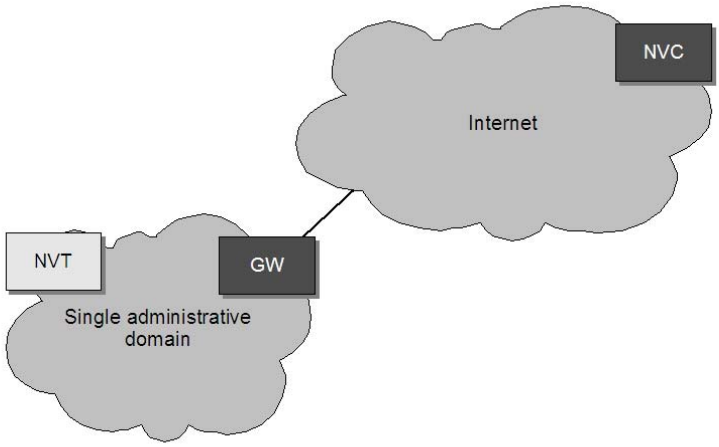


Figure 8: A device, for example a NVT, in an administrative domain (private) and the client (NVC) in a public network

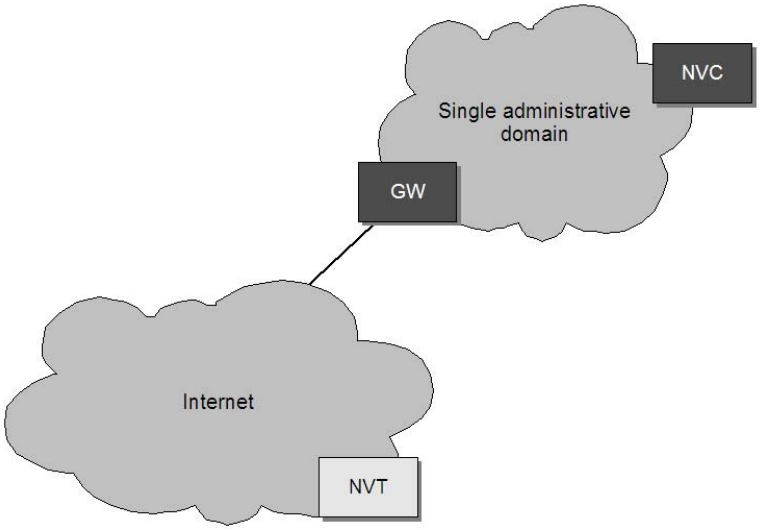


Figure 9: A device, for example a NVT, in public network and the client (NVC) in an adminstrative domain (private)

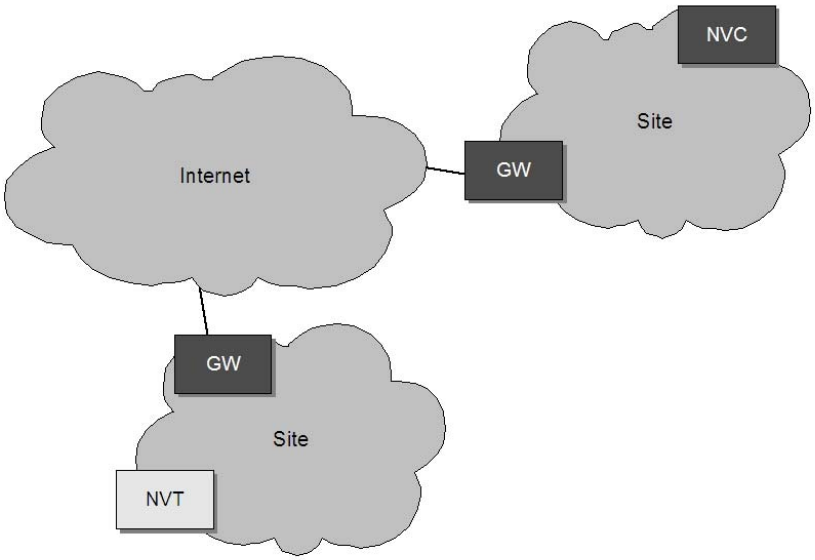


Figure 10: A device, for example a NVT, in an administrative domain (private) and the client (NVC) in another administrative domain (private)

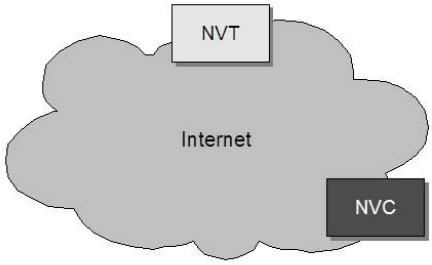


Figure 11: Both a device, for example a NVT, and the client (NVC) in a public network.

The [WS-Discovery] specification introduces a Discovery *Proxy* (DP*) to solve some of these scenarios.* However the [WS-Discovery] specification does not have support for all the network scenarios introduced in this specification. This specification defines a DP that enables “plug and play” also for the more complex network scenarios we have listed above. This DP *is not* compliant with [WS-Discovery] specification.

7.4.2 Discover proxy

A network administrator configuring a network for an NVT wide area network spanning several administrative domains, needs to introduce a DP endpoint into the system. The DP performs the following tasks:

1. Listen for device hello messages and responds to these as defined in Section 7.4.3.

2. Responds to probe queries on behalf of registered devices from clients.

The DP may reside in the same administrative domain as the device. In order to support network scenarios where the client and device reside in different domains without multicast connectivity, place the DP in a publicly available network so that device and client endpoints can access it. It shall be possible for the device to find the network address of its “home DP” in order to allow the announcement of its presence with a Hello message *directly* sent to its home DP. According to this specification, the home DP network address can be obtained in the following ways:

1. Direct address configuration.

2. DP discovery using DNS Service record (SRV) lookup.

The device tries to connect to a home DP once it gets network connectivity or when the home DP network address is changed using either of these methods.

It shall be possible to enable/disable the device remote discovery registration. A device supporting remote discovery shall implement the remote Hello disable/enable operation as defined in Section 8.3.21.

A device that is not configured with a home DP address or a device with remote Hello disabled shall NOT send a remote Hello as defined in Section 7.4.3.

7.4.2.1 Direct DP address configuration

This specification introduces a device management command for home DP address configuration over the network interface, see Section 8.3.22 and Section 8.3.23.

A device that supports remote discovery MAY also offer local configuration of the home DP address. Such configurations are done through a device local interface of choice such as a serial port or USB interface. Such local configuration is *outside* the scope of this specification.

7.4.2.2 DNS service record lookup

If a device has remote discovery enabled but *lacks* remote DP address configuration, it shall try to make a DNS SRV lookup for the home DP. The following record name and protocol definition [RFC2782] shall be used:

_onvifdiscover._tcp

In order to avoid a DNS SRV lookup by the device, a DP address shall be configured using direct address configuration before enabling remote discovery.

In order for devices to make a successful DP lookup for other devices, an administrator shall enter the DP address, port and priority into the DNS using SRVs. One or several enrolment servers need to be present. The exact number will depend on the load of the system and is *outside the scope* of this specification.

7.4.3 Remote Hello and Probe behaviour

The local discovery pattern as defined in [WS-Discovery] does not work for the remote discovery scenarios. If the device resides behind a NAT/Firewall, like the scenarios shown in Figure 8 or Figure 10, a unicast Probe from the DP will not automatically reach the device if the device does not return a public network address. Furthermore, if the device resides behind a firewall, the device following Probe Match unicast might not reach back to the DP. The specification defines a slightly different communication pattern for the remote discovery to solve this problem.

A device configured for remote Hello sends, in addition to the multicast Hello when it joins a network or its metadata changes, a remote Hello message to its home DP. This message is sent as a Web Services request operation from the device to the DP using the HTTP binding as defined in [ONVIF DP WSDL]. The remote Hello shall include its scope list in the Hello message.

Once the home DP receives a Hello message from any device, it responds with a Hello response message confirming the device registration through the hello message.

Similarly, when a device prepares for leaving a network it should send a Bye request to the remote DP. The DP acknowledges the Bye request through a Bye response message.

The DP Hello, Hello response, Bye and Bye response are provided as a DP service, see [ONVIF DP WSDL] for the WSDL definitions.

Using these extensions, the discovery messages can reach the desired endpoints.

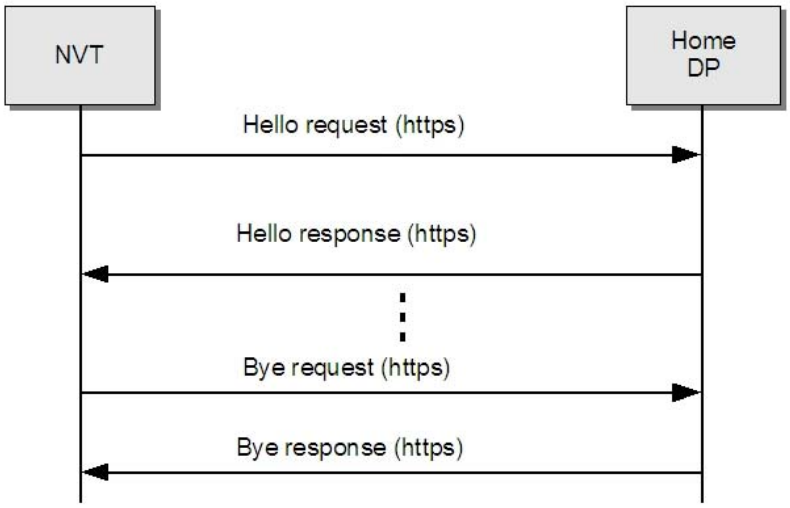


Figure 12: Remote discovery message exchange pattern between a device (for example a NVT) and a HomeDP

7.4.4 Client behaviour

For the remote discovery scenarios, the client needs to send probe messages to the home DP. The client then needs to be configured such that it can directly connect to the home DP.

7.4.4.1 NVC home DP configuration

The client can be configured to directly probe for new devices through the home DP. In this case the home DP discovery service address shall be pre-configured into the client. The exact means of this configuration is outside the scope of this specification.

An client configured for remote discovery sends probe requests directly to its home DP. The probe message is sent as a Web Services request operation from the client to the DP using the http binding (see [ONVIF DP WSDL]).

Once the home DP receives a Probe message from any client, it responses with corresponding Probe Match message according to the normal WS-Discovery message exchange pattern, see the sequence chart in Figure 13.

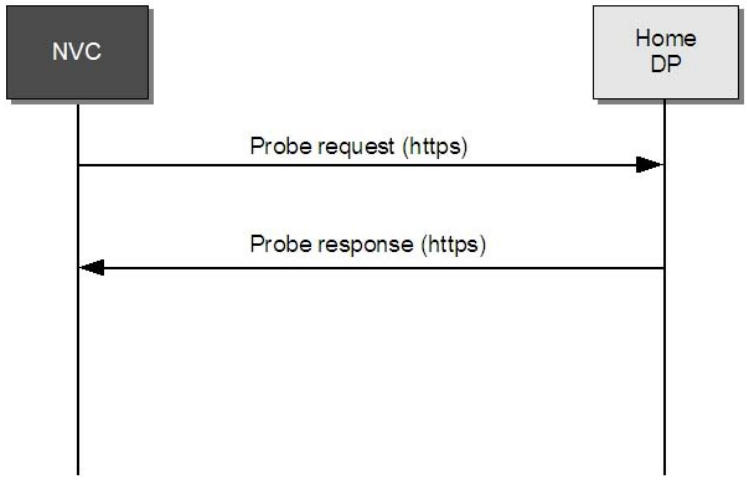


Figure 13: Message sequence for clients (NVC) pre-configured with home DP address

7.4.5 Security

7.4.5.1 Local discovery

Security and discovery can be viewed as contradictory goals. While the main idea behind a discovery protocol is to announce the presence of a service, it is hard to *exclude* other endpoints from access to the service announcements. WS-Discovery does not provide any extra access to services (if the other security mechanism specified in this specification are used), even on the same LAN; it merely announces their existence. Furthermore, local discovery works only within multicast reach range. Thus, the main security impact of WS-Discovery is the risk of denial of service attacks on devices or privacy issues if it is important to hide the presence of devices in the network. The risk of the latter two problems will very much depend on the device deployment environment. In order to reduce these threats, this specification has introduced the two different discovery modes, see Section 7.2. This always gives the possibility for the client to switch off the device discovery function in the device. In non-discoverable mode, a device will never announce its presence with Hello messages or respond to any Probe or Resolve requests.

7.4.5.2 Remote discovery

In the remote network scenario, the DP resides on the Internet and is vulnerable. Extra security measurements, then, shall be taken to protect the DP from attacks. The remote Hello and Probe and Probe Match messages, as defined in Section 7.4.3, shall be sent over HTTPS. This transport will *not* prevent denial of service attacks, but it can protect it from illegal device registrations if client authentication is used. If protection of denial of service is a major concern, other measurements need to be taken, which is outside the scope of the current specification.

Before registering a device in the device data base the DP should authenticate the device to make sure that it is a “legal” device that announces its presence, for example by using client certificates. Client certificate provisioning is outside the scope of the current specification.

The client to DP remote Probe and Probe Match messages shall be sent over HTTPS. The DP shall authenticate the NVC before responding to a Probe request. This can be done using TLS client certificates or any other suitable client authentication mechanism.

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. A 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 Get WSDL URL

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

Table 9: Get WSDL URL command

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

8.1.2 Capability exchange

Any endpoint can ask for the capabilities of a device using the capability exchange request response operation. The device shall indicate all its ONVIF compliant capabilities through the GetCapabilities command.

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

Table 11 describes how to interpret the indicated capability. Apart from the addresses, the capabilities only reflect optional functions in this specification.

Table 10: Get Capabilities command

GetCapabilities	Request-Response
Message name	**Description**
GetCapabilitiesRequest	*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.* tt:CapabilityCategory **Category** [0][unbounded]
GetCapabilitiesResponse	*The capability response message contains the requested device capabilities using a hierarchical XML capability structure.* tt:Capabilities **Capabilities** [1][1]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:NoSuchService	*The requested WSDL service category is not supported by the device.*

Table 11: The capabilities in the GetCapabilities command

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 Section 17.2.
	AnalyticsModuleSupport	Indication if the device supports the scene analytics module interface as specified in Section 17.3.
Device	XAddr	The address to the device service.
Device – Network	IPFilter	Indication if the device supports IP filtering control using the commands in Section 8.2.17, 8.2.18, 8.2.19 and 8.2.20.
	ZeroConfiguration	Indication if the device supports zero configuration according to the commands in Section 8.2.15 and Section 8.2.16.

	IPVersion6	Indication if the device supports IP version 6.
	DynDNS	Indication if the device supports Dynamic DNS configuration according to Section 8.2.7and Section 8.2.8 .
	Dot11Configuration	Indication if the device supports IEEE802.11 configuration as specified in Section 8.2.21
Device – System	DiscoveryResolve	Indication if the device responses to resolve requests as described in Section **7.3.5**.
	DiscoveryBye	Indication if the device sends bye messages as described in Section **7.3.6**
	RemoteDiscovery	Indication if the device supports remote discovery support as specified in Section 7.4.
	SupportedVersions	List of the device supported ONVIF specification versions.
	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 8.3.10.
	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.
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 as specified in Section 8.4.8.
	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].
	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
	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 8.4.21 and 8.4.22.
Event	XAddr	The address to the event service
	WSSubscriptionPolicySupport	Indication if the device supports the WS Subscription policy according to Section15.1.2
	WSPullPointSupport	Indication if the device supports the WS Pull Point according to Section 15.1.2
	WSPausableSubscription-ManagerInterfaceSupport	Indication if the device supports the WS Pausable Subscription Manager Interface according to Section15.1.2
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 Section 12.1.1.1.
	RTP_TCP	Indication if the device supports RTP over TCP, see Section 12.1.1.2.
	RTP_RTSP_TCP	Indication if the device supports RTP/RTSP/TCP transport, see Section 12.1.1.3.
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
	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 section 19.4 and 19.5.
	DynamicTracks	Indication if the device supports dynamic creation and deletion of tracks, see section 19.9 and 19.10.
	DeleteData	Indication if the device supports explicit deletion of data, see section **Fehler! Verweisquelle konnte nicht gefunden werden.**.
Search	XAddr	The address to the recording search service.
	MetadataSearch	Indication if the device supports generic search of recorded metadata, see section 20.13 and 20.14.
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 if the device has a certain set of 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.

8.2 Network

8.2.1 Get hostname

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

Table 12: GetHostname command

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

8.2.2 Set hostname

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

Table 13: SetHostname command

SetHostname	Request-Response
Message name	**Description**
SetHostnameRequest	*This message contains:* • *"Name": The host name.* xs:token **Name** [1][1]

SetHostnameResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidHostname	*The requested hostname cannot be accepted by the device.*

8.2.3 Get DNS settings

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

Table 14: GetDNS command

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

8.2.4 Set DNS settings

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

Table 15: Set DNS command

SetDNS	Request-Response
Message name	**Description**
SetDNSRequest	*This message contains:* • *"FromDHCP": True if the DNS servers are obtained via DHCP* • *"SearchDomain": The domain(s) to search if the hostname is not fully qualified.* • *"DNSManual": A list of manually given DNS servers* xs:boolean **FromDHCP** [1][1] xs:token **SearchDomain** [0][unbounded]

	tt:IPAddress **DNSManual** [0][unbounded]
SetDNSResponse	***This is an empty message***.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*

8.2.5 Get NTP settings

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.

Table 16: GetNTP command

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

8.2.6 Set NTP settings

This operation sets the NTP settings on a device. If the device supports NTP, it shall be possible to set the NTP server settings through the SetNTP command.

Table 17: SetNTP command

SetNTP	Request-Response
Message name	**Description**
SetNTPRequest	*This message contains:* • *"FromDHCP": True if the NTP servers are obtained via DHCP.* • *"NTPManual": A list of manually given NTP servers when they not are obtained via DHCP.*

	xs:boolean **FromDHCP** [1][1] tt:NetworkHost **NTPManual** [0][unbounded]
SetNTPResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidDnsName	*The suggested NTP server name is invalid.*

8.2.7 Get dynamic DNS settings

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.

Table 18: GetDynamicDNS command

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

8.2.8 Set dynamic DNS settings

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.

Table 19: SetDynamicDNS command

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

8.2.9 Get network interface configuration

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.

Table 20: GetNetworkInterfaces command

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

8.2.10 Set network interface configuration

This operation sets the network interface configuration on a device. The device shall support network configuration of supported network interfaces through the SetNetworkInterfaces 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.

Table 21: SetNetworkInterfaces command

SetNetworkInterfaces	Request-Response
Message name	**Description**
SetNetworkInterfacesRequest	*This message contains:* • *"InterfaceToken": The token of the network interface to operate on.* • *"NetworkInterface": The network interface to configure.* tt:ReferenceToken **InterfaceToken** [1][1] tt:NetworkInterfaceSetConfiguration **NetworkInterface** [1][1]
SetNetworkInterfacesResponse	*This message contains:* • *"RebootNeeded": An indication if a reboot is needed in case of changes in the network settings.* xs:boolean RebootNeeded [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidMtuValue	*The MTU value is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceSpeed	*The suggested speed is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidInterfaceType	*The suggested network interface type is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 Configuration is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidSecurityMode	*The selected security mode is not supported.*
env:Sender ter:InvalidArgVal ter:InvalidStationMode	*The selected station mode is not supported.*

env:Sender ter:InvalidArgVal ter:MissingDot11	*IEEE 802.11 value is missing in the security configuration.*
env:Sender ter:InvalidArgVal ter:MissingPSK	*PSK value is missing in security configuration.*
env:Sender ter:InvalidArgVal ter:MissingDot1X	*IEEE 802.1X value in security configuration is missing or none existing.*
env:Sender ter:InvalidArgVal ter:IncompatibleDot1X	*IEEE 802.1X value in security configuration is incompatible with the network interface.*

8.2.11 Get network protocols

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

Table 22: GetNetworkProtocols command

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

8.2.12 Set network protocols

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

Table 23: SetNetworkProtocols command

SetNetworkProtocols	Request-Response
Message name	**Description**
SetNetworkProtocolsRequest	*This message configures one or more defined network protocols supported by the device. There are currently three protocols defined, HTTP, HTTPS and RTSP. The following parameters can be set for each protocol:* • *Port* • *Enable/disable* tt:NetworkProtocol **NetworkProtocols** [1][unbounded]
SetNetworkProtocols-Response	*This is an empty message*.

Fault codes	Description
env:Sender ter:InvalidArgVal ter:ServiceNotSupported	*The supplied network service is not supported.*

8.2.13 Get default gateway

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

Table 24: GetNetworkDefaultGateway command

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

8.2.14 Set default gateway

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

Table 25: SetNetworkDefaultGateway command

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

8.2.15 Get zero configuration

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

Table 26: GetZeroConfiguration command

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

8.2.16 Set zero configuration

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.

Table 27: SetZeroConfiguration command

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

8.2.17 Get IP address filter

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.

Table 28:GetIPAddressFilter command

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

8.2.18 Set IP address filter

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.

Table 29: SetIPAddressFilter command

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

8.2.19 Add an IP filter address

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.

Table 30: AddIPAddressFilter command

AddIPAddressFilter	Request-Response
Message name	**Description**
AddIPAddressFilterRequest	*This message contains:* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
AddIPAddressFilterResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:IPFilterListIsFull	*It is not possible to add more IP filters since the IP filter list is full.*
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*
env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*

8.2.20 Remove an IP filter address

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.

Table 31: RemoveIPAddressFilter command

RemoveIPAddressFilter	Request-Response
Message name	**Description**
RemoveIPAddressFilter-Request	*This message contains:* • *"IPv4Address": The IPv4 filter address(es)* • *"IPv6Address": The IPv6 filter address(es)* tt:PrefixedIPv4Address **IPv4Address** [0][unbounded] tt:PrefixedIPv6Address **IPv6Address** [0][unbounded]
RemoveIPAddressFilter-Response	*This is an empty message*.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidIPv6Address	*The suggested IPv6 address is invalid.*

env:Sender ter:InvalidArgVal ter:InvalidIPv4Address	*The suggested IPv4 address is invalid.*
env:Sender ter:InvalidArgVal ter:NoIPv6Address	*The IPv6 address to be removed does not exist.*
env:Sender ter:InvalidArgVal ter:NoIPv4Address	*The IPv4 address to be removed does not exist.*

8.2.21 IEEE 802.11 configuration

Requirements in this section and subsections is only valid for a device with IEEE 802.11 support, 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

8.2.21.1 SSID

The device shall support configuration of the SSID.

8.2.21.2 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.

8.2.21.3 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 the priority value is missing from the configuration the lowest priority shall be assumed. 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.

8.2.21.4 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.

8.2.21.4.1 None mode

The device shall support the "None" security mode.

8.2.21.4.2 PSK mode

The device shall support the PSK security mode.

To minimise 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 sufficiently protected.

8.2.21.4.3 IEEE 802.1X-2004 Mode

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

8.2.21.5 Get Dot11 capabilities

This operation returns the IEEE802.11 capabilities, see Table 33. The device shall support this operation.

Table 32: GetDot11Capabilities

GetIEEE802.11Capabilities	Request-Response
Message name	**Description**
GetDot11Capabilities-Request	*This is an empty message*
GetDot11Capabilites-Response	*tt:Dot11Capabilities* ***Capabilities*** *[1][1]*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 configuration is not supported.*

Table 33: 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.

8.2.21.6 Get IEEE 802.11 Status

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)

Table 34: GetDot11Status

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

8.2.21.7 Scan Available IEEE 802.11 Networks

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)

Table 35: ScanAvailable802.11Networks

ScanAvailable802.11Networks	Request-Response
Message name	**Description**
ScanAvailableDot11-NetworksRequest	*tt:ReferenceToken **InterfaceToken** [1][1]*
ScanAvailableDot11-NetworksResponse	*tt:Dot11AvailableNetworks **Networks** [0][unbounded]*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:InvalidDot11	*IEEE 802.11 configuration is not supported.*
env:Sender ter:InvalidArgVal ter:NotDot11	*The interface is not an IEEE 802.11 interface.*
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exist.*
env;Receiver ter:ActionNotSupported ter:NotScanAvailable	*ScanAvailableDot11Networks is not supported.*

8.3 System

8.3.1 Device Information

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

Table 36: GetDeviceInformation command

GetDeviceInformation	Request-Response
Message name	**Description**
GetDeviceInformationRequest	*This is an empty message.*
GetDeviceInformationResponse	*The get device information response message returns following device information:* xs:string **Manufacturer** [1][1] xs:string **Model** [1][1] xs:string **FirmwareVersion** [1][1] xs:string **SerialNumber** [1][1] xs:string **HardwareId** [1][1]
Fault codes	**Description**
	No command specific faults!

8.3.2 Get System URIs

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

System Logs. Multiple system logs may be returned, of different types. The exact format of the system logs is outside the scope of this 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.

Table 37: GetSystemUris command

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

8.3.3 Backup

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

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

Table 38: GetSystemBackup command

GetSystemBackup	Request-Response
Message name	**Description**
GetSystemBackupRequest	*This is an empty message.*
GetSystemBackupResponse	*The get system backup response message contains the system backup configuration files(s).* tt:BackupFile **BackupFiles** [1][unbounded]

Fault codes	Description
	No command specific faults!

8.3.4 Restore

This operation restores the system backup configuration files(s) previously retrieved from a device. The device should support restore of backup configuration file(s) through the RestoreSystem command. The exact format of the backup configuration file(s) is *outside the scope* of this 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].

Table 39: RestoreSystem command

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

8.3.5 Start system restore

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

System restore over HTTP may be achieved using the following steps:

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

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

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

Table 40: StartSystemRestore command

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

8.3.6 Get system date and time

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.

Table 41: GetSystemDateAndTime command

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

8.3.7 Set system date and time

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.

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

Table 42: SetSystemDateAndTime command

SetSystemDateAndTime	Request-Response
Message name	**Description**
SetSystemDateAndTime-Request	*This message contains the date and time information of the device.* • *"DateTimeType": If the system time and date are set manually or by NTP* • *"DaylightSavings": Daylight savings on or off* • *"TimeZone": The time zone is defined in POSIX 1003.1 section 8.3* • *"UTCDateTime": The time and date in UTC. If DateTimeType is NTP, UTCDateTime has no meaning.* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1] tt:DateTime **UTCDateTime** [0][1]
SetSystemDateAndTime-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidTimeZone	*An invalid time zone was specified.*
env:Sender ter:InvalidArgVal ter:InvalidDateTime	*An invalid date or time was specified.*

8.3.8 Factory default

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. The meaning of *soft factory default* is device product-specific and vendor-specific. The effect of a *soft factory default* operation is not fully defined. However, it shall be guaranteed that after a soft reset the device is reachable on the same IP address as used before the reset. This means that basic network settings like IP address, subnet and gateway or DHCP settings are kept unchanged by the soft reset.

Table 43: SetSystemFactoryDefault command

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

8.3.9 Firmware upgrade

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

The firmware is transmitted through MTOM [MTOM].

Table 44: UpgradeSystemFirmware command

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

8.3.10 Start firmware upgrade

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

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

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

Table 45: StartFirmwareUpgrade command

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

8.3.11 Get system logs

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.

Table 46: GetSystemLog command

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

8.3.12 Get support information

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.

Table 47: GetSystemSupportInformation command

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

8.3.13 Reboot

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

Table 48: SystemReboot command

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

8.3.14 Get scope parameters

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 *cannot* be altered through the device management interface but are permanent device characteristics part of the device firmware configurations. The scope type is indicated in the scope list returned in the get scope parameters response. Configurable scope parameters can be set throught the set and add scope parameters operations, see Section 8.3.14 and Section 8.3.15. The device shall support retrieval of discovery scope parameters through the GetScopes command. As some scope parameters are mandatory, the client always expects a scope list in the response.

Table 49: GetScopes command

GetScopes	Request-Response
Message name	**Description**
GetScopesRequest	This is an *empty* message.
GetScopesResponse	*The scope response message contains a list of URIs defining the device scopes. See also Section 7 for the ONVIF scope definitions.* tt:Scope: **Scopes** [1][unbounded]
Fault codes	**Description**
env:Receiver ter:Action ter:EmptyScope	*Scope list is empty.*

8.3.15 Set scope parameters

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.

Table 50: SetScopes command

SetScopes	Request-Response
Message name	**Description**
SetScopesRequest	*The set scope contains a list of URIs defining the device scope. See also Section 7.* xs:anyURI: **Scopes** [1][unbounded]
SetScopesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:ScopeOverwrite	*Scope parameter overwrites fixed device scope setting, command rejected.*
env:Receiver ter:Action ter:TooManyScopes	*The requested scope list exceeds the supported number of scopes.*

8.3.16 Add scope parameters

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.

Table 51: AddScopes command

AddScopes	Request-Response
Message name	**Description**
AddScopesRequest	*The add scope contains a list of URIs to be added to the existing configurable scope list. See also Section 7..* xs:anyURI:**ScopeItem** [1][unbounded]
AddScopesResponse	*This is an empty message.*
Fault codes	**Description**
env:Receiver ter:Action ter:TooManyScopes	*The requested scope list exceeds the supported number of scopes.*

8.3.17 Remove scope parameters

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.

Table 52: RemoveScopes command

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

8.3.18 Get discovery mode

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 GetDiscoveryMode command.

Table 53: GetDiscoveryMode command

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

8.3.19 Set discovery mode

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.

Table 54: SetDiscoveryMode command

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

8.3.20 Get remote discovery mode

This operation gets the remote discovery mode of a device. See Section 7.4 for the definition of remote discovery extensions. A device that supports remote discovery shall support retrieval of the remote discovery mode setting through the GetRemoteDiscoveryMode command.

Table 55: GetRemoteDiscoveryMode command

GetRemoteDiscoveryMode	Request-Response
Message name	**Description**
GetRemoteDiscoveryMode-Request	*This is an empty message.*
GetRemoteDiscoveryMode-Response	*This message contains the current remote discovery mode setting, i.e. discoverable or non-discoverable.* tt:DiscoveryMode: **RemoteDiscoveryMode** [1][1]
Fault codes	**Description**
	No command specific faults!

8.3.21 Set remote discovery mode

This operation sets the remote discovery mode of operation of a device. See Section 7.4 for the definition of remote discovery remote extensions. A device that supports remote discovery shall support configuration of the discovery mode setting through the SetRemoteDiscoveryMode command.

Table 56: SetRemoteDiscoveryMode command

SetRemoteDiscoveryMode	Request-Response
Message name	**Description**
SetRemoteDiscoveryMode-Request	*This message contains the requested remote discovery mode setting, i.e. discoverable or non-discoverable.* tt:DiscoveryMode: **RemoteDiscoveryMode** [1][1]

SetRemoteDiscoveryMode-Response	*This is an empty message.*
Fault codes	**Description**
	No command specific faults!

8.3.22 Get remote DP addresses

This operation gets the remote DP address or addresses from a device. If the device supports remote discovery, as specified in Section 7.4, the device shall support retrieval of the remote DP address(es) through the GetDPAddresses command.

Table 57: GetDPAddresses command

GetDPAddresses	Request-Response
Message name	**Description**
GetDPAddressesRequest	*This is an empty message.*
GetDPAddressesResponse	*This message contains the device configured remote DP address or addresses. If no remote DP address is configured, an empty list is returned.* tt:NetworkHost: **DPAddress** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.3.23 Set remote DP addresses

This operation sets the remote DP address or addresses on a device. If the device supports remote discovery, as specified in Section 7.4, the device shall support configuration of the remote DP address(es) through the SetDPAddresses command.

Table 58: SetDPAddresses command

SetDPAddresses	Request-Response
Message name	**Description**
SetDPAddressesRequest	*This message contains the device configured remote DP address or addresses.* tt:NetworkHost: **DPAddress** [0][unbounded]
SetDPAddressesResponse	*This is an empty message.*
Fault codes	**Description**
	No command specific faults!

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. The WS-Security framework 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 based on WS-Security authentication, then the device shall support this command.

Table 59: GetAccessPolicy command

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

8.4.2 Set access policy

This command sets the 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.

Table 60: SetAccessPolicy command

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

8.4.3 Get users

This operation lists the registered users and corresponding credentials on a device. The device shall support retrieval of registered device users and their credentials for the user token through the GetUsers command.

Table 61: GetUsers command

GetUsers	Request-Response
Message name	**Description**
GetUsersRequest	*This is an empty message.*
GetUsersResponse	*This message contains list of users and corresponding credentials. Each entry includes:* • *Username* • *User level,* *i.e, the username password is not included into the response.* tt:User: **User** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

8.4.4 Create users

This operation creates new device users and corresponding credentials on a device for the user token profile, see Section 5.12 for user token definitions. The device shall support creation of device users and their credentials for the user token through the CreateUsers command. 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, as clients may follow the password derivation mechanism which results in 'password equivalent' of length 28 bytes, as described in 3.1.2 of [ONVIF Security].

Table 62: CreateUsers command

CreateUsers	Request-Response
Message name	**Description**
CreateUsersRequest	*This message contains a user parameters element for a new user. Each user entry includes:* • *Username* • *Password* • *UserLevel* tt:User: **User** [1][unbounded]
CreateUsersResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited	*Username already exists.*

ter:UsernameClash	
env:Sender ter:OperationProhibited ter:PasswordTooLong	*The password is too long*
env:Sender ter:OperationProhibited ter:UsernameTooLong	*The username is too long*
env:Sender ter:OperationProhibited ter:Password	*Too weak password.*
env:Receiver ter:Action ter:TooManyUsers	*Maximum number of supported users exceeded.*
env:Sender ter:OperationProhibited ter:AnonymousNotAllowed	*User level anonymous is not allowed.*
env:Sender ter:OperationProhibited ter:UsernameTooShort	*The username is too short*

8.4.5 Delete users

This operation deletes users on a device for the user token profile, see Section 5.12 for user token definitions. The device shall support deletion of device users and their credentials for the user token 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.

Table 63: DeleteUsers command

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

8.4.6 Set users settings

This operation updates the settings for one or several users on a device for the user token profile. The device shall support update of device users and their credentials for the user token through the SetUser command. Either all change requests are processed successfully or a fault message shall be returned and no change requests be processed.

Table 64: SetUser command

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

8.4.7 IEEE 802.1X configuration

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

- Configuration Token

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

- EAP Identity

  This parameter indicates the user name of supplicant which connects to IEEE 802.1X managed network. This is defined as 'Identity' in [ONVIF Schema].

- EAP method

  This parameter indicates authentication method used. This is defined as 'EAPMethod' in [ONVIF Schema].

- CA Certificate ID

  This parameter indicates the ID of CA certificate used for authentication server verification. This is defined as 'CACertificateID' in [ONVIF Schema].

- Respective configuration parameters for selected EAP method

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

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

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

This IEEE 802.1X parameters will be referred by security configuration as a part of a certain network interface configuration. For the details, please refer to 8.2.10.

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

Note that in ONVIF 2.0 support for IEEE 802.1X is limited to IEEE 802.11 interfaces.

8.4.7.1 Create IEEE 802.1X configuration

This operation newly creates IEEE 802.1X configuration parameter set of the device. The device shall support this command if it supports IEEE 802.1X. If the device receives this request with already existing configuration token (Dot1XConfigurationToken) specification, the device should respond with '*ter:ReferenceToken* ' error to indicate there is some configuration conflict.

Table 65: CreateDot1XConfiguration command

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

env:Sender ter:InvalidArgVal ter:InvalidDot1X	*Invalid IEEE 802.1X configuration.*

8.4.7.2 Set IEEE 802.1X configuration

While the CreateDot1XConfiguration command is trying to create a new configuration parameter set, this operation modifies existing IEEE 802.1X configuration parameter set of the device. A device that support IEEE 802.1X shall support this command.

Table 66: SetDot1XConfigurationRequest command

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

8.4.7.3 Get IEEE 802.1X configuration

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

A device that supports IEEE 802.1X shall support this command.

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

Table 67: GetDot1XConfiguration command

GetDot1XConfiguration	Request-Response

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

8.4.7.4 Get IEEE 802.1X configurations

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

A device that support IEEE 802.1X shall support this command.

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

Table 68: GetDot1XConfigurations command

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

8.4.7.5 Delete IEEE 802.1X configuration

This operation deletes an IEEE 802.1X configuration parameter set from the device. Which configuration should be deleted is specified by the 'Dot1XConfigurationToken' in the request. A device that support IEEE 802.1X shall support this command.

Table 69: DeleteDot1XConfigurations command

DeleteDot1XConfigurations	Request-Response
Message name	**Description**
DeleteDot1XConfigurationRequest	*This message contains:* tt:ReferenceToken **Dot1XConfigurationToken**[1][1]

DeleteDot1XConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:OperationProhibited ter:ReferenceToken	*Invalid Dot1XConfigurationToken error*
env:Receiver ter:OperationProhibited ter:ReferenceToken	*Cannot delete specified IEEE 802.1X configuration.*

8.4.8 Create self-signed certificate

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

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

Table 70: CreateCertificate command

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

8.4.9 Get certificates

This operation gets all device server certificates (including self-signed) for the purpose of TLS authentication and all device client certificates for the purpose of IEEE 802.1X authentication. This command lists only the TLS server certificates and IEEE 802.1X client certificates for the device (neither trusted CA certificates nor trusted root certificates). The certificates are returned as binary data. A device that supports TLS shall support this command and the certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.

Table 71: GetCertificates command

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

8.4.10 Get CA certificates

CA certificates will be loaded into a device and be used for the sake of following two cases. The one is for the purpose of TLS client authentication in TLS server function. The other one is for the purpose of Authentication Server authentication in IEEE 802.1X function. This operation gets all CA certificates loaded into a device. A device that supports either TLS client authentication or IEEE 802.1X shall support this command and the returned certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.

Table 72: GetCACertificates command

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

8.4.11 Get certificate status

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

Table 73: GetCertificatesStatus command

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

8.4.12 Set certificate status

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

Table 74: SetCertificatesStatus command

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

8.4.13 Get certificate request

This operation requests a PKCS #10 certificate signature request from the device. The returned information field shall be either formatted exactly as specified in [PKCS#10] or PEM encoded [PKCS#10] format. In order for this command to work, the device must already have a private/public key pair. This key pair should be referred by *CertificateID* as specified in the input parameter description. This CertificateID refers to the key pair generated using CreateCertificate command defined in Section 8.4.8.

A device that support *onboard* key pair generation that supports either TLS or IEEE 802.1X using client certificate shall support this command.

Table 75: GetPkcs10Request command

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

8.4.14 Get client certificate status

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

Table 76: GetClientCertificateMode command

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

8.4.15 Set client certificate status

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

Table 77: SetClientCertificateMode command

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

8.4.16 Load device certificate

TLS server certificate(s) or IEEE 802.1X client certificate(s) created using the PKCS#10 certificate request command can be loaded into the device using this command (see Section 8.4.13). The certificate ID in the request shall be present. The device may sort the received certificate(s) based on the public key and subject information in the certificate(s).

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

A device that supports *onboard* key pair generation that support either TLS or IEEE 802.1X shall support this command.

The certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.

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

Table 78: LoadCertificates command

LoadCertificates	Request-Response
Message name	**Description**
LoadCertificatesRequest	*This message contains a list of the device certificates to upload.* tt:Certificate **NVTCertificate** [1][unbounded]
LoadCertificatesResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:CertificateFormat	*Bad certificate format or the format is not supported by the device.*

env:Sender ter:InvalidArgVal ter:CertificateID	*Certificate ID already exists.*
env:Sender ter:InvalidArgVal ter:InvalidCertificate	*Invalid Certificate.*

8.4.17 Load device certificates in conjunction with its private key

There might be some cases that a Certificate Authority or some other equivalent creates a certificate without having PKCS#10 certificate signing request. In such cases, the certificate will be bundled in conjunction with its private key. This command will be used for such use case scenarios. The certificate ID in the request is optionally set to the ID value the client wish to have. If the certificate ID is not specified in the request, device can choose the ID accordingly.

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

The certificates shall be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules.

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

Table 79: LoadCertificateWithPrivateKey command

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

8.4.18 Get certificate information request

This operation requests the information of a certificate specified by certificate ID. The device should respond with its “Issuer DN”, “Subject DN”, “Key usage”, "Extended key usage”, “Key Length”, “Version”, “Serial Number”, “Signature Algorithm” and “Validity” data as the

information of the certificate, as long as the device can retrieve such information from the specified certificate. The IssuerDN and SubjectDN shall be encoded using the rules in [RFC 4514].

A device that supports either TLS or IEEE 802.1X should support this command.

Table 80: GetCertificateInformation command

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

8.4.19 Load CA certificates

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

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

Table 81: LoadCACertificates command

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

env:Receiver ter:OperationProhibited ter:MaxCertificates	*Maximum number of Certificates already loaded.*

8.4.20 Delete certificate

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

Table 82: DeleteCertificates command

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

8.4.21 Get remote user

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

The algorithm to use for deriving the password is described in section 5.12.2.1section in previous chapter 5].

Table 83: GetRemoteUser command

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

8.4.22 Set remote user

This operation sets the remote user. A device supporting remote user handling shall support this operation. The user is only valid for the WS-UserToken profile or as a HTTP / RTSP user.

The password that is set shall always be the original (not derived) password.

If UseDerivedPassword is set password derivation shall be done by the device when connecting to a remote device.The algorithm to use for deriving the password is described in section 5.12.2.1 section in previous chapter 5].

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

Table 84: SetRemoteUser command

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

8.4.23 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 shall support the GetEndpointReference command returning the address property of the device service endpoint reference.

Table 85: GetEndpointReference command

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

8.5 Input/Output (I/O)

The commands in section are depricated. Refer to section 9.

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 Get relay outputs

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

Table 86: GetRelayOutputs command

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

8.5.2 Set relay output settings

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.

Table 87: SetRelayOutputSettings command.

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

8.5.3 Trigger relay output

This operation triggers a relay output[1].

Table 88: SetRelayOutputState command

SetRelayOutputState	Request-Response
Message name	**Description**
SetRelayOutputStateRequest	*This message contains:* • *RelayToken": Token reference to the requested relay output.* • *"LogicalState": Trigger request, i.e., active or inactive.* tt:ReferenceToken **RelayOutputToken** [1][1] tt:RelayLogicalState **LogicalState** [1][1]

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

SetRelayOutputStateResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:RelayToken	*Unknown relay token reference.*

8.5.4 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 supported commands can be retrieved by the AuxiliaryData parameter which derives from GetCapabilities command response. The command transmitted by using this command should match one of the supported commands listed in the AuxiliaryData response. If the capability command response lists only *irlampon* command, then the SendAuxiliaryCommand argument will be *irlampon*, which may indicate turning the connected IR lamp on.

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

Table 89: Send auxiliary command

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

8.6 Service specific fault codes

Table 90 lists the device service-specific fault codes. In addition, each command can also generate a generic fault, see Table 6.

The specific faults are defined as sub code of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 90: Device service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:Action	The policy is empty	The device policy file does not exist or it is empty.

	ter:EmptyPolicy		
env:Receiver	ter:Action ter:EmptyScope	The scope list is empty	Scope list is empty.
env:Receiver	ter:Action ter:FirmwareUpgradeFailed	Upgrade failed	The firmware upgrade failed.
env:Receiver	ter:Action ter:KeyGeneration	Generating a key failed	The private/public key generation failed.
env:Receiver	ter:Action ter:Signature	Creating a signature failed	PKCS#10 signature creation failed.
env:Receiver	ter:InvalidArgVal ter:ClientAuth	Client authentication not supported	Trying to enable client authentication, but client authentication is not supported or not configured
env:Receiver	ter:Action ter:TooManyUsers	Too many users	Maximum number of supported users exceeded.
env:Receiver	ter:Action ter:TooManyScopes	Too large scope list	The scope list exceeds the supported number of scopes.
env:Receiver	ter:ActionNotSupported ter:NoSuchService	The service is not supported	The requested WSDL service category is not supported by the device.
env:Sender	ter: InvalidArgs ter:AccesslogUnavailable	No access log available	There is no access log information available.
env:Sender	ter:InvalidArgVal ter:CertificateFormat	Invalid format	Bad certificate format or the format is not supported by the device.
env:Sender	ter:InvalidArgVal ter:CertificateID	Invalid certificate ID	Unknown certificate reference or the certificate ID already exists.
env:Sender	ter:InvalidArgVal ter:CACertificateID	Invalid CA certificate ID	Unknown CA certificate reference or the CA certificate ID already exists.
env:Sender	ter:InvalidArgVal ter:InvalidBackupFile	Invalid file	The backup file(s) are invalid.
env:Sender	ter:InvalidArgVal	Invalid date and time.	An invalid date or time was specified.

	ter:InvalidDateTime		
env:Sender	ter:InvalidArgVal ter:InvalidDnsName	Invalid name	The suggested NTP server name is invalid.
env:Sender	ter: InvalidArgs ter:InvalidFirmware	Invalid firmware	The firmware was invalid i.e. not supported by this device.
env:Sender	ter:InvalidArgVal ter:InvalidGatewayAddress	Invalid address	The supplied gateway address was invalid.
env:Sender	ter:InvalidArgVal ter:InvalidHostname	Invalid name	The requested hostname cannot be accepted by the device.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceSpeed	Invalid speed	The suggested speed is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidInterfaceType	Invalid type	The suggested network interface type is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidIPv4Address	Invalid address	The suggested IPv4 address is invalid.
env:Sender	ter:InvalidArgVal ter:NoIPv4Address	Address does not exist	The IPv4 address to be removed does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidIPv6Address	Invalid address	The suggested IPv6 address is invalid.
env:Sender	ter:InvalidArgVal ter:NoIPv6Address	Address does not exist	The IPv6 address to be removed does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidMtuValue	Invalid data	The MTU value is invalid.
env:Sender	ter:InvalidArgVal ter:InvalidNetworkInterface	Invalid token	The supplied network interface token does not exists
env:Sender	ter:InvalidArgVal ter:InvalidTimeZone	Invalid data	An invalid time zone was specified.

env:Sender	ter:InvalidArgVal ter:IPFilterListIsFull	The list is full	It is not possible to add more IP filters since the IP filter list is full.
env:Sender	ter:InvalidArgVal ter:ModeError	Invalid data	Monostable delay time not valid.
env:Sender	ter:InvalidArgs ter:PolicyFormat	Invalid format	The requested policy cannot be set due to unknown policy format.
env:Sender	ter:InvalidArgVal ter:RelayToken	Unknown relay token.	The token reference is unknown.
env:Sender	ter:InvalidArgVal ter:ServiceNotSupported	The service is not supported	The supplied network service is not supported.
env:Sender	ter:InvalidArgVal ter:SupportInformationUnavailable	No support information available	There is no support information available.
env:Sender	ter: InvalidArgs ter:SystemlogUnavailable	No system log available	There is no system log information available.
env:Sender	ter:InvalidArgVal ter:UsernameMissing	Username not recognized	Username NOT recognized.
env:Sender	ter:OperationProhibited ter:FixedScope	Trying to delete fixed scope parameter	Trying to delete fixed scope parameter, command rejected.
env:Sender	ter:InvalidArgVal ter:NoScope	Scope does not exist	Trying to Remove scope which does not exist.
env:Sender	ter:OperationProhibited ter:Password	Too weak password	Too weak password.
env:Sender	ter:OperationProhibited ter:PasswordTooLong	Too long password	The password is too long.
env:Sender	ter:OperationProhibited ter:UsernameTooShort	Too long password	The password is too short.
env:Sender	ter:OperationProhibited	Trying overwriting permanent device	Scope parameter overwrites permanent device scope

	ter:ScopeOverwrite	scope setting	setting, command rejected.
env:Sender	ter:OperationProhibited ter:UsernameClash	Username already exists	Username already exists.
env:Sender	ter:OperationProhibited ter:UsernameTooLong	Too long username	The username is too long.
env:Receiver	ter:ActionNotSupported ter:InvalidDot11	Not supported	IEEE 802.11Configuration is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidSecurityMode	Not Supported	The selected security mode is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidStationMode	Not Supported	The selected station mode is not supported.
env:Sender	ter:InvalidArgVal ter:MissingDot11	IEEE 802.11 value missing	*IEEE 802.11 value is missing in the security configuration.*
env:Sender	ter:InvalidArgVal ter:MissingPSK	PSK value missing	PSK value is missing in security configuration.
env:Sender	ter:InvalidArgVal ter:MissingDot1X	IEEE 802.1X value is missing	IEEE 802.1X value in security configuration is missing or none existing.
env:Sender	ter:InvalidArgVal ter:IncompatibleDot1X	IEEE 802.!X value is incompatible	IEEE 802.1X value in security configuration is incompatible with the network interface.
env:Sender	ter:InvalidArgVal ter:NotDot11	Not IEEE 802.11	The interface is not an IEEE 802.11 interface.
env:Sender	ter:InvalidArgVal ter:InvalidDot1X	Invalid IEEE 802.1X configuration	Specified IEEE 802.1X configuration is not valid.
env:Receiver	ter:Action ter:NotConnectedDot11	IEEE 802.11 not connected	*IEEE 802.11 network is not connected.*
env:Receiver	ter:ActionNotSupported ter:NotScanAvailable	ScanAvailableIEEE802.11Networks is not supported.	ScanAvailableIEEE802.11Networks is not supported.
env:Receiver	ter:ActionNotSupported	Remote User handling is not supported.	Remote User handling is not supported.

	ter:NotRemoteUser		
env:Receiver	ter:ActionNotSupported ter:EAPMethodNotSupported	The suggested EAP method is not supported.	The suggested EAP method is not supported.
env:Receiver	ter:Action ter:MaxDot1X	Maximum number of IEEE 802.1X configurations reached.	Device reached maximum number of IEEE 802.1X configurations.
env:Receiver	ter:OperationProhibited ter:ReferenceToken	Cannot delete specified IEEE 802.1X configuration.	It is not possible to delete specified IEEE 802.1X configuration.
env:Receiver	ter:OperationProhibited ter:CertificateID	Cannot delete specified Certificate(s).	It is not possible to delete specified Certificate(s).
env:Sender	ter:OperationProhibited ter:ReferenceToken	Invalid Dot1XConfigurationToken error.	Specified IEEE 802.1X configuration token is invalid.
env:Sender	ter:OperationProhibited ter:CertificateID	Invalid Certificate ID error.	Specified Certificate ID is invalid.
env:Sender	ter:InvalidArgVal ter:ReferenceToken	Dot1XConfigurationToken already exists.	Specified Dot1XConfigurationToken already exists in the device.
env:Sender	ter:InvalidArgVal ter:InvalidCertificate	Invalid certificate.	Specified certificate is invalid.
env:Receiver	ter:OperationProhibited ter:MaxCertificates	Maximum number of Certificates already loaded.	Device reached maximum number of loaded Certificates.
env:Sender	ter:OperationProhibited ter:PasswordTooWeak	Too weak password	Too weak password
env:Sender	ter:InvalidArgVal ter:AuxiliaryDataNotSupported	The requested AuxiliaryCommand is not supported.	The requested AuxiliaryCommand is not supported.
env:Sender	ter:InvalidArgVal ter:InvalidTimeOutValue	Invalid Timeout value specified.	Specified TimeOut value is invalid.
env:Sender	ter:OperationProhibited ter:DataLengthOver	Number of available bytes exceeded.	Number of available bytes exceeded.
env:Sender	ter:OperationProhibited ter:DelimiterNotSupport	Sequence of character (delimiter) is not supported.	Sequence of character (delimiter) is not supported.
env:Receiver	ter:OperationProhibited	Device is not ready to operate in command	Device is not ready to operate in command mode.

	ter:InvalidMode	mode.	
env:Sender	ter:InvalidArgVal ter:FixedUser	Removing fixed user	Client trying to remove fixed user.
env:Sender	ter:OperationProhibited ter:AnonymousNotAllowed	User level anonymous is not allowed.	User level anonymous is not allowed.
Env:Sender	ter:InvalidArgVal ter:KeysNotMatching	Keys not matching	The public and private key is not matching.

9 Device IO Service

This service offers commands to retrieve and configure the physical Inputs and Outputs of a device.

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

A device that has physical sources and outputs SHALL support this service as described in [DeviceIOService.wsdl].

Some functionality of this service overlaps with functionality that is defined in the Media Service. If a device (e.g. a NVT) needs to implement both services it should use the commands that are defined in this service to configure its audio in- and outputs or its video sources.

9.1 VideoOutputs

The VideoOutput type represents the physical Video Outputs of a device that can be connected to a monitor to display the video signal. The structure contains the Layout Settings that can be configured using the Display Service (see Section 14).

9.1.1 GetVideoOutputs

This command lists all available video outputs of a device. A device that has one or more physical video outputs shall support listing of available video outputs through the GetVideoOutputs command.

Table 91: GetVideoOutputs command

GetVideoOutputs	Request-Response
Message name	**Description**
GetVideoOutputsRequest	*This is an empty message.*
GetVideoOutputsResponse	*Contains a list of structures describing all available video outputs of the device. If a device has no VideoOutputs an empty list is returned.* tt:VideoOutput **VideoOutputs** [0][unbounded]
Fault codes	**Description**
No specific fault codes.	

9.2 VideoOutputConfiguration

9.2.1 GetVideoOutputConfiguration

This operation requests the configuration of a Video Output. A device that has one or more Video Outputs shall support the retrieval of the VideoOutputConfiguration through this command.

Table 92: GetVideoOutputConfiguration command

GetVideoOutputConfiguration	Request-Response
Message name	**Description**
GetVideoOutputConfigurationRequest	*This message contains the token of the VideoOutput.* tt:ReferenceToken **VideoOutputToken** [1][1]
GetVideoOutputConfigurationResponse	*This message contains the requested VideoOutputConfiguration with the matching token.* tt:VideoOutputConfiguration **VideoOutputConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested VideoOutput indicated with **VideoOutputToken** does not exist.*

9.2.2 SetVideoOutputConfiguration

This operation modifies a video output configuration. A device that has one or more video outputs shall support the setting of its video output configuration through this command.

Table 93: SetVideoOutputConfiguration command

SetVideoOutputConfiguration	Request-Response
Message name	**Description**
SetVideoOutputConfiguration-Request	*The **Configuration** element contains the modified VideoOutput configuration.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoOutputConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoOutputConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*

9.2.3 GetVideoOutputConfigurationOptions

This operation requests the VideoOutputConfigurationOptions of a VideoOutput. A device that has one or more video outputs shall support the retrieval of VideoOutputConfigurationOptions through this command.

Table 94: GetVideoOutputConfigurationOptions command

GetVideoOutputConfigurationOptions	Request-Response
Message name	**Description**
GetVideoOutputConfiguration-OptionsRequest	*The **VideoOutputToken** element specifies the VideoOutput whose options are requested. The VideoOutput shall exist in the device* tt:ReferenceToken **VideoOutputToken**[1][1]
GetVideoOutputConfiguration-OptionsResponse	*The response contains the VideoOutputOptions of the device.* tt:VideoOutputConfigurationOptions **VideoOutputOptions**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*

9.3 VideoSources

A VideoSource represents physical video input. The structure contains the pixel resolution of the video, framerate and imaging settings. The imaging settings can be manipulated through the ImagingService if supported and contains parameters for focus, exposure and brightness, for example.

9.3.1 GetVideoSources

This operation lists all available video sources for the device. The device that has one or more video inputs shall support the listing of available video sources through the GetVideoSources command.

Table 95: GetVideoSources command

GetVideoSources	Request-Response
Message name	**Description**
GetVideoSourcesRequest	*This is an empty message.*
GetVideoSourcesResponse	*Contains a list of structures describing all available video sources of the device. If a device has no Video Source an empty list is returned*

	tt:VideoSource **VideoSource** [0][unbounded]
Fault codes	**Description**
No specific fault codes.	

9.4 VideoSourceConfiguration

A VideoSourceConfiguration contains a reference to a VideoSource and a Bounds structure containing either the whole VideoSource pixel area or a sub-portion of it. The Bounds and VideoSource define the image that is streamed to a client.

9.4.1 GetVideoSourceConfiguration

This operation lists the video source configurations of a VideoSource. A device with one or more video sources shall support the GetVideoSourceConfigurations command.

Table 96: GetVideoSourceConfiguration command

GetVideoSourceConfiguration	Request-Response
Message name	**Description**
GetVideoSourceConfigurationRequest	*This message contains the token of the video input.* tt:ReferenceToken **VideoSourceToken** [1][1]
GetVideoSourceConfigurationResponse	*This message contains the requested VideoSourceConfiguration with the matching token.* tt:VideoSourceConfiguration **VideoSourceConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoSource	*The requested VideoSource indicated with **VideoSourceToken** does not exist.*

9.4.2 SetVideoSourceConfiguration

This operation modifies a video input configuration. A device that has one or more video sources shall support the setting of the VideoSourceConfiguration through this command.

Table 97: SetVideoSourceConfiguration command

SetVideoSourceConfiguration	Request-Response
Message name	**Description**
SetVideoSourceConfiguration-Request	*The **Configuration** element contains the modified VideoSource configuration. The Configuration contains an element that specifies the VideoSource whose configuration is to be modified. The VideoSource shall exist in the device* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoSourceConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoSourceConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoSource	*The requested VideoSource does not exist*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*

9.4.3 GetVideoSourceConfigurationOptions

This operation requests the VideoSourceConfigurationOptions of a VideoSource. A device with one or more video sources shall support this command.

Table 98: GetVideoSourceConfiguartionOptions command

GetVideoSourceConfiguartionOptions	Request-Response
Message name	**Description**
GetVideoSourceConfiguration-OptionsRequest	*The **VideoSourceToken** element specifies the Video Input whose options are requested. The Video Input shall exist in the device* tt:ReferenceToken **VideoSourceToken**[1][1]
GetVideoSourceConfiguartion-OptionsResponse	*The **VideoSourceOptions** return the valid Bounds as well as a element that delivers the VideoSourceToken available. This field shall be set to the Source whose options are requested.* tt:VideoSourceConfigurationOptions **VideoSourceOptions**[1][1]

Fault codes	Description
env:Sender ter:InvalidArgVal ter:NoVideoSource	*The requested Video Input does not exist*

9.5 AudioOutputs

The Audio Output represents the physical audio outputs that can be connected to a loudspeaker.

9.5.1 GetAudioOutputs

This command lists all available audio outputs of a device. A device that has one ore more physical audio outputs shall support listing of available audio outputs through the GetAudioOutputs command.

Table 99: GetAudioOutputs command

GetAudioOutputs	Request-Response
Message name	**Description**
GetAudioOutputsRequest	*This is an empty message.*
GetAudioOutputsResponse	*Contains a list of structures describing all available audio outputs of the device. If a device has no AudioOutputs an empty list is returned.* tt:AudioOutput **AudioOutputs** [0][unbounded]
Fault codes	**Description**
env:Receiver *ter:ActionNotSupported* *ter:AudioOutputNotSupported*	*Audio or Audio Outputs are not supported by the Device*

9.6 AudioOutputConfiguration

An AudioOutputConfiguration contains a reference to an existing AudioOutput. The AudioOutput configuration contains a parameter to control the output level.

9.6.1 GetAudioOutputConfiguration

This operation requests the AudioOutputConfiguration of an AudioOutput. A device that has one or more AudioOutputs shall support the retrieval of the AudioOutputConfiguration through this command.

Table 100: GetAudioOutputConfiguration command

GetAudioOutputConfiguration	Request-Response
Message name	**Description**
GetAudioOutputConfigurationRequest	*This message contains the token of the AudioOutput.* tt:ReferenceToken **AudioOutputToken** [1][1]
GetAudioOutputConfigurationResponse	*This message contains the requested AudioOutputConfiguration with the matching token.* tt:AudioOutputConfiguration **AudioOutputConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioOutput	*The requested AudioOutput indicated with **AudioOutputToken** does not exist.*

9.6.2 SetAudioOutputConfiguration

This operation modifies an audio output configuration. A device that has one ore more audio outputs shall support the setting of the AudioOutputConfiguration through this command.

Table 101: SetAudioOutputConfiguration command

SetAudioOutputConfiguration	Request-Response
Message name	**Description**
SetAudioOutputConfiguration-Request	*The **Configuration** element contains the modified AudioOutput configuration. The Configuration contains an element that specifies specifies the Audio Output whose configuration is to be modified. The Audio Output shall exist in the device.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioOutputConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioOutputConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioOutput	*The requested Audio Output does not exist*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*

9.6.3 GetAudioOutputConfigurationOptions

This operation requests the AudioOutputConfigurationOptions of an AudioOutput. A device that has one or more AudioOutputs shall support this command.

Table 102: GetAudioOutputConfigurationOptions command

GetAudioOutputConfigurationOptions	Request-Response
Message name	**Description**
GetAudioOutputConfiguration-OptionsRequest	*The **AudioOutputToken** element specifies the Audio Output whose options are requested. The Audio Output shall exist in the device* tt:ReferenceToken **AudioOutputToken**[1][1]
GetAudioOutputConfiguration-OptionsResponse	*The **AudioOutputsOptions** return the valid value ranges for SendPrimacy and OutputLevel as well as the AudioOutputToken available. This field shall be set to the Output whose options are requested.* tt:AudioOutputConfigurationOptions **AudioOutputOptions**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioOutput	*The requested Audio Output does not exist*

9.7 AudioSources

An AudioSource represents unencoded audio input and states the number of input channels

9.7.1 GetAudioSources

This operation lists all available audio sources for the device. The device that has one or more audio sources shall support the listing of available audio inputs through the GetAudioSources command.

Table 103: GetAudioSources command

GetAudioSources	Request-Response
Message name	**Description**
GetAudioSourcesRequest	*This is an empty message.*

GetAudioSourcesResponse	*Contains a list of structures describing all available audio sources of the device. If a device has no Audio Input an empty list is returned* tt:AudioSource **AudioSource** [0][unbounded]
Fault codes	**Description**
env:Receiver *ter:ActionNotSupported* *ter:AudioOutputNotSupported*	*NVT does not support audio.*

9.8 AudioSourceConfiguration

An AudioSourceConfiguration contains a reference to an Audio Source.

9.8.1 GetAudioSourceConfiguration

This operation lists the configuration of an Audio Input. A device with one or more audio inputs shall support the GetAudioSourceConfiguration command.

Table 104: GetAudioSourceConfiguration command

GetAudioSourceConfiguration	Request-Response
Message name	**Description**
GetAudioSourceConfigurationRequest	*This message contains the token of the AudioSource.* tt:ReferenceToken **AudioSourceToken** [1][1]
GetAudioSourceConfigurationResponse	*This message contains the requested AudioSourceConfiguration with the matching token.* tt:AudioSourceConfiguration **AudioSourceConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioSource	*The requested AudioSource indicated with* ***AudioSourceToken*** *does not exist.*

9.8.2 SetAudioSourceConfiguration

This operation modifies an audio source configuration. A device that has a one or more audio sources shall support the setting of the AudioSourceConfiguration through this command.

Table 105: SetAudioSourceConfiguration command

SetAudioSourceConfiguration	Request-Response
Message name	**Description**
SetAudioSourceConfiguration-Request	*The **Configuration** element contains the modified AudioSource configuration. The Configuration contains an element that specifies specifies the AudioSource whose configuration is to be modified. The Audio Input shall exist in the device* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioSourceConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioSourceConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioSource	*The requested AudioSource does not exist*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*

9.8.3 GetAudioSourceConfigurationOptions

This operation requests the AudioSourceConfigurationOptions of an AudioSource. A device with one ore more AudioSources shall support this command.

Table 106: GetAudioSourceConfigurationOptions command

GetAudioSourceConfigurationOptions	Request-Response
Message name	**Description**
GetAudioSourceConfigurationOptions-Request	*The **AudioSourceToken** element specifies the Audio Input whose options are requested. The AudioSource shall exist in the device* tt:ReferenceToken **AudioSourceToken**[1][1]
GetAudioSourceConfiguration-Response	*The **AudioSourcesOptions** return the AudioSourceToken available. This field shall be set to the source whose options are requested.* tt:AudioSourceConfigurationOptions **AudioSourceOptions**[1][1]

Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAudioSource	*The requested Audio Input does not exist*

9.9 Relay Outputs

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.

Relay outputs is also defined in DeviceManagement(See Input/Output(I/O). Relay outputs can access both DeviceManagement service and DeviceIO.

9.9.1 Get relay outputs

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

Table 107: GetRelayOutputs command

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

9.9.2 Set relay output settings

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.

Table 108: SetRelayOutputSettings command.

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

9.9.3 Trigger relay output

This operation triggers a relay output[2].

Table 109: SetRelayOutputState command

SetRelayOutputState	Request-Response
Message name	**Description**
SetRelayOutputStateRequest	*This message contains:* • *RelayOutputToken”: Token reference to the requested relay output.* • *“LogicalState”: Trigger request, i.e., active or inactive.* tt:ReferenceToken **RelayOutputToken** [1][1] tt:RelayLogicalState **LogicalState** [1][1]
SetRelayOutputStateResponse	*This is an empty message.*
Fault codes	**Description**

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

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

9.10 Service specific fault codes

Table 110 lists the DeviceIO service specific fault codes. Additionally, each command can also generate a generic fault, see Table 6.

Table 110: DeviceIO service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Sender	ter:InvalidArgVal ter:ConfigModify	Invalid configuration parameters	The configuration parameters are not possible to set.
env:Sender	ter:InvalidArgVal ter:NoVideoOutput	Video output token does not exist.	The requested VideoOutput indicated with **VideoOutputToken** does not exist.
env:Sender	ter:InvalidArgVal ter:NoVideoSource	Video source token does not exist.	The requested VideoSource indicated with **VideoSourceToken** does not exist.
env:Sender	ter:InvalidArgVal ter:NoAudioOutput	Audio output token does not exist.	The requested AudioOutput indicated with **AudioOutputToken** does not exist.
env:Sender	ter:InvalidArgVal ter:NoAudioSource	Audio source token does not exist.	The requested AudioSource indicated with **AudioSourceToken** does not exist.
env:Sender	ter:InvalidArgVal ter:RelayToken	Unknown relay token reference	The requested RelayOutput indicated **RelayOutputToken** does not exist.
env:Sender	ter:InvalidArgVal ter:ModeError	Monostable delay time not valid	

10 Imaging configuration

The imaging service provides operations used to control and configure imaging properties on a device. A device that has one or more video sources should support the imaging service as defined in [ONVIF Imaging WSDL]. The imaging settings are part of the VideoSource entity. This means that imaging parameters directly affect a specific video source.

10.1 Imaging settings

The imaging service provides operations to get or set imaging parameters and the valid ranges for those parameters. Some parameters have no effect if a specific mode is not set. Some of the parameters included in the settings require a specific imaging capability that can be requested through the GetOptions command. The following settings are available through the imaging service operations:

BacklightCompensation: Enables/disables BLC mode (on/off)

- On
    - Optional level parameter (unspecified unit).
- Off

Brightness: Adjusts the image brightness (unspecified unit).

ColorSaturation: Adjusts the color saturation in the image (unspecified unit).

Sharpness: Adjusts the sharpness in the image (unspecified unit).

Contrast: Adjusts the image contrast (unspecified unit).

Exposure:

- Auto – Enables the exposure algorithm on the device:
    - Priority – Sets the exposure priority mode (low noise/framerate).
    - Window – Rectangular exposure mask.
    - Min/MaxExposureTime – Exposure time range allowed to be used by the algorithm.
    - Min/MaxGain – The sensor gain range that is allowed to be used by the algorithm.
    - Min/MaxIris – The iris range allowed to be used by the algorithm.
- Manual – Disables the exposure algorithm on the device:
    - ExposureTime – The fixed exposure time used by the image sensor (μs).
    - Gain – The fixed gain used by the image sensor (dB).

- Iris – The fixed attenuation of input light affected by the iris (dB). 0dB maps to a fully opened iris.

Focus:

- Auto (parameters that apply to automatic mode only):
  - Near/FarLimit – Limits for focus lens (m).
- Manual (parameters that apply to manual mode only):
  - Default speed – The default speed for focus move operation (when the speed parameter not is present). Manual control is done through the move command, see Section 10.1.4.

Ir cut filter: Toggles the Ir cut filter state between on, off and auto. The auto state lets the exposure algorithm handle when the Ir cut filter should be turned on or off.

Whitebalance:

- Auto whitebalancing mode (auto/manual).
- Manual (parameters that apply to manual mode only):
  - Rgain (unitless).
  - Bgain (unitless).

WideDynamicRange: Whide dynamic range (on/off):

- On
  - Optional level parameter (unitless).
- Off

The available imaging settings can be retrieved through the GetVideoSources command part of the media service, as specified in Section 11.3.1. The imaging settings are part of the video source.

10.1.1 Get imaging settings

This operation requests the imaging setting for a video source on the device. If the Video Source supports any of the imaging settings as defined by the ImagingSettings type in the [ONVIF Schema], then it should be possible to retrieve the imaging settings from the device through the GetImagingSettings command.

The imaging settings parameters are described in Section 10.1.

Table 111: GetImagingSettings command

GetImagingSettings	Request-Response
Message name	**Description**
GetImagingSettingsRequest	*This message contains a reference to the VideoSource for which the ImagingSettings should be requested.* tt:ReferenceToken **VideoSourceToken**[1][1]
GetImagingSettingsResponse	*This message contains the ImagingSettings for the VideoSource that was requested* tt:ImagingSettings20**ImagingSettings**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.1.2 Set imaging settings

This operation sets the imaging settings for a video source on a device. If the device supports any of the imaging settings as defined by the ImagingSettings type in [ONVIF Schema], then the it should be possible to configure these parameters in the device through the SetImagingSettings command.

The possible configurable imaging settings parameters are described in Section 10.1. Settings options are obtained through the command defined in Section 10.1.3

Table 112: SetImagingSettings command

SetImagingSettings	Request-Response
Message name	**Description**
SetImagingSettingsRequest	*This message contains a reference to the VideoSource and ImagingSettings that should be set.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:ReferenceToken **VideoSourceToken**[1][1] tt:ImagingSettings20**ImagingSettings**[1][1] xs:boolean **ForcePersistence** [0][1]
SetImagingSettingsResponse	*This message contains no response.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*

env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*
env:Sender ter:InvalidArgVal ter:SettingsInvalid	*The requested settings are incorrect.*

10.1.3 Get options

This operation gets the valid ranges for the imaging parameters that have device specific ranges. If the device supports the SetImagingSettings command to set imaging parameter on the device, then it shall get the configuration options from the device through the GetOptions command.

Table 113:GetOptions command

GetOptions	Request-Response
Message name	**Description**
GetOptionsRequest	*Reference to the VideoSource for which the imaging parameter options are requested.* tt:ReferenceToken **VideoSourceToken**[1][1]
GetOptionsResponse	*This message contains the valid ranges for the imaging parameters that are categorized as device specific.* tt:ImagingOptions20 **ImagingOptions**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.1.4 Move

The Move command moves the focus lens in an absolute, a relative or in a continuous manner from its current position. The speed argument is optional for absolute and relative control, but required for continuous. If no speed argument is used, the default speed is used. Focus adjustments through this operation will turn off the autofocus. A device with support for remote focus control should support absolute, relative or continuous control through the Move operation.

Imaging capabilities specifies which specific focus operations are supported by this operation. At least one focus control capability is required for this operation to be functional.

The move operation contains the following commands:

Absolute – Requires position parameter and optionally takes a speed argument. A unitless type is used by default for focus positioning and speed. Optionally, if supported, the position may be requested in m^{-1} units.

Relative – Requires distance parameter and optionally takes a speed argument. Negative distance means negative direction.

Continuous – Requires a speed argument. Negative speed argument means negative direction.

Table 114: Move (focus) command

Move	Request-Response
Message name	**Description**
MoveRequest	*Reference to the VideoSource for the requested move (focus) operation.* tt:ReferenceToken **VideoSourceToken**[1][1] tt:FocusMove **Focus**[1][1]
MoveResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.1.5 Get move options

The GetMoveOptions command retrieves the focus lens move options to be used in the move command as defined in Section 10.1.4. A device that supports the lens move operation shall also support the GetMoveOptions command.

Table 115: GetMoveOptions (focus) command

GetMoveOptions	Request-Response
Message name	**Description**
GetMoveOptionsRequest	*Reference to the VideoSource for the requested move options.* tt:ReferenceToken **VideoSourceToken**[1][1]
GetMoveOptionsResponse	*This message contains the valid ranges for the focus lens move options.* tt:MoveOptions20 **MoveOptions**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.1.6 Stop

The Stop command stops all ongoing focus movements of the lense. If the device supports focus, it should be possible to stop focus through the stop operation. The operation will not affect ongoing autofocus operation.

Table 116: Stop (focus) command

Stop	Request-Response
Message name	**Description**
StopRequest	*Reference to the VideoSource where the focus movement should be stopped.* tt:ReferenceToken **VideoSourceToken**[1][1]
StopResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.1.7 Get imaging status

The GetStatus command requested the current imaging status from the device. If the device supports focus move control, then it should be possible to get the available imaging status through the GetStatus command.

The imaging status contains:

- Focus position, move status and error information.
    - The focus position is represented in a unitless type.
    - Move status may be in a MOVING, IDLE or UNKNOWN state.
    - Error information provided as a string, for example a positioning error indicated by the hardware.

Table 117: GetStatus (focus) command

GetStatus	Request-Response
Message name	**Description**
GetStatusRequest	*This message contains a reference to the VideoSource where the imaging status should be requested.* tt:VideoSourceToken **VideoSourceToken**[1][1]
GetStatusResponse	This message contains the requested imaging status. tt:ImagingStatus20 **ImagingStatus**[1][1]

Fault codes	Description
env:Sender ter:InvalidArgVal ter:NoSource	*The requested VideoSource does not exist.*
env:Receiver ter:ActionNotSupported ter:NoImagingForSource	*The requested VideoSource does not support imaging settings.*

10.2 Service specific fault codes

Table 118 lists the imaging service specific fault codes. In addition each command can also generate a generic fault, see Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 118: Imaging specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:ActionNotSupported ter:NoImagingForSource	VideoSource does not support imaging settings	The requested VideoSource does not support imaging settings.
env:Sender	ter:InvalidArgVal ter:SettingsInvalid	Invalid configuration	The requested settings are incorrect.
env:Sender	ter:InvalidArgVal ter:NoSource	Video source does not exist	The requested VideoSource does not exist.

11 Media configuration

The media service is used to configure the NVT media streaming properties. The NVT shall support the media service as specified in [ONVIF Media WSDL].

The media service allows a client to configure media and other real time streaming configurations. Media configurations are handled through media profiles. An overview of the ONVIF media configuration model is given in Section 4.8.

The media service commands are divided into two major categories:

- Media configuration:
    - Media profile commands
    - Video source commands
    - Vide encoder commands
    - Audio source commands
    - Audio encoder commands
    - Video analytics commands
    - Metadata commands
    - Audio output commands
    - Audio decoder commands
- Media streaming:
    - Request stream URI
    - Get snapshot URI
    - Multicast control commands
    - Media synchronization point

A basic set of operations are required for the media service; other operations are recommended to support. The detailed requirements are listed under the command descriptions.

11.1 Audio and video codecs

The NVT streams audio and video data using suitable encoding algorithms. The NVT may also able to decode audio. The NVT supports any audio and video codecs, bitrates and resolution according to the manufacturer's choice. In order to ensure interoperability between the client and NVT, this standard mandates the following codec profiles:

- The NVT shall support JPEG QVGA.

- The NVT shall support G.711μ Law (Simplex-Camera Microphone Only, 1ch) [ITU-T G.711] if the NVT supports audio.

11.2 Media Profile

A media profile consists of a set of media configurations. Media profiles are used by a client to configure properties of a media stream from an NVT.

An NVT shall provide at least one media profile at boot. An NVT should provide “ready to use” profiles for the most common media configurations that the device offers.

A profile consists of a set of interconnected *configuration entities.* Configurations are provided by the NVT and can be either static or created dynamically by the NVT. For example, the dynamic configurations can be created by the NVT depending on current available encoding resources. A configuration entity is one of the following:

- Video source configuration

- Audio source configuration

- Video encoder configuration

- Audio encoder configuration

- PTZ configuration

- Video analytics configuration

- Metadata configuration

- Audio output configuration

- Audio decoder configuration

A profile consists of all or a subset of these configuration entities. Depending on the capabilities of the NVT, a particular configuration entity can be part of a profile or not. For example, a profile with an audio source and an audio encoder configuration can exist only in a device with audio support.

11.2.1 Create media profile

This operation creates a new empty media profile. The media profile shall be created in the NVT and shall be persistent (remain after reboot). The NVT shall support the creation of media profiles as defined in this standard through the CreateProfile command.

A created profile shall be deletable and an NVT shall set the “fixed” attribute to false in the returned Profile.

Table 119: CreateProfile command

CreateProfile	Request-Response
Message name	**Description**
CreateProfileRequest	*Contains the friendly **Name** of the Profile to create as well as an optional **Token** parameter, specifying the unique identifier of the new media profile* tt:Name **Name** [1][1] tt:ReferenceToken **Token** [0][1]
CreateProfileResponse	*Returns an empty Profile structure with no configuration entities.* tt:Profile **Profile** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:ProfileExists	*A profile with the token **ProfileToken** already exists.*
env:Receiver ter:Action ter:MaxNVTProfiles	*The maximum number of supported profiles has been reached.*

11.2.2 Get media profiles

Any endpoint can ask for the *existing* media profiles of an NVT using the GetProfiles command. Pre-configured or dynamically configured profiles can be retrieved using this command. This command lists *all* configured profiles in a device. The client does not need to know the media profile in order to use the command. The NVT shall support the retrieval of media profiles through the GetProfiles command.

A NVT shall include the “fixed” attribute in all the returned Profile elements.

Table 120: GetProfiles command

GetProfiles	Request-Response
Message name	**Description**
GetProfilesRequest	*This is an empty message.*
GetProfilesResponse	*The response contains a list of profiles. Each profile contains a set of configuration entities defining a specific configuration that can be used for media streaming, analytics, metadata streaming etc.* tt:Profile **Profiles** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

11.2.3 Get media profile

If the profile token is already known, a profile can be fetched through the GetProfile command. The NVT shall support the retrieval of a specific media profile through the GetProfile command.

A NVT shall include the “fixed” attribute in the returned Profile element.

Table 121: GetProfile command

GetProfile	Request-Response
Message name	**Description**
GetProfileRequest	*This message contains the token to the requested profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetProfileResponse	*The response contains the **Profile** indicated by the **Token** parameter. A Profile contains a set of configuration entities defining a specific configuration that can be used for media streaming, analytics, metadata streaming etc.* tt:Profile **Profile** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*

11.2.4 Add video source configuration to a profile

This operation adds a VideoSourceConfiguration to an existing media profile. If such a configuration exists in the media profile, it will be replaced. The change shall be persistent. The NVT shall support addition of a video source configuration to a profile through the AddVideoSourceConfiguration command.

Table 122: AddVideoSourceConfiguration command

AddVideoSourceConfiguration	Request-Response
Message name	**Description**
AddVideoSourceConfiguration Request	*Contains a reference to the **VideoSourceConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddVideoSourceConfiguration Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoSourceConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*

11.2.5 Add video encoder configuration to a profile

This operation adds a VideoEncoderConfiguration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT shall support addition of a video encoder configuration to a profile through the AddVideoEncoderConfiguration command.

Adding a VideoEncoderConfiguration to a Profile means that a stream using that Profile will contain video data. Video encoder configurations should be added after adding a video source configuration.

Table 123: AddVideoEncoderConfiguration command

AddVideoEncoderConfiguration	Request-Response
Message name	**Description**
AddVideoEncoderConfiguration Request	*Contains a reference to the **VideoEncoderConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddVideoEncoderConfiguration Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoEncoderConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*

11.2.6 Add audio source configuration to a profile

This operation adds an AudioSourceConfiguration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT that supports audio streaming from NVT to client shall support addition of audio source configuration to a profile through the AddAudioSourceConfiguration command.

Table 124: AddAudioSourceConfiguration command

AddAudioSourceConfiguration	Request-Response
Message name	**Description**
AddAudioSourceConfiguration Request	*Contains a reference to the **AudioSourceConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddAudioSourceConfiguration Response	*This is an empty message.*
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The AudioSourceConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*Audio is not supported.*

11.2.7 Add audio encoder configuration to a profile

This operation adds an AudioEncoderConfiguration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT that supports audio streaming from NVT to client shall support addition of audio encoder configurations to a profile through the AddAudioEncoderConfiguration command.

Adding an AudioEncoderConfiguration to a media profile means that streams using that media profile will contain audio data. Audio encoder configurations should be added after adding an audio source configuration.

Table 125: AddAudioEncoderConfiguration command

AddAudioEncoderConfiguration	Request-Response
Message name	**Description**
AddAudioEncoderConfiguration Request	*Contains a reference to the **AudioEncoderConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddAudioEncoderConfiguration Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The AudioEncoderConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*Audio is not supported.*

11.2.8 Add PTZ configuration to a profile

This operation adds a PTZConfiguration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT that supports PTZ control shall support addition of PTZ configurations to a profile through the AddPTZConfiguration command.

Adding a PTZConfiguration to a media profile means that streams using that media profile can contain PTZ status (in the metadata), and that the media profile can be used for controlling PTZ movement, see Section 16.

Table 126: AddPTZConfiguration command

AddPTZConfiguration	Request-Response
Message name	**Description**
AddPTZConfigurationRequest	*Contains a reference to the **PTZConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddPTZConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The PTZConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

11.2.9 Add video analytics configuration to a profile

This operation adds a VideoAnalytics configuration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT that supports video analytics shall support addition of video analytics configurations to a profile through the AddVideoAnalyticsConfiguration command.

Adding a VideoAnalyticsConfiguration to a media profile means that streams using that media profile can contain video analytics data (in the metadata) as defined by the submitted configuration reference. Video analytics data is specified in Section 17.1 and analytics configurations are managed through the commands defined in Section 11.9.

A profile containing only a video analytics configuration but no video source configuration is incomplete. Therefore, a client should first add a video source configuration to a profile before adding a video analytics configuration. The NVT can deny adding of a video analytics configuration before a video source configuration. In this case, it should respond with a ConfigurationConflict Fault.

Table 127: AddVideoAnalytics command

AddVideoAnalytics	Request-Response
Message name	**Description**
AddVideoAnalyticsRequest	*Contains a reference to the **VideoAnalytics** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1]

	tt:ReferenceToken **ConfigurationToken** [1][1]
AddVideoAnalyticsResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalytics indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:VideoAnalyticsNotSupported	*VideoAnalytics is not supported.*

11.2.10 Add metadata configuration to a profile

This operation adds a Metadata configuration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT shall support the addition of a metadata configuration to a profile though the AddMetadataConfiguration command.

Adding a MetadataConfiguration to a Profile means that streams using that profile contain metadata. Metadata can consist of events, PTZ status, and/or video analytics data. Metadata configurations are handled through the commands defined in Section 11.10 and 11.9.4.

Table 128: AddMetadataConfiguration command

AddMetadataConfiguration	Request-Response
Message name	**Description**
AddMetadataConfiguration Request	*Contains a reference to the **MetadataConfiguration** to add and the **Profile** where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddMetadataConfiguration Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The MetadataConfiguration indicated by the **ConfigurationToken** does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*

11.2.11 Add audio output configuration

This operation adds an AudioOutputConfiguration to an existing media profile. If a configuration exists in the media profile, it will be replaced. The change shall be persistent. An NVT that has an Audio Output shall support addition of an audio output configuration to a profile through the AddAudioOutputConfiguration command.

Table 129: AddAudioOutputConfiguration

AddAudioOutputConfiguration	Request-Response
Message name	**Description**
AddAudioOutputConfiguration Request	*Contains a reference to the AudioOutputConfiguration to add and the Profile where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddAudioOutputConfiguration Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgs ter:NoConfig	*The AudioOutputConfiguration indicated by the ConfigurationToken does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Output is not supported*

11.2.12 Add audio decoder configuration

This operation adds an AudioDecoderConfiguration to an existing media profile. If a configuration exists in the media profile, it shall be replaced. The change shall be persistent. An NVT that has audio decoding capabilities shall support addition of an audio decoder configuration to a profile through the AddAudioDecoderConfiguration command.

Table 130: AddAudioDecoderConfiguration

AddAudioDecoderConfiguration	Request-Response
Message name	**Description**
AddAudioDecoderConfiguration Request	*Contains a reference to the AudioConfiguration to add and the Profile where it shall be added.* tt:ReferenceToken **ProfileToken** [1][1] tt:ReferenceToken **ConfigurationToken** [1][1]
AddAudioDecoderConfiguration Response	*This is an empty message.*
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgs ter:NoConfig	*The AudioDecoderConfiguration indicated by the ConfigurationToken does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile conflicts with the one to add and adding it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio Decoding is not supported*

11.2.13 Remove video source configuration from a profile

This operation removes a VideoSourceConfiguration from an existing media profile. If the media profile does not contain a VideoSourceConfiguration, the operation has no effect. The removal shall be persistent. The NVT shall support removal of a video source configuration from a profile through the RemoveVideoSourceConfiguration command.

Video source configurations should only be removed after removing a VideoEncoderConfiguration from the media profile.

Table 131: RemoveVideoSourceConfiguration command

RemoveVideoSourceConfiguration	Request-Response
Message name	**Description**
RemoveVideoSourceConfiguration-Request	*Contains a reference to the media profile from which the VideoSourceConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveVideoSourceConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no video source configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the VideoSourceConfiguration and removing it would cause a conflicting media profile.*

11.2.14 Remove video encoder configuration from a profile

This operation removes a VideoEncoderConfiguration from an existing media profile. If the media profile does not contain a VideoEncoderConfiguration, the operation has no effect. The removal shall be persistent. The NVT shall support removal of a video encoder configuration from a profile through the RemoveVideoEncoderConfiguration command.

Table 132: RemoveVideoEncoderConfiguration command

RemoveVideoEncoderConfiguration	Request-Response
Message name	**Description**
RemoveVideoEncoderConfiguration-Request	*Contains a reference to the media profile from which the VideoEncoderConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveVideoEncoderConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no video encoder configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the VideoEncoderConfiguration and removing it would cause a conflicting media profile.*

11.2.15 Remove audio source configuration from a profile

This operation removes an AudioSourceConfiguration from an existing media profile. If the media profile does not contain an AudioSourceConfiguration, the operation has no effect. The removal shall be persistent. An NVT that supports audio streaming from NVT to client shall support removal of an audio source configuration from a profile through the RemoveAudioSourceConfiguration command.

Audio source configurations should only be removed after removing an AudioEncoderConfiguration from the media profile.

Table 133: RemoveAudioSourceConfiguration command

RemoveAudioSourceConfiguration	Request-Response
Message name	**Description**
RemoveAudioSourceConfiguration-Request	*Contains a reference to the media profile from which the AudioSourceConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveAudioSourceConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no audio source configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the AudioSourceConfiguration and removing it would cause a conflicting media profile.*

env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*Audio is not supported.*

11.2.16 Remove audio encoder configuration from a profile

This operation removes an AudioEncoderConfiguration from an existing media profile. If the media profile does not contain an AudioEncoderConfiguration, the operation has no effect. The removal shall be persistent. An NVT that supports audio streaming from NVT to client shall support removal of audio encoder configurations from a profile through the RemoveAudioEncoderConfiguration command.

Table 134: RemoveAudioEncoderConfiguration command

RemoveAudioEncoderConfiguration	Request-Response
Message name	**Description**
RemoveAudioEncoderConfiguration-Request	*Contains a reference to the media profile from which the AudioEncoderConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveAudioEncoderConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no audio encoder configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the AudioEncoderConfiguration and removing it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*Audio is not supported.*

11.2.17 Remove PTZ configuration from a profile

This operation removes a PTZConfiguration from an existing media profile. If the media profile does not contain a PTZConfiguration, the operation has no effect. The removal shall be persistent. An NVT that supports PTZ control shall support removal of PTZ configurations from a profile through the RemovePTZConfiguration command.

Table 135: RemovePTZConfiguration command

RemovePTZConfiguration	Request-Response
Message name	**Description**
RemovePTZConfiguration-Request	*Contains a reference to the media profile from which the PTZConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemovePTZConfiguration-Response	*This is an empty message.*

Fault codes	Description
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no PTZ configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the PTZConfiguration and removing it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

11.2.18 Remove video analytics configuration from a profile

This operation removes a VideoAnalyticsConfiguration from an existing media profile. If the media profile does not contain a VideoAnalyticsConfiguration, the operation has no effect. The removal shall be persistent. An NVT that supports video analytics shall support removal of a video analytics configuration from a profile through the RemoveVideoAnalyticsConfiguration command.

Table 136: RemoveVideoAnalyticsConfiguration command

RemoveVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
RemoveVideoAnalyticsConfiguration-Request	*Contains a reference to the media profile from which the VideoAnalyticsConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveVideoAnalyticsConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no video analytics configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the VideoAnalyticsConfiguration and removing it would cause a conflicting media profile.*
env:Receiver ter:ActionNotSupported ter:VideoAnalyticsNotSupported	*VideoAnalytics is not supported.*

11.2.19 Remove metadata configuration from a profile

This operation removes a MetadataConfiguration from an existing media profile. If the media profile does not contain a MetadataConfiguration, the operation has no effect. The removal shall be persistent. An NVT shall support the removal of a metadata configuration from a profile through the RemoveMetadataConfiguration command.

Table 137: RemoveMetadataConfiguration command

RemoveMetadataConfiguration	Request-Response
Message name	**Description**
RemoveMetadataConfiguration-Request	*Contains a reference to the media profile from which the MetadataConfiguration shall be removed.* tt:ReferenceToken **ProfileToken** [1][1]
RemoveMetadataConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no metadata configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the MetadataConfiguration and removing it would cause a conflicting media profile.*

11.2.20 Remove audio output configuration

This operation removes an AudioOutputConfiguration from an existing media profile. If the media profile does not contain an AudioOutputConfiguration, the operation has no effect. The removal shall be persistent. An NVT that has at least one audio output shall support removal of an audio output configuration from a profile through the RemoveAudioOutputConfiguration command.

Table 138: RemoveAudioOutputConfiguration

RemoveAudioOutputConfiguration	Request-Response
Message name	**Description**
RemoveAudioOutputConfiguration-Request	*Contains a reference to the media profile from which the AudioOutputConfiguration shall be removed*. tt:ReferenceToken **ProfileToken** [1][1]
RemoveAudioOutputConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no audio output configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the AudioOutputConfiguration and removing it would cause a conflicting media profile.*
env: Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	Audio or Audio output is not supported

11.2.21 Remove audio decoder configuration

This operation removes an AudioDecoderConfiguration from an existing media profile. If the media profile does not contain an AudioDecoderConfiguration, the operation has no effect. The removal shall be persistent. An NVT that supports audio decoding shall support removal of an audio decoder configuration from a profile through the RemoveAudioDecoderConfiguration command.

Table 139: RemoveAudioDecoderConfiguration

RemoveAudioDecoderConfiguration	Request-Response
Message name	**Description**
RemoveAudioDecoderConfiguration-Request	*Contains a reference to the media profile from which the AudioDecoderConfiguration shall be removed*. tt:ReferenceToken **ProfileToken** [1][1]
RemoveAudioDecoderConfiguration-Response	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*There exists no audio decoder configuration in the media profile.*
env:Receiver ter:Action ter:ConfigurationConflict	*Other configurations of the media profile are dependant on the AudioDecoder Configuration and removing it would cause a conflicting media profile.*
env: Receiver ter:ActionNotSupported ter::AudioDecodingNotSupported	*Audio or AudioDecoding is not supported*

11.2.22 Delete media profile

This operation deletes a profile. This change shall always be persistent. The NVT shall support the deletion of a media profile through the DeleteProfile command.

Table 140: DeleteProfile command

DeleteProfile	Request-Response
Message name	**Description**
DeleteProfileRequest	*Contains a **ProfileToken** that indicates what media profile to delete.* tt:ReferenceToken **ProfileToken** [1][1]
DeleteProfileResponse	*This is an empty message.*
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:Action ter:DeletionOfFixedProfile	*The fixed Profile cannot be deleted.*

11.3 Video source

A VideoSource represents unencoded video input. The structure contains the pixel resolution of the video, framerate and imaging settings. The imaging settings can be manipulated through the ImagingService if supported and contains parameters for focus, exposure and brightness, for example. See Section 10 for more information.

11.3.1 GetVideoSources

This operation lists all available video sources for the device. The NVT shall support the listing of available video sources through the GetVideoSources command.

Table 141: GetVideoSources command

GetVideoSources	Request-Response
Message name	**Description**
GetVideoSourcesRequest	*This is an empty message.*
GetVideoSourcesResponse	*Contains a list of structures describing all available video sources of the device.* tt:VideoSource **VideoSources** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

11.4 Video source configuration

A VideoSourceConfiguration contains a reference to a VideoSource and a Bounds structure containing either the whole VideoSource pixel area or a sub-portion of it. The Bounds and VideoSource define the image that is streamed to a client. If a VideoSourceConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.4.1 Get video source configurations

This operation lists all *existing* video source configurations for an NVT. This command lists *all* video source configurations in a device. The client need not know anything about the video source configurations in order to use the command. The NVT shall support the listing of available video source configurations through the GetVideoSourceConfigurations command.

Table 142: GetVideoSourceConfigurations command

GetVideoSourceConfigurations	Request-Response

Message name	Description
GetVideoSourceConfigurations-Request	*This is an empty message.*
GetVideoSourceConfigurations-Response	*This message contains a list of* all *existing video source configurations in the NVT. A video source configuration does always point at a real video source with the SourceToken element.* tt:VideoSourceConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

11.4.2 Get video source configuration

If the video source configuration token is already known, the video source configuration can be fetched through the GetVideoSourceConfiguration command. The NVT shall support retrieval of specific video source configurations through the GetVideoSourceConfiguration command.

Table 143: GetVideoSourceConfiguration command

GetVideoSourceConfiguration	Request-Response
Message name	**Description**
GetVideoSourceConfiguration-Request	*This message contains the token of the requested video source configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetVideoSourceConfiguration-Response	*This message contains the requested VideoSourceConfiguration with the matching token. A video source configuration does always point at a real video source with the SourceToken element.* tt:VideoSourceConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

11.4.3 Get compatible video source configurations

This operation requests all the video source configurations of the NVT that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddVideoSourceConfiguration command on the media profile. The result will vary depending on the capabilities, configurations and settings in the device. The NVT shall support the listing of compatible (with a specific profile) video source configurations through the GetCompatibleVideoSourceConfigurations command.

Table 144: GetCompatibleVideoSourceConfigurations command

GetCompatibleVideoSourceConfigurations	Request-Response
Message name	**Description**

GetCompatibleVideoSource-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleVideoSource-ConfigurationsResponse	*Contains a list of video source configurations that are compatible with the media profile.* tt:VideoSourceConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*

11.4.4 Get video source configuration options

This operation returns the available options when the video source parameters are reconfigured If a video source configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. The NVT shall support the listing of available video source parameter options (for a given profile and configuration) through the GetVideoSourceConfigurationOptions command.

Table 145: GetVideoSourceConfigurationOptions command

GetVideoSourceConfigurationOptions	Request-Response
Message name	**Description**
GetVideoSourceConfiguration-OptionsRequest	*This message contains optional tokens of a video source configuration and a media profile.* ***ConfigurationToken** specifies an existing configuration that the options are intended for.* ***ProfileToken** specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]
GetVideoSourceConfiguration-OptionsResponse	*This message contains the video configuration options. If a video source configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:VideoSourceConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*

11.4.5 Modify a video source configuration

This operation modifies a video source configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the NVT. Running streams using this configuration may be immediately updated according to the new settings. The changes are not guaranteed to take effect unless the client requests a new stream URI and restarts any affected stream. NVC methods for changing a running stream are out of scope for this specification. The NVT shall support the modification of video source parameters through the SetVideoSourceConfiguration command.

Table 146: SetVideoSourceConfiguration command

SetVideoSourceConfiguration	Request-Response
Message name	**Description**
SetVideoSourceConfiguration-Request	*The **Configuration** element contains the modified video source configuration. The configuration shall exist in the NVT.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoSourceConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoSourceConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*

11.5 Video encoder configuration

A VideoEncoderConfiguration contains the following parameters for configuring the encoding of video data:

- Encoder – The encoding used for the video data.
- Resolution – The pixel resolution of the encoded video data.
- Quality – Determines the quality of the video. A high value within supported quality range means higher quality.
- RateControl – Defines parameters to configure the bitrate [kbps] as well as an EncodingInterval parameter (Interval at which images are encoded and transmitted) and a FrameRateLimit [fps] parameter to configure the output framerate.

- MPEG4/H264 specifics – Defines the encoding profile and GOV length [frame].

TheVideoEncoderConfiguration structure also contains multicast parameters and a session timeout to define video streaming behaviour. If a VideoEncoderConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.5.1 Get video encoder configurations

This operation lists all *existing* video encoder configurations of an NVT. This command lists *all* configured video encoder configurations in a device. The client need not know anything apriori about the video encoder configurations in order to use the command. The NVT shall support the listing of available video encoder configurations through the GetVideoEncoderConfigurations command.

Table 147: GetVideoEncoderConfigurations command

GetVideoEncoderConfigurations	Request-Response
Message name	**Description**
GetVideoEncoderConfigurations-Request	*This is an empty message.*
GetVideoEncoderConfigurations-Response	*This message contains a list of* all *existing video encoder configurations in the NVT.* tt:VideoEncoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

11.5.2 Get video encoder configuration

If the video encoder configuration token is already known, the encoder configuration can be fetched through the GetVideoEncoderConfiguration command. The NVT shall support the retrieval of a specific video encoder configuration through the GetVideoEncoderConfiguration command.

Table 148: GetVideoEncoderConfiguration command

GetVideoEncoderConfiguration	Request-Response
Message name	**Description**
GetVideoEncoderConfiguration-Request	*This message contains the token of the requested video encoder configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetVideoEncoderConfiguration-Response	*This message contains the requested VideoEncoderConfiguration with the matching token.* tt:VideoEncoderConfiguration **Configuration** [1][1]
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

11.5.3 Get compatible video encoder configurations

This operation lists all the video encoder configurations of the NVT that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddVideoEncoderConfiguration command on the media profile. The result will vary depending on the capabilities, configurations and settings in the device. The NVT shall support the listing of compatible (with a specific profile) video encoder configurations through the GetCompatibleVideoEncoderConfigurations command.

Table 149: GetCompatibleVideoEncoderConfigurations command

GetCompatibleVideoEncoderConfigurations	Request-Response
Message name	**Description**
GetCompatibleVideoEncoder-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleVideoEncoder-ConfigurationsResponse	*Contains a list of video encoder configurations that are compatible with the given media profile.* tt:VideoEncoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*

11.5.4 Get video encoder configuration options

This operation returns the available options when the video encoder parameters are reconfigured. The NVT shall support the listing of available video parameter options (for a given profile and configuration) through the GetVideoEncoderConfigurationOptions command.

Table 150: GetVideoEncoderConfigurationOptions command

GetVideoEncoderConfigurationOptions	Request-Response
Message name	**Description**
GetVideoEncoderConfiguration-OptionsRequest	*This message contains optional tokens of a video encoder configuration and a media profile.* ***ConfigurationToken** specifies an existing configuration that the options are intended for.* ***ProfileToken** specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]

GetVideoEncoderConfiguration-OptionsResponse	*This message contains the video configuration options. If a video encoder configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:VideoEncoderConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*

11.5.5 Modify a video encoder configuration

This operation modifies a video encoder configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the NVT. Changes in the Multicast settings shall always be persistent. Running streams using this configuration may be immediately updated according to the new settings, but the changes are not guaranteed to take effect unless the client requests a new stream URI and restarts any affected stream. If the new settings invalidate any parameters already negotiated using RTSP, for example by changing codec type, the NVT must not apply these settings to existing streams. Instead it must either continue to stream using the old settings or stop sending data on the affected streams.

NVC methods for changing a running stream are out of scope for this specification. The NVT shall support the modification of video encoder parameters through the SetVideoEncoderConfiguration command.

Table 151: SetVideoEncoderConfiguration command

SetVideoEncoderConfiguration	Request-Response
Message name	**Description**
SetVideoEncoderConfiguration-Request	*The **Configuration** element contains the modified video encoder configuration. The configuration shall exist in the NVT.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoEncoderConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoEncoderConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*

env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*

11.5.6 Get guaranteed number of video encoder instances

The GetGuaranteedNumberOfVideoEncoderInstances command can be used to request the minimum number of guaranteed video encoder instances (applications) per Video Source Configuration. An NVT SHALL support this command. This command was added in ONVIF 1.02.

Table 152: GetGuaranteedNumberOfVideoEncoderInstances command

GetGuaranteedNumberOfVideoEncoderInstances	Request-Response
Message name	**Description**
GetGuaranteedNumberOf-EncoderInstancesRequest	*This request contains a token to the video source configuration.* tt: ReferenceToken **ConfigurationToken** [1][1]
GetGuaranteedNumberOf-EncoderInstancesResponse	*This message contains the minimum guaranteed **TotalNumber** of encoder instances (applications) per VideoSourceConfiguration. If a device limits the number of instances for respective Video Codecs the response contains the information how many **Jpeg**, **H264** and **Mpeg4** can be set up at the same time. In all other cases the device is able to deliver the **TotalNumber** of streams independend from the configured VideoCodec at the same time.* xs:int **TotalNumber** [1][1] xs:int **JPEG** [0][1] xs:int **H264** [0][1] xs:int **MPEG4** [0][1]
Fault codes	**Description**
env:Sender *ter:InvalidArgVal* *ter:NoConfig*	*The requested configuration indicated with **ConfigurationToken** does not exist.*

11.6 Audio source

An AudioSource represents unencoded audio input and states the number of input channels.

11.6.1 Get audio sources

This operation lists all available audio sources of the device. An NVT that supports audio streaming from NVT to client shall support listing of available audio sources through the GetAudioSources command.

Table 153: GetAudioSources command

GetAudioSources	Request-Response
Message name	**Description**
GetAudioSourcesRequest	*This message is empty.*

GetAudioSourcesResponse	*Contains a list of structures describing all available audio sources of the device.* tt:AudioSource **AudioSources** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.7 Audio source configuration

An AudioSourceConfiguration contains a reference to an AudioSource that is to be used for input in a media profile. If an AudioSourceConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.7.1 Get audio source configurations

This operation lists all *existing* audio source configurations of an NVT. This command lists *all* audio source configurations in a device. The client need not know anything apriori about the audio source configurations in order to use the command. An NVT that supports audio streaming from NVT to client shall support listing of available audio source configurations through the GetAudioSourceConfigurations command.

Table 154: GetAudioSourceConfigurations command

GetAudioSourceConfigurations	Request-Response
Message name	**Description**
GetAudioSourceConfigurations-Request	*This is an empty message.*
GetAudioSourceConfigurations-Response	*This message contains a list of* all *existing audio source configurations in the NVT. An audio source configuration does always point at a real audio source with the SourceToken element.* tt:AudioSourceConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.7.2 Get audio source configuration

The GetAudioSourceConfiguration command fetches the audio source configurations if the audio source configuration token is already known. An NVT that supports audio streaming from NVT to client shall support the retrieval of a specific audio source configuration through the GetAudioSourceConfiguration command.

Table 155: GetAudioSourceConfiguration command

GetAudioSourceConfiguration	Request-Response
Message name	**Description**
GetAudioSourceConfiguration-Request	*This message contains the token of the requested audio source configuration. An audio source configuration does always point at a real audio source with the SourceToken element.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAudioSourceConfiguration-Response	*This message contains the requested AudioSourceConfiguration with the matching token.* tt:AudioSourceConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.7.3 Get compatible audio source configurations

This operation requests all audio source configurations of a device that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddAudioSourceConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. An NVT that supports audio streaming from NVT to client shall support listing of compatible (with a specific profile) audio source configurations through the GetCompatibleAudioSourceConfigurations command.

Table 156: GetCompatibleAudioSourceConfigurations command

GetCompatibleAudioSourceConfigurations	Request-Response
Message name	**Description**
GetCompatibleAudioSource-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleAudioSource-ConfigurationsResponse	*Contains a list of audio source configurations that are compatible with the media profile.* tt:AudioSourceConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.7.4 Get audio source configuration options

This operation returns the available options when the audio source parameters are reconfigured. If an audio source configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. An NVT that supports audio streaming from NVT to client shall support the listing of available audio parameter options (for a given profile and configuration) through the GetAudioSourceConfigurationOptions command.

Table 157: GetAudioSourceConfigurationOptions command

GetAudioSourceConfigurationOptions	Request-Response
Message name	**Description**
GetAudioSourceConfiguration-OptionsRequest	*This message contains optional tokens of an audio source configuration and a media profile.* ***ConfigurationToken*** *specifies an existing configuration that the options are intended for.* ***ProfileToken*** *specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]
GetAudioSourceConfiguration-OptionsResponse	*This message contains the audio configuration options. If an audio source configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:AudioSourceConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token* ***ProfileToken*** *does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.7.5 Modify an audio source configuration

This operation modifies an audio source configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the NVT. Running streams using this configuration may be immediately updated according to the new settings, but the changes are not guaranteed to take effect unless the client requests a new stream URI and restarts any affected stream. If the new settings invalidate any parameters already negotiated using RTSP, for example by changing codec type, the NVT must not apply these settings to existing streams. Instead it must either continue to stream using the old settings or stop sending data on the affected streams.

NVC methods for changing a running stream are out of scope for this specification. An NVT that supports audio streaming from NVT to client shall support the configuration of audio source parameters through the SetAudioSourceConfiguration command.

Table 158: SetAudioSourceConfiguration command

SetAudioSourceConfiguration	Request-Response
Message name	**Description**
SetAudioSourceConfiguration-Request	*The **Configuration** element contains the modified audio source configuration. The configuration shall exist in the NVT.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioSourceConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioSourceConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.8 Audio encoder configuration

An AudioEncoderConfiguration contains the following parameters for encoding audio data:

- Encoder – The encoding used for audio data.
- Bitrate – The output bitrate [kbps].
- SampleRate – The output sample rate [kHz].

The AudioEncoderConfiguration structure also contains multicast parameters and a session timeout to define audio streaming behaviour.

If an AudioEncoderConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.8.1 Get audio encoder configurations

This operation lists all *existing* device audio encoder configurations. The client need not know anything apriori about the audio encoder configurations in order to use the command. An NVT that supports audio streaming from NVT to client shall support the listing of available audio encoder configurations through the GetAudioEncoderConfigurations command.

Table 159: GetAudioEncoderConfigurations command

GetAudioEncoderConfigurations	Request-Response
Message name	**Description**
GetAudioEncoderConfigurations-Request	*This is an empty message.*
GetAudioEncoderConfigurations-Response	*This message contains a list of* all *existing audio encoder configurations in the NVT.* tt:AudioEncoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.8.2 Get audio encoder configuration

The GetAudioEncoderConfiguration command fetches the encoder configuration if the audio encoder configuration token is known. An NVT that supports audio streaming from NVT to client shall support the listing of a specific audio encoder configuration through the GetAudioEncoderConfiguration command.

Table 160: GetAudioEncoderConfiguration command

GetAudioEncoderConfiguration	Request-Response
Message name	**Description**
GetAudioEncoderConfiguration-Request	*This message contains the token of the requested audio encoder configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAudioEncoderConfiguration-Response	*This message contains the requested AudioEncoderConfiguration with the matching token.* tt:AudioEncoderConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.8.3 Get compatible audio encoder configurations

This operation requests all audio encoder configurations of the NVT that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddAudioEncoderConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. An NVT that supports audio streaming from NVT to client shall support listing of compatible (with a specific profile) audio encoder configurations through the GetCompatibleAudioEncoderConfigurations command.

Table 161: GetCompatibleAudioEncoderConfigurations command

GetCompatibleAudioEncoderConfigurations	Request-Response
Message name	**Description**
GetCompatibleAudioEncoder-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleAudioEncoder-ConfigurationsResponse	*Contains a list of audio encoder configurations that are compatible with the given media profile.* tt:AudioEncoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.8.4 Get audio encoder configuration options

This operation returns the available options when the audio encoder parameters are reconfigured. An NVT that supports audio streaming from NVT to client shall support the listing of available audio encoder parameter options (for a given profile and configuration) through the GetAudioEncoderConfigurationOptions command.

Table 162: GetAudioEncoderConfigurationOptions command

GetAudioEncoderConfigurationOptions	Request-Response
Message name	**Description**
GetAudioEncoderConfiguration-OptionsRequest	*This message contains optional tokens of an audio encoder configuration and a media profile.* ***ConfigurationToken** specifies an existing configuration that the options are intended for.* ***ProfileToken** specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]

GetAudioEncoderConfiguration-OptionsResponse	*This message contains the audio configuration options. If a audio encoder configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:AudioEncoderConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.8.5 Modify audio encoder configurations

This operation modifies an audio encoder configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the NVT. Changes in the Multicast settings shall always be persistent. Running streams using this configuration may be immediately updated according to the new settings. The changes are not guaranteed to take effect unless the client requests a new stream URI and restarts any affected streams. NVC methods for changing a running stream are out of scope for this specification. An NVT that supports audio streaming from NVT to client shall support the configuration of audio encoder parameters through the SetAudioEncoderConfiguration command.

Table 163: SetAudioEncoderConfiguration command

SetAudioEncoderConfiguration	Request-Response
Message name	**Description**
SetAudioEncoderConfiguration-Request	*The **Configuration** element contains the modified audio encoder configuration. The configuration shall exist in the NVT.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioEncoderConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioEncoderConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*

env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

11.9 Video analytics configuration

VideoAnalyticsConfiguration contains parameters for an *analytics engine* and a *rule engine* (see Section 4.12). Thereby, the analytics engine consists of multiple modules which can be managed by the analytics module part of the analytics service. Similarly, the rule engine consists of multiple rules which can be managed by the rule engine part of the analytics service. The subsequent commands are introduced to handle complete video analytics configuration in an atomar way. For instance, the ModifyVideoAnalyticsConfiguration command changes analytics and rule engine configuration in an atomar operation. When a video analytics configuration is present in a profile, the metadata configuration can activate the streaming of the scene description within the RTP streams (see Section 11.10).

A device MAY NOT allow referencing the very same VideoAnalyticsConfiguration from multiple media profiles with different VideoSourceConfigurations. If the device allows it, it shall generate individual scene descriptions for each profile, since the coordinate system of a scene description relates to a specific VideoSourceConfiguration. Also masking and geometrical rules relate to the coordinate system of the VideoSourceConfiguration. This MAY require separate processing of the whole video analytics for each VideoSourceConfiguration, even if they refer to the very same VideoSource.

Since the options of a VideoAnalyticsConfiguration are dynamic and often vendor specific, they can only be retrieved via the video analytics service.

11.9.1 Get video analytics configurations

This operation lists all video analytics configurations of a device. This command lists *all* configured video analytics in a device. The client need not know anything apriori about the video analytics in order to use the command. A device that supports video analytics shall support the listing of available video analytics configuration through the GetVideoAnalyticsConfigurations command.

Table 164: GetVideoAnalyticsConfigurations command

GetVideoAnalyticsConfigurations	Request-Response
Message name	**Description**
GetVideoAnalyticsConfigurations-Request	*This message is empty.*
GetVideoAnalyticsConfigurations-Response	*This message contains a list of* all *existing video analytics configurations in the device.* tt:VideoAnalyticsConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:ActionNotSupported ter:VideoAnalyticsNot-Supported	*Device does not support video analytics.*

11.9.2 Get video analytics configuration

The GetVideoAnalyticsConfiguration command fetches the video analytics configuration if the video analytics token is known. A device that supports video analytics shall support the listing

of a specific video analytics configuration through the GetVideoAnalyticsConfiguration command.

Table 165: GetVideoAnalyticsConfiguration command

GetVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
GetVideoAnalyticsConfiguration-Request	*This message contains the token of an existing video analytics configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetVideoAnalyticsConfiguration-Response	*This message contains the requested video analytics configuration.* tt:VideoAnalyticsConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*
env:Sender ter:ActionNotSupported ter:VideoAnalyticsNot-Supported	*The device does not support video analytics.*

11.9.3 Get compatible video analytics configurations

This operation requests all video analytic configurations of the device that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddVideoAnalyticsConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. A device that supports video analytics shall support the listing of compatible (with a specific profile) video analytics configuration through the GetCompatibleVideoAnalyticsConfigurations command.

Table 166: GetCompatibleVideoAnalyticsConfigurations command

GetCompatibleVideoAnalyticsConfigurations	Request-Response
Message name	**Description**
GetCompatibleVideoAnalytics-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleVideoAnalytics-ConfigurationsResponse	*Contains a list of video analytics configurations that are compatible with the given media profile.* tt:VideoAnalyticsConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:ActionNotSupported	*The device does not support video analytics.*

ter:VideoAnalyticsNot-Supported	

11.9.4 Modify a video analytics configuration

A video analytics configuration is modified using this command. The ForcePersistence flag indicates if the changes shall remain after reboot of the device or not. Running streams using this configuration shall be immediately updated according to the new settings. Otherwise inconsistencies can occur between the scene description processed by the rule engine and the notifications produced by analytics engine and rule engine which reference the very same video analytics configuration token. A device that supports video analytics shall support the configuration of video analytics parameters through the SetVideoAnalyticsConfiguration command.

Table 167: SetVideoAnalyticsConfiguration command

SetVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
SetVideoAnalyticsConfiguration-Request	*The **Configuration** element contains the modified video analytics configuration. The configuration shall exist in the device.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoAnalyticsConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoAnalyticsConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*
env:Sender ter:ActionNotSupported ter:VideoAnalyticsNot-Supported	*The device does not support video analytics.*

11.10 Metadata configuration

A MetadataConfiguration contains parameters for selecting the data to include in the metadata stream. The choices include PTZ status, PTZ position, events as defined by a subscription and analytics data . The event subscription data is described in Section 15.5. The analytics parameters define which data to include from the analytics engine part of the profile, see Section 11.9.

The structure also contains multicast parameters used to configure and control multicast of the metadata stream. A session timeout parameter defines the session timeout (see Section 12.2.1.1.1)

If a MetadataConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.10.1 Get metadata configurations

This operation lists all *existing* metadata configurations. The client need not know anything apriori about the metadata in order to use the command. A NVT or another device that supports metadata streaming shall support the listing of existing metadata configurations through the GetMetadataConfigurations command.

Table 168: GetMetadataConfigurations command

GetMetadataConfigurations	Request-Response
Message name	**Description**
GetMetadataConfigurations-Request	*This message is empty.*
GetMetadataConfigurations-Response	*This message contains a list of* all *existing metadata configurations in the device.* tt:MetadataConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

11.10.2 Get metadata configuration

The GetMetadataConfiguration command fetches the metadata configuration if the metadata token is known. A NVT or another device that supports metadata streaming shall support the listing of a specific metadata configuration through the GetMetadataConfiguration command.

Table 169: GetMetadataConfiguration command

GetMetadataConfiguration	Request-Response
Message name	**Description**
GetMetadataConfiguration-Request	*This message contains the token of an existing metadata configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetMetadataConfiguration-Response	*This message contains the requested metadata configuration.* tt:MetadataConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with* ***ConfigurationToken*** *does not exist.*

11.10.3 Get compatible metadata configurations

This operation requests all the metadata configurations of the device that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddMetadataConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. A NVT or other device that supports metadata streaming shall support the listing of compatible (with a specific profile) metadata configuration through the GetCompatibleMetadataConfigurations command.

Table 170: GetCompatibleMetadataConfigurations command

GetCompatibleMetadataConfigurations	Request-Response
Message name	**Description**
GetCompatibleMetadata-ConfigurationsRequest	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleMetadata-ConfigurationsResponse	*Contains a list of metadata configurations that are compatible with the given media profile.* tt:MetadataConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*

11.10.4 Get metadata configuration options

This operation returns the available options for changing the metadata configuration. A NVT or another device that supports metadata streaming shall support the listing of available metadata parameter options (for a given profile and configuration) through the GetMetadataConfigurationOptions command.

Table 171: GetMetadataConfigurationOptions command

GetMetadataConfigurationOptions	Request-Response
Message name	**Description**
GetMetadataConfiguration-OptionsRequest	*This message contains optional tokens of a metadata configuration and a media profile.* ***ConfigurationToken** specifies an existing configuration that the options are intended for.* ***ProfileToken** specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]
GetMetadataConfiguration-OptionsResponse	*This message contains the metadata configuration options. If a metadata configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.*

	tt:MetadataConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*

11.10.5 Modify a metadata configuration

This operation modifies a metadata configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the device. Changes in the Multicast settings shall always be persistent. Running streams using this configuration may be updated immediately according to the new settings. The changes are not guaranteed to take effect unless the client requests a new stream URI and restarts any affected streams. NVC methods for changing a running stream are out of scope for this specification. A NVT or another device that supports metadata streaming shall support the configuration of metadata parameters through the SetMetadataConfiguration command.

Table 172: SetMetadataConfiguration command

SetMetadataConfiguration	Request-Response
Message name	**Description**
SetMetadataConfiguration-Request	*The **Configuration** element contains multicast settings as well as a set of filters determining what data to include in the metadata stream.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:MetadataConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetMetadataConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*

11.11 Audio outputs

The Audio Output represents the physical audio outputs that can be connected to a loudspeaker.

11.11.1 Get audio outputs

This command lists all available audio outputs of a device. An NVT that has one or more physical audio outputs shall support listing of available audio outputs through the GetAudioOutputs command.

Table 173: GetAudioOutputs

GetAudioOutputs	Request-Response
Message name	**Description**
GetAudioOutputsRequest	*This is an empty message.*
GetAudioOutputsResponse	*Contains a list of structures describing all available audio outputs of the device. If a device has no AudioOutputs an empty list is returned.* tt:AudioOutput **AudioOutputs** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the NVT*

11.12 Audio output configuration

The audio output configuration contains the following parameters:

- SourceToken: a reference to an existing audio output.
- OutputLevel: a parameter to configure the output volume
- SendPrimacy: a parameter that can be used for NVTs with a half duplex audio in/output to configure the active transmission direction (see Section 11.14).

If an AudioOutputConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.12.1 Get audio output configurations

This command lists all existing AudioOutputConfigurations of a device. The NVC need not know anything apriori about the audio configurations to use this command. An NVT that is able to output audio shall support the listing of AudioOutputConfigurations through this command.

Table 174: GetAudioOutputConfiguration

GetAudioOutputConfigurations	Request-Response
Message name	**Description**
GetAudioOutputConfigurationsRequest	*This is an empty message.*

GetAudioOutputConfigurationsResponse	*Contains a list of AudioOutputConfigurations that are available on the device* tt:AudioOutputConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env: Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the device*

11.12.2 Get audio output configuration

If the audio output configuration token is already known, the output configuration can be fetched through the GetAudioOutputConfiguration command. An NVT that has one or more audio outputs shall support the retrieval of a specific audio output configuration through the GetAudioOutputConfiguration command.

Table 175: GetAudioOutputConfiguration

GetAudioOutputConfiguration	Request-Response
Message name	**Description**
GetAudioOutputConfigurationRequest	*This message contains the token of the requested AudioOutput configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAudioOutputConfigurationResponse	*This message contains the requested AudioOutputConfiguration with the matching token.* tt:AudioOutputConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*
env: Receiver ter:ActionNotSupported ter::AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the device*

11.12.3 Get compatible audio output configurations

This command lists all audio output configurations of a device that are compatible with a certain media profile. Each returned configuration shall be a valid input for the AddAudioOutputConfiguration command. An NVT that has one or more audio outputs shall support the listing of compatible (with a specific profile) AudioOutputConfigurations through the GetCompatibleAudioOutputConfigurations command.

Table 176: GetCompatibleAudioOutputConfiguration

GetCompatibleAudioOutputConfigurations	Request-Response
Message name	**Description**

GetCompatibleAudioOutputConfigurations Request	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleAudioOutputConfigurations Response	*Contains a list of audio output configurations that are compatible with the given media profile.* tt:AudioOutputConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the device*

11.12.4 Get audio output configuration options

This operation returns the available options for configuring an audio output. An NVT that has one or more audio outputs shall support the listing of available audio output configuration options (for a given profile and configuration) through the GetAudioOutputConfigurationOptions command.

Table 177: GetAudioOutputConfigurationOptions

GetAudioOutputConfigurationOptions	Request-Response
Message name	**Description**
GetAudioOutputConfiguration-OptionsRequest	*This message contains optional tokens of an audio output configuration and a media profile.* *ConfigurationToken specifies an existing configuration that the options are intended for.* *ProfileToken specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]
GetAudioOutputConfiguration-OptionsResponse	*This message contains the audio output configuration options. If a audio output configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:AudioOutputConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*

env:Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the device*

11.12.5 Modify audio output configuration

This operation modifies an audio output configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the device. An NVT that has one or more audio outputs shall support the modification of audio output parameters through the SetAudioOutputConfiguration command.

Table 178: SetAudioOutputConfiguration

SetAudioOutputConfiguration	Request-Response
Message name	**Description**
SetAudioOutputConfiguration-Request	*The Configuration element contains the modified Audio Output configuration. The configuration must exist in the device.* *The ForcePersistence element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioOutputConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioOutputConfiguration-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*
env: Receiver ter:ActionNotSupported ter:AudioOutputNotSupported	*Audio or Audio Outputs are not supported by the device*

11.13 Audio decoder configuration

The Audio Decoder Configuration does not contain any that parameter to configure the decoding .A decoder shall decode every data it receives (according to its capabilities).

If an AudioDecoderConfiguration is used inside a profile its UseCount parameter is increased to indicate that changing this configuration could affect other users.

11.13.1 Get audio decoder configurations

This command lists all existing AudioDecoderConfigurations of a device.

The NVC need not know anything apriori about the audio decoder configurations in order to use this command. An NVT that is able to decode audio shall support the listing of AudioOutputConfigurations through this command.

Table 179: GetAudioDecoderConfigurations

GetAudioDecoderConfigurations	Request-Response
Message name	**Description**
GetAudioDecoderConfigurationsRequest	*This is an empty message.*
GetAudioDecoderConfigurationsResponse	*Contains a list of AudioDecoderConfigurations that are available on the device* tt:AudioDecoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio decoding is not supported by the device*

11.13.2 Get audio decoder configuration

If the audio decoder configuration token is already known, the decoder configuration can be fetched through the GetAudioDecoderConfiguration command. An NVT that is able to decode audio shall support the retrieval of a specific audio decoder configuration through the GetAudioDecoderConfiguration command.

Table 180: GetAudioDecoderConfiguration

GetAudioDecoderConfiguration	Request-Response
Message name	**Description**
GetAudioDecoderConfigurationRequest	*This message contains the token of the requested AudioDecoder configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAudioDecoderConfigurationResponse	*This message contains the requested AudioDecoder Configuration with the matching token.* tt:AudioDecoderConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio decoding is not supported by the device*

11.13.3 Get compatible audio decoder configurations

This operation lists all the audio decoder configurations of the device that are compatible with a certain media profile. Each of the returned configurations shall be a valid input parameter for the AddAudioDecoderConfiguration command on the media profile. An NVT that is able to decode audio shall support the listing of compatible (with a specific profile) audio decoder configurations through the GetCompatibleAudioDecoderConfigurations command.

Table 181: GetCompatibleAudioDecoderConfigurations

GetCompatibleAudioDecoderConfigurations	Request-Response
Message name	**Description**
GetCompatibleAudioDecoderConfigurations Request	*Contains the token of an existing media profile.* tt:ReferenceToken **ProfileToken** [1][1]
GetCompatibleAudioDecoderConfigurations Response	*Contains a list of audiodecoder configurations that are compatible with the given media profile.* tt:AudioDecoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio decoding is not supported by the device*

11.13.4 Get audio decoder configuration options

This command list the audio decoding capabilities for a given profile and configuration of a device. A device that is able to decode audio shall support the retrieval of AudioDecoderConfigurationOptions through this command.

Table 182: GetAudioDecoderConfigurationOptions

GetAudioDecoderConfigurationOptions	Request-Response
Message name	**Description**
GetAudioDecoderConfiguration-OptionsRequest	This message contains optional tokens of a audio decoder configuration and a media profile. ConfigurationToken specifies an existing configuration that the options are intended for. *ProfileToken specifies an existing media profile that the options shall be compatible with.* tt:ReferenceToken **ConfigurationToken** [0][1] tt:ReferenceToken **ProfileToken** [0][1]

GetAudioDecoderConfiguration-OptionsResponse	*This message contains the audio decoder configuration options. If a audio decoder configuration is specified, the options shall concern that particular configuration. If a media profile is specified, the options shall be compatible with that media profile. If no tokens are specified, the options shall be considered generic for the device.* tt:AudioDecoderConfigurationOptions **Options** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token ProfileToken does not exist.*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio decoding is not supported by the device*

11.13.5 Modify audio decoder configuration

This operation modifies an audio decoder configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the device. The device that is able to decode audio shall support the modification of audio decoder parameters through the SetAudioDecoderConfiguration command.

Table 183: SetAudioDecoderConfiguration

SetAudioDecoderConfiguration	Request-Response
Message name	Description
SetAudioDecoderConfiguration-Request	*The Configuration element contains the modified AudioDecoder configuration. The configuration must exist in the device.* *The ForcePersistence element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AudioDecoderConfiguration **Configuration** [1][1] xs:boolean **ForcePersistence** [1][1]
SetAudioDecoderConfiguration-Response	*This message is empty.*
Fault codes	Description
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*

env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*
env: Receiver ter:ActionNotSupported ter:AudioDecodingNotSupported	*Audio or Audio decoding is not supported by the device*

11.14 Audio channel modes

An audio channel MAY support different types of audio transmission. While for full duplex operation no special handling is required, in half duplex operation the transmission direction needs to be switched.

An optional Send-Primacy Parameter inside the AudioOutputConfiguration indicates which direction is currently active. An NVC can switch between different modes by setting the AudioOutputConfiguration.

The following modes for the Send-Primacy are defined:

- www.onvif.org/ver20/HalfDuplex/Server
  The server is allowed to send audio data to the client. The client shall not send audio data via the backchannel to the NVT in this mode.

- www.onvif.org/ver20/HalfDuplex/Client
  The client is allowed to send audio data via the backchannel to the server. The NVT shall not send audio data to the client in this mode.

- www.onvif.org/ver20/HalfDuplex/Auto
  It is up to the device how to deal with sending and receiving audio data.

Acoustic echo cancellation is out of ONVIF scope.

11.15 Stream URI

11.15.1 Request stream URI

This operation requests a URI that can be used to initiate a live media stream using RTSP as the control protocol. The returned URI shall remain valid indefinitely even if the profile is changed. The ValidUntilConnect, ValidUntilReboot and Timeout Parameter shall be set accordingly (ValidUntilConnect=false, ValidUntilReboot=false, timeout=PT0S). An NVT shall support the retrieval of a media stream URI for a specific media profile through the GetStreamUri command.

For full compatibility with other ONVIF services a device should not generate Uris longer than 128 octets.

Table 184: GetStreamUri command

GetStreamUri	Request-Response
Message name	**Description**
GetStreamUriRequest	*The **StreamSetup** element contains two parts. StreamType defines if a unicast or multicast media stream is requested. Transport specifies a chain of transport protocols defining the tunnelling of the media stream over different network protocols.* *The **ProfileToken** element indicates the media profile to use and will define the configuration of the content of the stream.* tt:StreamSetup **StreamSetup** [1][1] tt:ReferenceToken **ProfileToken** [1][1]
GetStreamUriResponse	*Contains the stable **Uri** to be used for requesting the media stream as well as parameters defining the lifetime of the Uri. The **ValidUntilConnect** and **ValidUntilReboot** parameter shall be set to false, the **timeout** parameter shall be set to PT0S to indicate that this stream URI is indefinitely valid even if the profile changes.* xs:anyURI **Uri** [1][1] xs:boolean **InvalidAfterConnect** [1][1] xs:boolean **InvalidAfterReboot** [1][1] xs:duration **Timeout** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The media profile does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidStreamSetup	*Specification of StreamType or Transport part in **StreamSetup** is not supported.*
env:Sender ter:OperationProhibited ter:StreamConflict	*Specification of StreamType or Transport part in **StreamSetup** causes conflict with other streams.*
env:Receiver ter:Action ter:IncompleteConfiguration	*The specified media profile does contain either unused sources or encoder configurations without a corresponding source.*

11.16 Snapshot

11.16.1 Request snapshot URI

A Network client uses the GetSnapshotUri command to obtain a JPEG snhapshot from the NVT. The returned URI shall remain valid indefinitely even if the profile is changed. The ValidUntilConnect, ValidUntilReboot and Timeout Parameter shall be set accordingly (ValidUntilConnect=false, ValidUntilReboot=false, timeout=PT0S). The URI can be used for acquiring a JPEG image through a HTTP GET operation. The image encoding will always be JPEG regardless of the encoding setting in the media profile. A NVT shall support this command.

Table 185: GetSnapshotUri command

GetSnapshotUri	Request-Response
Message name	**Description**
GetSnapshotUriRequest	*The **ProfileToken** element indicates the media profile to use and will define the source and dimensions of the snapshot.* tt:ReferenceToken **ProfileToken** [1][1]
GetSnapshotUriResponse	*Contains a stable **Uri** to be used for acquiring a snapshot in JPEG format as well as parameters defining the lifetime of the Uri. The **ValidUntilConnect** and **ValidUntilReboot** parameter shall be set to false, the **timeout** parameter shall be set to PT0S to indicate that this stream URI is indefinitely valid even if the profile changes.* xs:anyURI **Uri** [1][1] xs:boolean **InvalidAfterConnect** [1][1] xs:boolean **InvalidAfterReboot** [1][1] xs:duration **Timeout** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The media profile does not exist.*
env:Receiver ter:Action ter:IncompleteConfiguration	*The specified media profile does not contain either a reference to a video encoder configuration or a reference to a video source configuration.*

11.17 Multicast

See Section 12.1 for a detailed discussion of NVT and client multicast streaming.

11.17.1 Start multicast streaming

This command starts multicast streaming using a specified media profile of an NVT. Streaming continues until StopMulticastStreaming is called for the same Profile. The streaming shall continue after a reboot of the NVT until a StopMulticastStreaming request is received. The multicast address, port and TTL are configured in the VideoEncoderConfiguration, AudioEncoderConfiguration and MetadataConfiguration respectively. An NVT that supports video, audio or metadata multicast streaming shall support the starting of a multicast stream through the StartMulticastStreaming command.

Table 186: StartMulticastStreaming command

StartMulticastStreaming	Request-Response
Message name	**Description**
StartMulticastStreaming-Request	*Contains the token of the Profile that is used to define the multicast stream.* tt:ReferenceToken **ProfileToken** [1][1]
StartMulticastStreaming-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The profile does not exist.*
env:Receiver ter:Action ter:IncompleteConfiguration	*The specified media profile does not contain either a reference to a video encoder a video source configuration, to a audio source or to audio encoder configuration or a reference to a metadata configuration*

11.17.2 Stop multicast streaming

This command stop multicast streaming using a specified media profile of an NVT An NVT that supports video, audio or metadata multicast streaming shall support the stopping of a multicast stream through the StopMulticastStreaming command.

Table 187: StopMulticastStreaming command

StopMulticastStreaming	Request-Response
Message name	**Description**
StopMulticastStreaming-Request	*Contains the token of the Profile that is used to define the multicast stream.* tt:ReferenceToken **ProfileToken** [1][1]
StopMulticastStreaming-Response	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The profile does not exist.*
env:Receiver ter:Action ter:IncompleteConfiguration	*The specified media profile does not contain either a reference to a video encoder a video source configuration, to a audio source or to audio encoder configuration or a reference to a metadata configuration*

11.18 Synchronization Points

11.18.1 Set synchronization point

Synchronization points allow clients to decode and correctly use all data after the synchronization point.

For example, if a video stream is configured with a large I-frame distance and a client loses a single packet, the client does not display video until the next I-frame is transmitted. In such cases, the client can request a Synchronization Point which enforces the NVT to add an I-

Frame as soon as possible. Clients can request Synchronization Points for profiles. The NVT shall add synchronization points for all streams associated with this profile.

Similarly, a synchronization point is used to get an update on full PTZ or event status through the metadata stream.

If a video stream is associated with the profile, an I-frame shall be added to this video stream. If an event stream is associated to the profile, the synchronization point request shall be handled as described in Section 15.6). If a PTZ metadata stream is associated to the profile, the PTZ position shall be repeated within the metadata stream.

An NVT that supports MPEG-4 or H.264 shall support the request for an I-Frame through the SetSynchronizationPoint command.

Table 188: SetSynchronizationPoint command

SetSynchronizationPoint	Request-response
Message name	**Description**
SetSynchronizationPointRequest	Contains a Profile reference for which a Synchronization Point is requested. tt:ReferenceToken **ProfileToken** [1][1]
SetSynchronizationPointResponse	This message is empty.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	The profile does not exist.

11.19 Service specific fault codes

The table below lists the media service specific fault codes. Additionally, each command can also generate a generic fault, see Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 189: Media service specific fault codes

Fault Code	**Parent Subcode** **Subcode**	**Fault Reason**	**Description**
env:Receiver	ter:ActionNotSupported ter:AudioNotSupported	No audio capability	NVT does not support audio.
env:Receiver	ter:Action	Maximum number	The maximum number of

	ter:MaxNVTProfiles	reached	supported profiles has been reached.
env:Receiver	ter:ActionNotSupported ter:AudioOutputNotSupported	No audio output capability	Audio or Audio Outputs are not supported by the NVT
env:Receiver	ter:ActionNotSupported ter:AudioDecodingNotSupported	No audio decoding capability	Audio or Audio Decoding is not supported by the NVT
env:Receiver	ter:Action ter:IncompleteConfiguration	Configuration not complete	Entities required by this action are missing in the specified profile.
env:Receiver	ter:Action ter:ConfigurationConflict	Conflict when using new settings	The new settings conflicts with other uses of the configuration.
env:Sender	ter:InvalidArgVal ter:ProfileExists	Profile token already exists	A profile with the token ProfileToken already exists.
env:Sender	ter:InvalidArgVal ter:NoConfig	Configuration token does not exist	The requested configuration indicated by the ConfigurationToken does not exist.
env:Sender	ter:InvalidArgVal ter:NoProfile	Profile token does not exist	The requested profile token ProfileToken does not exist.
env:Sender	ter:Action ter:DeletionOfFixedProfile	Fixed profile can not be deleted	The fixed Profile cannot be deleted.
env:Sender	ter:InvalidArgVal ter:ConfigModify	Parameters can not be set	The configuration parameters are not possible to set.
env:Sender	ter:ActionNotSupported ter:VideoAnalyticsNot-Supported	No video analytics capability	NVT does not support video analytics.
env:Sender	ter:InvalidArgVal ter:InvalidStreamSetup	Invalid Stream setup	Specification of StreamType or Transport part in StreamSetup is not supported.
env:Sender	ter:OperationProhibited ter:StreamConflict	Stream conflict	Specification of StreamType or Transport part in StreamSetup causes conflict with other streams.

12 Real time streaming

This section describes real-time streaming of video, audio and metadata. There is *no specific* service associated with the real-time streaming. The real-time configurations via Web Service commands are defined in the Media Service and the ReceiverService.

12.1 Media stream protocol

12.1.1 Transport format

Real-time Transport Protocol (RTP) is a media transfer protocol (see Section 12.1.2). The following four sections describe RTP data transfer.

12.1.1.1 RTP data transfer via UDP

UDP has the smallest overhead and is able to transfer real-time data in an efficient manner. A device shall support the RTP/UDP protocol and the device should support RTP/UDP multicasting.

12.1.1.2 RTP/TCP

If there is a packet loss during media transfer via UDP, then the standard allows for RTP data transfer via TCP as an alternative means of media transport. A device MAY support the RTP/TCP based option. If the device supports the RTP/TCP protocol, then this protocol shall conform to [RFC 4571] (Framing Real-time Transport Protocol and RTP Control Protocol [RTCP] Packets over Connection-Oriented Transport).

12.1.1.3 RTP/RTSP/TCP

The device should support media transfer using RTP/RTSP to traverse a firewall using an RTSP tunnel. This protocol shall conform to [RFC 2326] Section 10.12.

12.1.1.4 RTP/RTSP/HTTP/TCP

The data stream shall be sent via HTTP to traverse a firewall. A device shall support media transfer using RTP/RTSP/HTTP/TCP. And if a device supports TLS1.0, the data stream shall be sent or received via HTTPS to traverse a firewall, and a device shall support media transfer using RTP/RTSP/HTTPS/TCP.

This protocol shall conform to [RFC 2326] (RTSP Section 12.2.1.1: Embedded [Interleaved] Binary Data).

This tunnelling method shall also conform to QuickTime available from Apple Inc. The mandatory parts of the following document shall be implemented by an NVT.

http://developer.apple.com/quicktime/icefloe/dispatch028.html

12.1.2 Media Transport

12.1.2.1 RTP

The Real-time Transport Protocol provides real-time transfer for media streams between two end points. The RTP protocol provides support for re-ordering, de-jittering and media synchronization.

All media streams transferred by the RTP protocol shall conform to [RFC 3550], [RFC 3551], [RFC 3984], [RFC 3016] and JPEG over RTP (see Section 12.1.3).

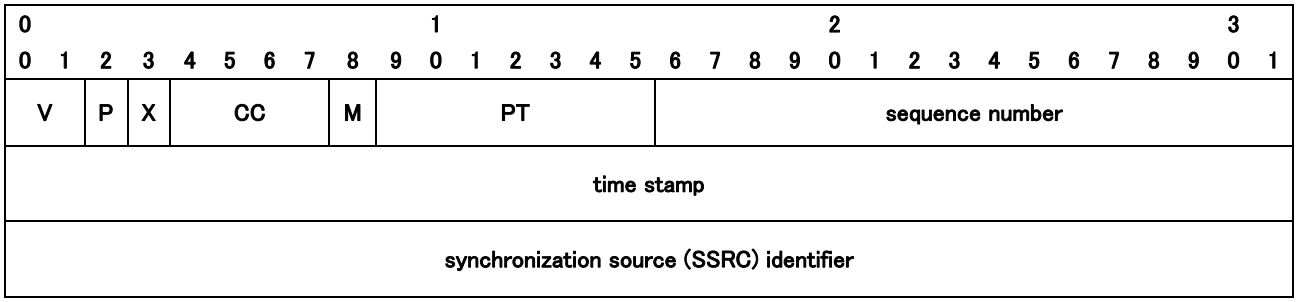


Figure 14: RTP header

An RTP header shall be filled up with following values.

Table 190: RTP header value

Header field	Value	Description
Version (V): 2 bits	2	
Padding (P): 1 bit	0/1	If the payload includes padding octet, this should be set to “1”
Extension (X): 1 bit	0/1	Depends on the use of extension of RTP header. The specification defines two scenarios where a RTP header extension could be used to transmit additional information: 1) “JPEG over RTP” (see Section 12.1.3). 2) Replay (see Section 21) If the header extension is used the Extension bit shall be set.
CSRC count (CC): 4 bits	0	

Marker (M): 1 bit	0/1	The usage shall be conform to related RFCs (e.g. [RFC 3984] for H.264 Video) or to this standard e.g “JPEG over RTP” (see Section 12.1.3) or RTP streaming of metadata (see Section 12.1.2.1.1).
Payload type (PT): 7 bits	See [RFC 3551] Section 6.	
Sequence Number: 16 bits		The initial value of the “sequence number” should be random (unpredictable) to make known-plaintext attacks on encryption more difficult. This number increments by one for each RTP data packet sent
timestamp: 32 bits		The initial value of the “timestamp” should be random (unpredictable) to make known-plaintext attacks on encryption more difficult. See Section 12.1.2.2.1 for further details of Media Synchronization. The usage of the timestamp is dependent on the codec.
SSRC 32 bits		The synchronization source for the data stream. This specification makes no restrictions on the use of this field.

12.1.2.1.1 RTP for Metadata stream

Metadata streams are also transported by RTP. The usage of payload type, marker and timestamp for RTP header for the metadata stream is defined in the following way:

- A dynamic payload type (96-127) shall be used for payload type which is assigned in the process of a RTSP session setup.
- The RTP marker bit shall be set to “1” when the XML document is closed.
- It is RECOMMENDED to use an RTP timestamp representing the creation time of the RTP packet with a RTP clock rate of 90000 Hz. Only UTC timestamps shall be used within the

metadata stream. The synchronization of video and audio data streams is done using RTCP.

The Metadata payload is an XML document with root node `tt:MetaDataStream`. There is no limitation on the size of the XML document. When a synchronization point (see Section 11.18.1) is requested for the stream, the previous XML document shall be closed and a new one started. It is RECOMMENDED to start new XML documents after 1 second, at the longest. The RTP timestamp of the Metadata stream has no specific meaning. The Metadata stream multiplexes Metadata from different sources. This specification defines placeholders for the Scene Description of the Video Analytics, the PTZ Status of the PTZ controller and the Notifications of the Event Configuration. A device can select which of these parts should be multiplexed into the Metadata during the Media Configuration (see Section 11.10). Each part can appear multiple times in arbitrary order within the document. A Metadata connection can be bi-directional using the backchannel mechanism (see Section 12.3).

Metadata stream contains the following elements:

- VideoAnalyticsStream
- PTZStream
- EventStream

The place-holders for the different metadata sources have the following XMLstructure:

```
<xs:complexType name="VideoAnalyticsStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Frame" type="tt:Frame"/>
    ...
  </xs:choice>
</xs:complexType>

<xs:complexType name="PTZStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="PTZStatus"/>
    ...
  </xs:choice>
</xs:complexType>

<xs:complexType name="EventStream">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="wsnt:NotificationMessage"/>
    ...
  </xs:choice>
</xs:complexType>
```

The following is an example of a metadata XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<tt:MetaDataStream xmlns:tt="http://www.onvif.org/ver10/schema">
  <tt:VideoAnalytics>
    <tt:Frame UtcTime="2008-10-10T12:24:57.321">
      ...
    </tt:Frame>
    <tt:Frame UtcTime="2008-10-10T12:24:57.621">
      ...
    </tt:Frame>
  </tt:VideoAnalytics>
</tt:MetaDataStream>

<?xml version="1.0" encoding="UTF-8"?>
<tt:MetaDataStream xmlns:tt="http://www.onvif.org/ver10/schema">
```

```
  <tt:Event>
    <wsnt:NotficationMessage>
      <wsnt:Message>
        <tt:Message UtcTime= "2008-10-10T12:24:57.628">
        ...
        </tt:Message>
      </wsnt:Message>
    </wsnt:NotficationMessage>
  </tt:Event>
</tt:MetaDataStream>
```

12.1.2.2 RTCP

The RTP Control Protocol provides feedback on quality of service being provided by RTP and synchronization of different media streams. The RTCP protocol shall conform to [RFC 3550].

For a feedback request, [RFC 4585] and [RFC 5104] should be supported.

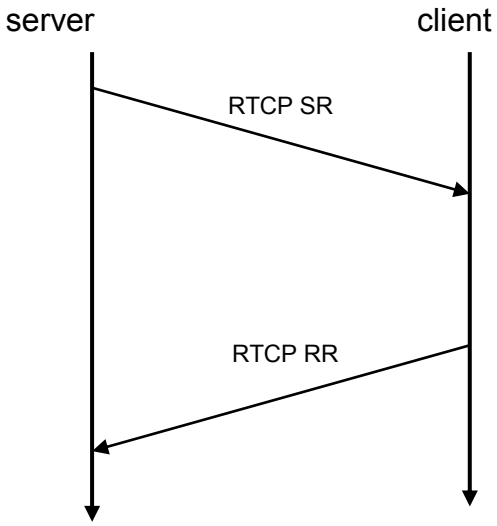


Figure 15: RTCP sequence

12.1.2.2.1 Media synchronization

A client MAY receive audio and video streams simultaneously from more than one device. In this case, each stream uses a different clock (from data acquisition to packet receiving). RTCP Sender Reports (SR) are used to synchronize different media streams. RTCP SRs shall conform to [RFC 3550].

The RTCP Sender Report (SR) packet has fields for the RTP timestamp and for a wall clock timestamp (absolute date and time, 64bit NTP [Network Time Protocol]). See Figure 16.

A device shall support RTCP Sender Report for media synchronization. The client should use RTCP for the media synchronization.

0								1								2								3		
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V			P		RC			PT=SR=200								length															
SSRC of sender																															
NTP timestamp, most significant word																															
NTP timestamp, least significant word																															
RTP timestamp																															
sender's packet count																															
:																															
:																															

Figure 16: RTCP Sender Report

The wall clock should be common in the device and each timestamp value should be determined properly. The client can synchronize different media streams at the appropriate timing based on the RTP clock and wall clock timestamps (see Figure 17).

In case of multiple devices, the NTP timestamp should be common to all devices, and the NTP server should be required in the system [3].

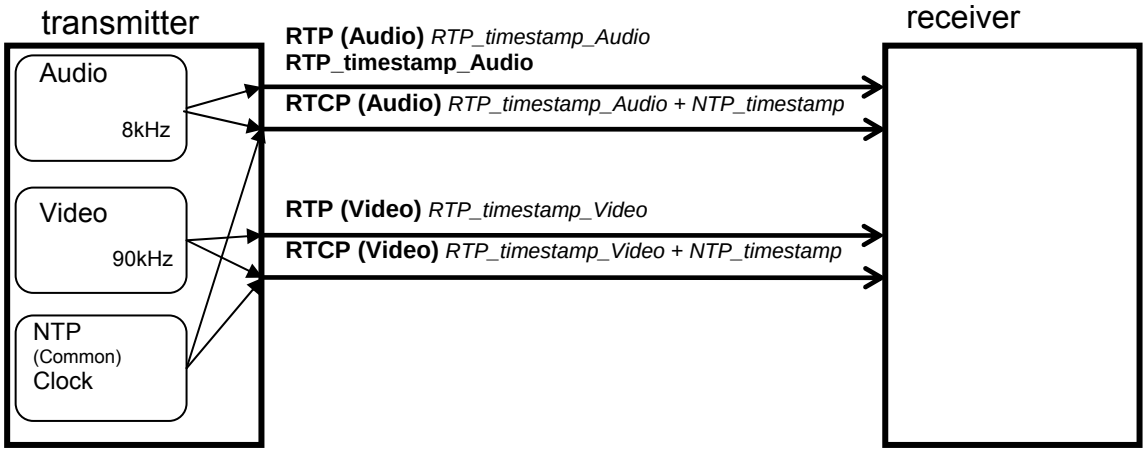


Figure 17: Media Synchronization

12.1.3 Synchronization Point

Synchronization points allow clients to decode and correctly use data after the synchronization point. A synchronization point MAY be requested by a client in case of decoder error (e.g. in consequence of packet loss) to enforce the device to add an I-Frame as soon as possible or to request the current ptz or event status.

In addition to the WebService based methods (see Section 11.18.1 and Section 15.6) a device shall support, this standard RECOMMENDS the PLI messages as described in [RFC 4585] to request an SynchronizationPoint.

[3] The client can get information about "NTP server availability" from the devices by using the GetNTP command. Refer to Section 8.2.5

12.1.4 JPEG over RTP

12.1.4.1 Overall packet structure

The syntax for transmitting JPEG streams follows [RFC 2435]. The syntax does allow embedding additional data, beyond the limits of [RFC 2435], by using an optional RTP header extension, as specified below, with some of the RTP packets. This option, however, changes the exact semantics for frames which include such packets.

The overall format of the JPEG RTP packet is shown in Figure 18.

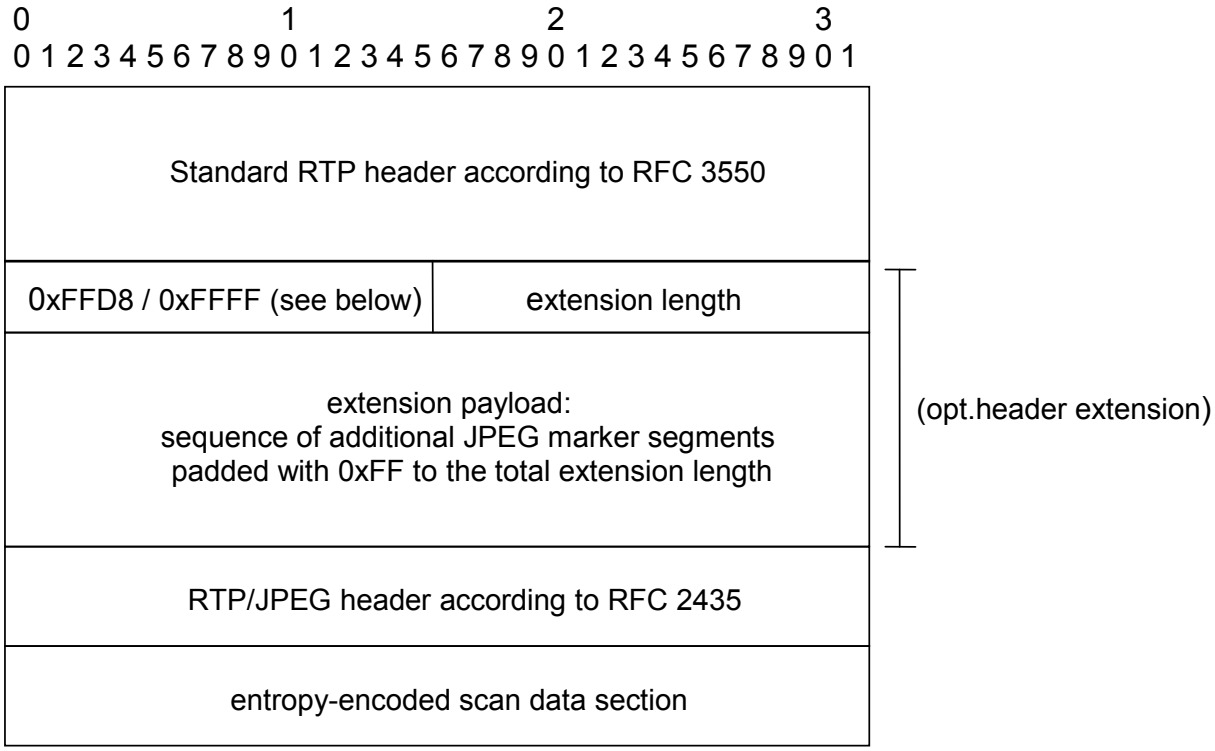


Figure 18: RTP/JPEG packet structure (only the typical content is listed for the extension payload)

In order to distinguish an optional RTP header extension from possible other header extensions, the first 16 bits (the first two octets of the four-octet extension header) of an RTP shall have the value `0xFFD8` (JPEG SOI marker) for the initial packet and `0xFFFF` for other RTP packets within a frame.

As required by [RFC 3550], the presence of the optional header extension shall be signalled via the X-bit of the RTP header. The extension length field within the header extension counts the number of 32-bit items following as extension payloads. For example, a zero-length field following the 32-bit extension header represents an empty header extension).

The entropy-encoded scan data section MAY not be present in all RTP packets. A complete RTP/JPEG header however shall be present in the initial packet of every frame and all packets containing an entropy-encoded scan data section, otherwise it MAY be missing.

The fragment offset field within the RTP/JPEG header, according to [RFC 2435], should be used as if no header extension would be present. Additionally, if a packet does not contain an entropy-encoded scan data segment, but contains a header extension the fragment offset

field shall not be zero if any packets containing an entropy-encoded scan data section for the same frame have been transmitted. If the initial packet of a frame contains no header extension, according to this standard, its fragment offset field shall be zero, otherwise it should be zero. All packets including an RTP/JPEG header with a fragment offset of zero and a Q value between 128-255 shall include a quantization table header according to Section 3.1.8 of [RFC 2435], other packets shall NOT include this header.

12.1.4.2 Logical decoding specification

For the decoding specification, it is assumed that the original packet order within the RTP stream has been restored according to the RTP sequence numbering.

If the initial packet of a frame contains no RTP header extension as specified above, decoders shall generate the complete scan header and perform the decoding as specified by [RFC 2435]. The scan data sections and payloads of any header extension conforming to this specification, up to and including the next RTP packet with its marker bit set, shall be concatenated as they occur within the stream ignoring their fragment offset values.

Otherwise (at least an empty header extension as specified above is present in the initial packet of a frame), the following rules apply for each such frame:

- If the initial packet of a frame does not contain an entropy-encoded scan data segment, but contains a header extension as specified above, then decoders shall concatenate its header extension payload with (possibly empty or not existing) header extension payload(s) conforming to this specification of the subsequent packets up to and including the first packet with the RTP marker bit set or containing an entropy-encoded scan data segment.

- The concatenated initial RTP header extension payload (sequence) shall be logically prepended with a JPEG SOI marker (0xFFD8).

- If the Q-value of the RTP/JPEG scan header within the initial packet of a frame is not zero, the quantization tables shall be pre-initialized according to the rules of [RFC 2435]. If Q is equal to zero the quantization tables shall be copied from the previous frame, allowing for DQT markers within this initial header extension payload (sequence) to override them.

- If this frame is the initial frame of a sequence, the Huffman tables shall be pre-initialized according to [RFC 2435]. The Huffman tables for all subsequent frames shall be copied from the previous frame, allowing the frames to be overridden by DHT markers within the initial header extension payload (sequence).

- If the initial RTP header extension payload (sequence) supplies no DRI marker, but the RTP/JPEG header of the initial packet of a frame contains an RTP/JPEG restart marker, a DRI marker corresponding to the rules of [RFC 2435] shall be appended to the initial header extension payload (sequence). Otherwise, if the initial RTP header extension (sequence) supplies a DRI marker, the marker shall take precedence over any other RTP/JPEG restart marker according to [RFC 2435] for the same frame. However, for compatibility with decoders conforming to [RFC 2435] only, encoders normally should use an RTP/JPEG restart marker with consistent values, if restart intervals are to be used.

- DRI markers shall NOT be derived from previous frames.

- If the initial RTP header extension payload (sequence) supplies no SOF marker, which otherwise takes precedence, a SOF marker shall be appended to it with the following values:

- If both the width and height field of the RTP/JPEG header are zero, the SOF marker of the previous frame shall be used.

- Otherwise it shall be derived according to the rules of [RFC 2435].

However, as long as the (rounded up) image size fits within the range as specified in [RFC 2435], encoders should specify the image size within the RTP/JPEG header consistent with the values of an additional SOF header.

- If the initial header extension payload (sequence) supplies no SOS marker, a corresponding marker shall be derived according to [RFC 2435] and appended to it, otherwise the SOS marker in the extension takes precedence.

  An SOS marker shall NOT be derived from previous frames.

  If the SOS marker is present and not followed by entropy-encoded scan data within the extension, the marker shall be the final marker within the initial extension payload (sequence) of a frame. Necessary padding with 0xFF-octets shall NOT follow this marker but MAY precede it.

- The remaining entropy-encoded scan data and header extensions payloads shall be logically appended in the same order as they occur within the RTP stream up to the end of the frame as indicated by the RTP marker bit. A final EOI marker shall also be added if it is not yet present within the logical sequence for this frame,.

  For each frame, the resulting sequence up to and including the first (possibly added) EOI marker shall be a valid (possibly abbreviated) JPEG stream, resulting in one complete image from the decoding process for this frame. The meaning of any data after this first EOI marker for each frame is outside the scope of this specification.

12.1.4.3 Supported colour spaces and sampling factors

A Transmitter should use only greyscale and YCbCr colour space. A Client shall support both greyscale and YCbCr.

The sampling factors for YCbCr shall correspond to the values supported by [RFC 2435]. For example, a sampling factor of 4:2:0 (preferred) or 4:2:2.

12.1.4.4 Pixel aspect ratio handling

The pixel aspect ratio of JPEG files can be specified within the JFIF marker. If the pixel aspect ratio is different from the standard 1:1 and 1:2 ratio according to [RFC 2435], this marker should be transmitted in the initial header extension payload (sequence) of every frame to specify the (for interlaced material field-based) pixel aspect ratio.

12.1.4.5 Interlaced handling

Interlaced video is encoded as two independent fields and signalled as specified by [RFC 2435] within the RTP/JPEG header.

Both fields shall use the same colour space, sampling factors and pixel aspect ratio.

Interlaced encoding should NOT be used if the frame was originally scanned progressively.

12.2 Media control protocol

12.2.1 Stream control

The media stream is controlled using the protocol defined in the URI. The URI is returned in response to the GetStreamUri command defined in Section 11.15.1.

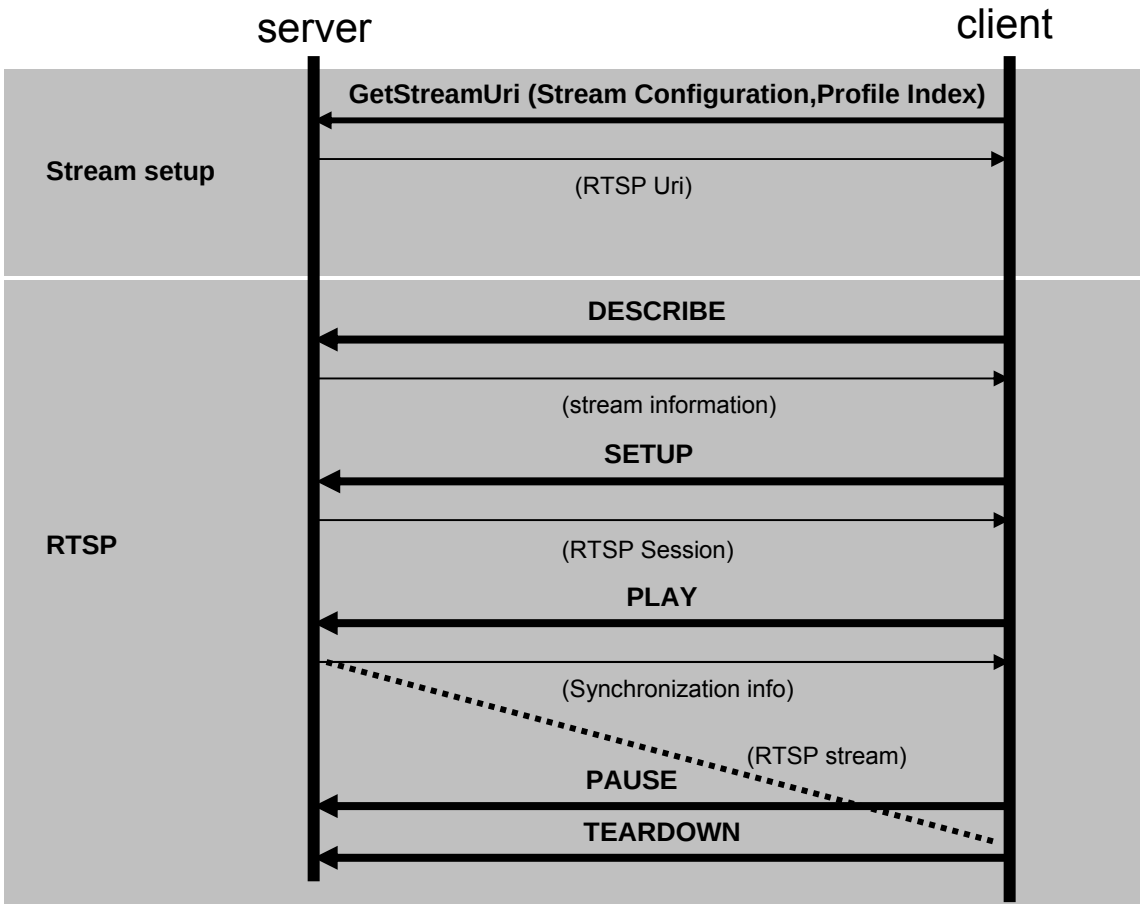


Figure 19: Stream Control

12.2.1.1 RTSP

All devices and clients shall support RTSP ([RFC 2326]) for session initiation and playback control. RTSP shall use TCP as its transport protocol, the default TCP port for RTSP traffic is 554. The Session Description Protocol (SDP) shall be used to provide media stream information and SDP shall conform to [RFC 4566].

Table 191: RTSP methods.

Method	Direction	SPEC[4]	Description
OPTIONS	R->T T->R	M X	Required to get optional method capability and to allow different versions in the future.
DESCRIBE	R->T	M	Required to retrieve media parameters within the designated profile.
ANNOUNCE	R->T T->R	X	
SETUP	R->T	M	Required to set media session parameters.
PLAY	R->T	M	Required to start media stream.
PAUSE	R->T	O	Required to temporarily stop media stream. Handling multiple streams in a narrow bandwidth network, by suspending RTP stream, the traffic can be well controlled by reducing redundant data and congested network traffic can be avoided.
TEARDOWN	R->T	M	Required to release a media session.
GET_PARAMETER	R->T T->R	O	
SET_PARAMETER	R->T T->R	O O	An optional method to keep an RTSP session alive (R->T direction only).
REDIRECT	T->R	X	
RECORD	R->T	X	

12.2.1.1.1 Keep-alive method for RTSP session

A RTSP client keeps the RTSP Session alive and prevents it from session timeout (see [RFC 2326) Section 12.37). This specification recommends the following methods to keep RTSP alive for both Unicast and Multicast streaming.

1) The client can optionally set the Timeout parameter (in seconds) using the `Set<configurationEntity>EncoderConfiguration` command defined in Section 11.2, otherwise a default value of ”60” is used.

2) In all RTSP SETUP responses, a transmitter should include the Timeout value according to [RFC 2326] Section 12.37 and the transmitter should use the Timeout value for keep-alive.

4 X: Not supported, M: Mandatory, O: Optional

3) To keep the RTSP Session alive, a client shall call the RTSP server using any RTSP method or send RTCP receiver reports. SET_PARAMETER is the RECOMMENDED RTSP method to use.

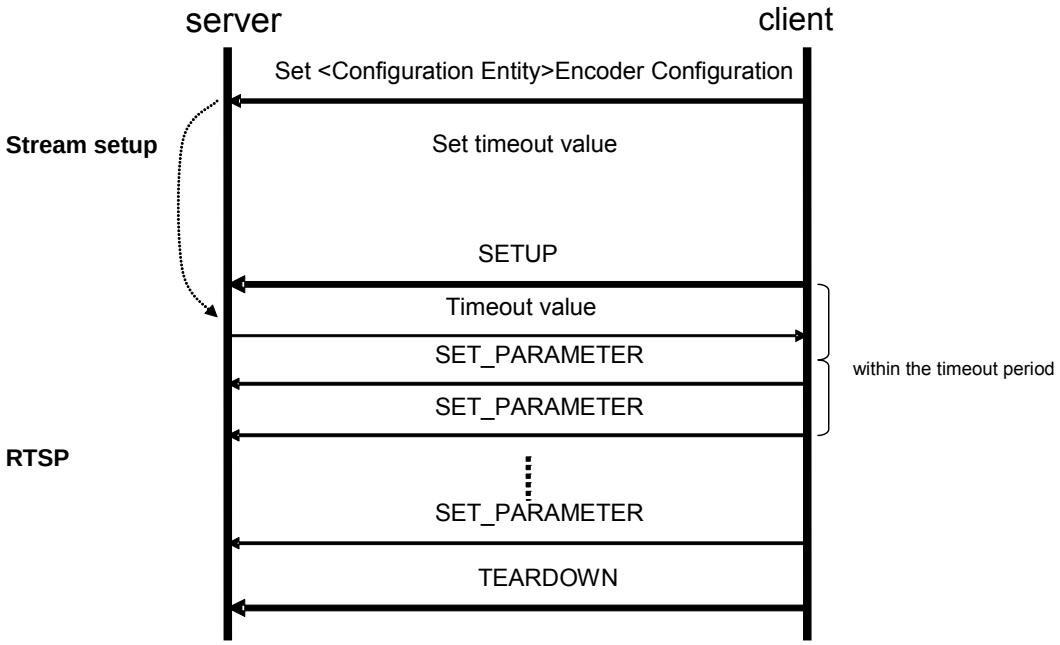


Figure 20: Keep Alive

12.2.1.1.2 RTSP Audio and Video Synchronization

In order that clients may immediately begin synchronizing video and audio streams, and computing absolute UTC timestamps for incoming packets for recording purposes, a transmitter should include the following header fields in the RTSP PLAY response:

- Range ([RFC 2326] section 12.29). This SHALL include a start time in clock units ([RFC 2326] section 3.7), *not* SMPTE or NPT units.

- RTP-Info ([RFC 2326] section 12.33). This SHALL include an rtptime value which corresponds to the start time specified in the Range header.

Example:

```
client->server:   PLAY rtsp://example.com/onvif_camera/video RTSP/1.0
                  CSeq: 4
                  Range: npt=now-
                  Session: 12345678

server->client:         RTSP/1.0 200 OK
                  CSeq: 4
                  Session: 12345678
                  Range: 20100217T143720.257Z-
                  RTP-Info: url=rtsp://example.com/onvif_camera/video;
seq=1234;rtptime=3450012
```

12.2.1.1.3 RTSP session for a Metadata stream

In the case of a metadata stream, the SDP description “application” should be used in the DESCRIBE response for media type and “vnd.onvif.metadata” should be used for encoding a name.

Example RTSP DESCRIBE message exchange between an RTSP Server (server) and an RTSP client (client):

```
        client->server:  DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
                         CSeq: 1

    server->client:      RTSP/1.0 200 OK
                         CSeq: 1
                         Content-Type: application/sdp
                         Content-Length: XXX
                         v=0
                         o=- 2890844256 2890842807 IN IP4 172.16.2.93
                         s=RTSP Session
                         m=audio 0 RTP/AVP 0
                         a=control:rtsp://example.com/onvif_camera /audio
                         m=video 0 RTP/AVP 26
                         a=control:rtsp://example.com/onvif_camera /video
                         m=application 0 RTP/AVP 107
                         a=control:rtsp://example.com/onvif_camera/metadata
                         a=recvonly
                         a=rtpmap
                         a=rtpmap:107 vnd.onvif.metadata/90000
```

12.2.1.1.4 RTSP message example

This example shows the message transfer between an RTSP client (client) and an RTSP server (server). The client requests one audio and one video stream from the device. The Stream Uri “rtsp://example.com/onvif_camera” can be retrieved using the GetStreamUri command. Refer to Section 11.15.1.

```
      client->server:    DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
                         CSeq: 1

      server->client:    RTSP/1.0 200 OK
                         CSeq: 1
                         Content-Type: application/sdp
                         Content-Length: XXX
                         v=0
                         o=- 2890844256 2890842807 IN IP4 172.16.2.93
                         s=RTSP Session
                         m=audio 0 RTP/AVP 0
                         a=control:rtsp://example.com/onvif_camera/audio
                         m=video 0 RTP/AVP 26
                         a=control:rtsp://example.com/onvif_camera/video

      client->server:    SETUP rtsp://example.com/onvif_camera/audio RTSP/1.0
                         CSeq: 2
                         Transport: RTP/AVP;unicast;client_port=8002-8003

      server->client:    RTSP/1.0 200 OK
                         CSeq: 2
                         Transport: RTP/AVP;unicast;client_port=8002-8003;
                                      server_port=9004-9005
                         Session: 12345678; timeout=60

      client->server:    SETUP rtsp://example.com/onvif_camera/video RTSP/1.0
                         CSeq: 3
                         Transport: RTP/AVP;unicast;client_port=8004-8005
                         Session: 12345678

      server->client:    RTSP/1.0 200 OK
                         CSeq: 3
                         Transport: RTP/AVP;unicast;client_port=8004-8005;
                         server_port=9006-9007
                         Session: 12345678; timeout=60

      client->server:    PLAY rtsp://example.com/onvif_camera RTSP/1.0
```

```
                    CSeq: 4
                    Range: npt=now-
                    Session: 12345678

server->client:     RTSP/1.0 200 OK
                    CSeq: 4
                    Session: 12345678
                    RTP-Info: url=rtsp://example.com/onvif_camera/video;
                    seq=1234;rtptime=3450012,
                    url=rtsp://example.com/onvif_camera/audio;
                    seq=22434;rtptime=1234566

client->server:     TEARDOWN rtsp://example.com/onvif_camera RTSP/1.0
                    CSeq: 5
                    Session: 12345678

server->client:     RTSP/1.0 200 OK
                    CSeq: 5
                    Session: 12345678
```

12.2.1.2 RTSP over HTTP

The RTSP over HTTP/HTTPS shall be supported in order to traverse a firewall. See Section 12.1.1.4 RTP/RTSP/HTTP/TCP.

12.3 Back Channel Connection

This section describes how a bidirectional connection can be established between a client and a server. The backchannel connection handling is done using RTSP [RFC 2326]. Therefore a mechanism is introduced which indicates that a client wants to built up a backchannel connection. RTSP provides feature-tags to deal with such functionality additions.

A device that supports bi-directional connections (e.g audio or metadata connections) shall support the introduced RTSP extensions.

12.3.1 RTSP Require- Tag

The RTSP standard [RFC 2326] can be extended by using additional headers objects. For that purpose a Require tag is introduced to handle special functionality additions (see [RFC 2326], 1.5 Extending Rtsp and 12.32 Require).

The Require-tag is used to determine the support of this feature. This header shall be included in any request where the server is required to understand that feature to correctly perform the request.

A device that supports backchannel shall understand the backchannel tag:

- `www.onvif.org/ver20/backchannel`

An RTSP client that wants to built up an RTSP connection with a data backchannel shall include the Require header in its requests.

12.3.2 Connection setup for a bi- directional connection

A client shall include the feature tag in it's DESCRIBE request to indicate that a bidirectional data connection shall be established.

A server that understands this Require tag shall include an additional media stream in its SDP file as configured in its Media Profile.

An RTSP server that does not understand the backchannel feature tag or does not support bidirectional data connections shall respond with an error code *551 Option not supported* according to the RTSP standard. The client can then try to establish an RTSP connection without backchannel.

A SDP file is used to describe the session. The server shall include the a=sendonly or the a=recvonly attributes in each media section of the SDP file to indicate the direction the media data will be send.

The server shall list all supported decoding codecs as own media section and the client chooses which one is used.

12.3.2.1 Example 1: Server without backchannel support:

```
Client – Server:        DESCRIBE rtsp://192.168.0.1 RTSP/1.0
                        CSeq: 1
                        User-Agent: ONVIF Rtsp client
                        Accept: application/sdp
                        Require: www.onvif.org/ver20/backchannel

Server – Client:        RTSP/1.0 551 Option not supported
                        CSeq: 1
                        Unsupported: www.onvif.org/ver20/backchannel
```

12.3.2.2 Example 2: Server with Onvif backchannel support:

```
Client – Server:        DESCRIBE rtsp://192.168.0.1 RTSP/1.0
                        CSeq: 1
                        User-Agent: ONVIF Rtsp client
                        Accept: application/sdp
                        Require: www.onvif.org/ver20/backchannel

Server – Client:        RTSP/1.0 200 OK
                        CSeq: 1
                        Content-Type: application/sdp
                        Content-Length: xxx

                        v=0
                        o= 2890842807 IN IP4 192.168.0.1
                        s=RTSP Session with audiobackchannel
                        m=video 0 RTP/AVP 26
                        a=control:rtsp://192.168.0.1/video
                         a=recvonly
                         m=audio 0 RTP/AVP 0
                        a=control:rtsp://192.168.0.1/audio
                        a=recvonly
                        m=audio 0 RTP/AVP 0
                        a=control:rtsp://192.168.0.1/audioback
                        a=rtpmap:0 PCMU/8000
```

```
                        a=sendonly
```

This SDP file completely describes the RTSP session. The Server gives the client its control URLs to setup the streams.

In the next step the client can setup the sessions:

```
Client – Server:          SETUP rtsp://192.168.0.1/video RTSP/1.0
                          CSeq: 2
                          Transport: RTP/AVP;unicast;client_port=4588-4589

Server – Client:    RTSP/1.0 200 OK
                          CSeq: 2
                          Session: 123124;timeout=60
                          Transport:RTP/AVP;unicast;client_port=4588-4589;
                          server_port=6256-6257

Client – Server:          SETUP rtsp://192.168.0.1/audio RTSP/1.0
                          CSeq: 3
                          Session: 123124
                          Transport: RTP/AVP;unicast;client_port=4578-4579

Server – Client:          RTSP/1.0 200 OK
                          CSeq: 3
                          Session: 123124;timeout=60
                          Transport:RTP/AVP;unicast;client_port=4578-4579;
                          server_port=6276-6277

Client – Server:          SETUP rtsp://192.168.0.1/audioback RTSP/1.0
                          CSeq: 4
                          Session: 123124
                          Transport: RTP/AVP;unicast;client_port=6296-6297
                          Require: www.onvif.org/ver20/backchannel

Server – Client:          RTSP/1.0 200 OK
                          CSeq: 4
                          Session: 123124;timeout=60
                          Transport:RTP/AVP;unicast;client_port=6296-6297;
                          server_port=2346-2347
```

The third setup request establishes the audio backchannel connection.

In the next step the client starts the session by sending a PLAY request.

```
Client – Server:          PLAY rtsp://192.168.0.1 RTSP/1.0
                          CSeq: 5
                          Session: 123124
                          Require: www.onvif.org/ver20/backchannel

Server – Client:          RTSP/1.0 200 OK
                          CSeq: 5
                          Session: 123124;timeout=60
```

After receiving the OK response to the PLAY request the client MAY start sending audio data to the server. It shall not start sending data to the server before it has received the response.

The Require-header indicates that a special interpretation of the PLAY command is necessary. The command covers both starting of the video and audio stream from NVT to the client and starting the audio connection from client to server.

To terminate the session the client sends a TEARDOWN request.

```
Client – NVT:       TEARDOWN rtsp://192.168.0.1 RTSP/1.0
                    CSeq: 6
                    Session: 123124
                    Require: www.onvif.org/ver20/backchannel

NVT – Client:       RTSP/1.0 200 OK
                    CSeq: 6
                    Session: 123124
```

12.3.3 Multicast streaming

If the client intents to send its data in multicast it uses the transport parameter in the SETUP request to tell the server the multicast address and port.

12.3.3.1 Example: Multicast Setup

```
Client – Server:          SETUP rtsp://192.168.0.1/audioback RTSP/1.0
                          CSeq: 4
                          Session: 123124
                          Transport:RTP/AVP;multicast;destination=224.2.1.1;port=60
                          000-60001;ttl=128
                          Require: www.onvif.org/ver20/backchannel

Server – Client:          RTSP/1.0 200 OK
                          CSeq: 4
                          Session: 123124;timeout=60
                          Transport:RTP/AVP;multicast;destination=224.2.1.1;port=60
                          000-60001;ttl=128;mode=”PLAY”
```

12.4 Error Handling

RTSP and HTTP protocol errors are classified into different categories (for example, status codes 1xx, 2xx, 3xx, 4xx and 5xx respectively). The device and the client shall support and handle these status codes. For RTSP status code definitions refer to [RFC 2326], Section 11.0. For HTTP status code definitions refer HTTP/1.1 [RFC 2616], Section 10.0

13 Receiver Configuration

This service offers commands to manage Receiver objects, which are used to receive media streams from other devices. A Receiver object contains the information how to setup the stream, the mode of the receiver and the Stream Uri (MediaUri). A device shall at least support Media Uris of 128 octet length. The Receiver - MaximumRTSPURILength capability indicates the maximum length supported by the device. The Receiver Service shall be implemented by devices that can receive media streams.

The IP or DNS address in the transmit URI given to the receiver, is the address that the device hosting the receiver service will use to access the transmit device. If, for example, the client has to communicate through a NAT router to access the transmitter and the receiver, the transmitter address that the client gives the receiver (in this case a local network address) may not be the same address that the client would use to access the transmitter (in this case an external network address).

A device shall support RTP transfer via RTP, see section 12.1.1.1, and RTP transfer via RTSP/HTTP/TCP, see section 12.1.1.4. A device may support other RTP transport protocols and shall indicate what it supports with the appropriate capability, see Receiver category in Table 11.

13.1 Persistence

All the objects created within the receiver service shall be persistent – i.e. they shall survive a power cycle. Likewise, all the configuration data in the objects shall be persistent.

13.2 Receiver modes

A receiver can operate in three distinct modes:

Always Connect. The receiver attempts to maintain a persistent connection to the configured endpoint.

Never Connect. The receiver does not attempt to connect.

Auto Connect. The receiver connects on demand, as required by consumers of the media streams.

13.3 Receiver commands

This section describes the commands offered by the Receiver Service.

13.3.1 Get Receivers

This operation lists all receivers that currently exist on the device. The Receiver Service shall support this command.

Table 192: GetReceivers command

GetReceivers	
Message name	**Description**
GetReceiversRequest	*This message is empty.*

GetReceiversResponse	*Contains a list of receivers.* tt:Receiver **Receivers** [0][unbounded]
Fault codes	**Description**
No specific fault codes.	

13.3.2 Get Receiver

This operation retrieves the details of a specific receiver whose token is known to the client. The Receiver Service shall support this command.

Table 193: GetReceiver command

GetReceiver	
Message name	**Description**
GetReceiverRequest	*Contains the token of the requested receiver.* tt:ReceiverToken **ReceiverToken** [1][1]
GetReceiverResponse	*Contains the details of the requested receiver.* tt:Receiver **Receiver** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UnknownToken	*The receiver indicated by **ReceiverToken** does not exist.*

13.3.3 Create Receiver

This operation creates a new receiver. The Receiver Service shall support this command.

Table 194: CreateReceiver command

CreateReceiver	
Message name	**Description**
CreateReceiverRequest	*Contains the initial configuration of the receiver.* tt:ReceiverConfiguration **Configuration** [1][1]
CreateReceiverResponse	*Contains the details of the receiver that was created.* tt:Receiver **Receiver** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter: InvalidStreamSetup	*The specified configuration is invalid.*
env:Receiver ter:Action ter:MaxReceivers	*The maximum supported number of receivers has been reached.*

13.3.4 Delete Receiver

This operation deletes an existing receiver. A receiver MAY NOT be deleted if it is in use. The Receiver Service shall support this command.

Table 195: DeleteReceiver command

DeleteReceiver	
Message name	**Description**
DeleteReceiverRequest	*Contains the token of the receiver to be deleted.* tt:ReceiverToken **ReceiverToken** [1][1]
DeleteReceiverResponse	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UnknownToken	*The receiver indicated by **ReceiverToken** does not exist.*
env: Receiver ter:Action ter:CannotDeleteReceiver	*It is not possible to delete the specified receiver, for example because it is currently in use.*

13.3.5 Configure Receiver

This operation configures a receiver. The Receiver Service shall support this command.

Table 196: ConfigureReceiver command

ConfigureReceiver	
Message name	**Description**
ConfigureReceiverRequest	*Contains the token of the requested receiver and the new configuration.* tt:ReceiverToken **ReceiverToken** [1][1] tt:ReceiverConfiguration **Configuration** [1][1]
ConfigureReceiverResponse	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UnknownToken	*The receiver indicated by **ReceiverToken** does not exist.*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The specified configuration is invalid.*

13.3.6 SetReceiverMode

This operation may be used to set the mode of the receiver independently of the rest of its configuration. The Receiver Service shall support this command.

Table 197: SetReceiverMode command

SetReceiverMode	
Message name	**Description**
SetReceiverModeRequest	*Contains the token of the requested receiver and the new mode.* tt:ReceiverToken **ReceiverToken** [1][1] tt:ReceiverMode **ReceiverMode** [1][1]
SetReceiverModeResponse	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UnknownToken	*The receiver indicated by **ReceiverToken** does not exist.*

13.3.7 GetReceiverState

This operation determines whether the receiver is currently disconnected, connected or attempting to connect. The Receiver Service shall support this command.

Table 198: GetReceiverState command

GetReceiverState	
Message name	**Description**
GetReceiverStateRequest	*Contains the token of the requested receiver.* tt:ReceiverToken **ReceiverToken** [1][1]
GetReceiverStateResponse	*Contains the current state of the receiver.* tt:ReceiverState **State** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:UnknownToken	*The receiver indicated by **ReceiverToken** does not exist.*

13.4 Events

The receiver service shall dispatch events through the event service. It shall be capable of generating the events listed in this chapter whenever the condition that fires the event occurs.

13.4.1 ChangeState

Whenever a receiver changes state, the device shall dispatch the following event:

```
Topic: tns1: Receiver/ChangeState
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="ReceiverToken" Type="tt:ReceiverToken"/>
  </tt:Source>
  <tt:Data>
```

```
      <tt:SimpleItemDescription Name="NewState" Type="tt:ReceiverState"/>
      <tt:SimpleItemDescription Name="MediaUri" Type="tt:MediaUri" minOccurs="0"/>
    </tt:Data>
  </tt:MessageDescription>
```

13.4.2 Connection Failed

If a receiver fails to establish a connection, the device shall dispatch the following event:

```
Topic: tns1: Receiver/ConnectionFailed
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="ReceiverToken" Type="tt:ReceiverToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="MediaUri" Type="tt:MediaUri"/>
  </tt:Data>
</tt:MessageDescription>
```

13.5 Service specific fault codes

Table 199 lists the display service specific fault codes. Additionally, each command can also generate a generic fault, see Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 199: Service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
Env:Sender	ter:InvalidArgVal ter:UnknownToken	Receiver does not exist.	The receiver indicated by ReceiverToken does not exist.
Env:Sender	ter:Action ter:MaxReceivers	Maximum number of receivers has been reached.	The maximum supported number of receivers has been reached.
Env:Sender	ter:InvalidArgVal ter:InvalidStreamSetup	The StreamSetup is not supported.	Specification of StreamType or Transport part in ReceiverConfiguration StreamSetup is not supported.
Env:Sender	ter:Action ter:CannotDeleteReceiver	It is not possible to delete the receiver.	It is not possible to delete the specified receiver, for example because it is currently in use.

14 Display Service

A display device has a fixed number of video outputs, each of which may be attached to a monitor. A client can request the video outputs of the device using the DeviceIO service. Each of these outputs is configured with a layout (e.g. single view or split screen). The layout defines a number of video panes, each of which occupies an area of the physical display.

A network video display MAY also have fixed number of audio inputs and audio outputs. Each of these outputs MAY be associated with a pane. Associating an audio input or output with a pane maps the audio and video streams from a transmitter device automatically to the correct outputs. The pane also contains a pointer to a receiver where the necessary information to connect the display device to a transmitter is stored.

The Display Service offers functions to configure the Panes and describe and change the layout of the display device. The possible layouts and coding capabilities of a video output can be requested.

A display device shall support the Display Service as defines in [DisplayService.wsdl].

14.1 Panes

A Pane is a display area on the monitor that is attached to a video output. A pane has a PaneConfiguration that describes which entities are associated with the pane. The PaneConfiguration includes:

Pane Token: **A** unique identifier in the display device.

Pane Name: Configuration name.

AudioOutputToken: A pointer to the audio output that is associated with the pane. A client can retrieve the available audio outputs of a device using the GetAudioOutputs command of the DeviceIO service.

AudioSourceToken: A pointer to the audio source that is associated with this pane. The audio connection from a display device to the NVT is established using the backchannel mechanism. A client can retrieve the available audio sources of a device using the GetAudioSources command of the DeviceIO service.

AudioEncoderConfiguration: The configuration of the audio encoder including codec, bitrate and sample rate.

ReceiverToken: A pointer to a Receiver that has the necessary information to receive Data from a Transmitter. This Receiver can be connected and the network video display displays the received data on the specified outputs. A client can retrieve the available Receivers using the GetReceivers command of the Receiver Service.

A client has to configure the pane according to the connection to be established by setting the AudioOutput and/or AudioSourceToken. If a Token is not set, the corresponding session will not be established.

Changing the PaneConfiguration or the parameters of a referenced receiver shall not affect the RTSP connection. If a client intends to apply the new parameters it shall restart the RTSP connection.

The pane layout (see 14.2) of the video output defines if and where (position, size) a Pane is currently visible. The receiver shall only establish a RTSP connection to receive data if the pane is visible. Layout changes shall NOT affect running streams.

14.1.1 GetPaneConfigurations

This command list all currently defined Panes of a device for a specified video output (regardless if this pane is visible at a moment). A display device shall support the retrieval of its configured panes through this command.

Table 200: GetPaneConfigurations

GetPaneConfigurations	Request-Response
Message name	**Description**
GetPaneConfigurationsRequest	*The **VideoOutput** element specifies the Video Output whose PaneConfigurations are requested.* tt:ReferenceToken **VideoOutput**[1][1]
GetPaneConfigurationsResponse	*Contains a list of defined Panes of the specified VideoOutput. Each VideoOutput has at least one PaneConfiguration.* tt:PaneConfiguration **PaneConfiguration** [1][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*

14.1.2 GetPaneConfiguration

If the pane token is already known this command can be used to get the pane configuration. A display device shall support the retrieval of a specific pane configuration through this command.

Table 201: GetPaneConfiguration

GetPaneConfiguration	Request-Response
Message name	**Description**
GetPaneConfigurationRequest	*This message contains the a token of the pane whose configuration is requested. It also conatins a **VideoOutput** token that specifies the Video Output that contains the requested pane.* *This message also contains the token of the requested pane* tt:ReferenceToken **VideoOutput**[1][1] tt:ReferenceToken **Pane**[1][1]

GetPaneConfigurationResponse	*Contains the requested PaneConfiguration* tt:PaneConfiguration **PaneConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoPane	*The requested pane does not exist*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*

14.1.3 SetPaneConfigurations

This command changes the Configuration of all existing panes in one step. The message contains all PaneConfigurations (modified and not modified ones) of the video output. A display device shall support the modification of its panes configuration through this command.

Table 202: SetPaneConfigurations

SetPaneConfigurations	Request-Response
Message name	**Description**
SetPaneConfigurationsRequest	*This message contains the configuration of all panes of the specified VideoOutput.* *The PaneConfiguration element contains the modified configuration.* tt:ReferenceToken **VideoOutput**[1][1] tt:PaneConfiguration **PaneConfiguration**[1][unbounded]
SetPaneConfigurationsResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:invalidConfig	*The configuration is not possible to set*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*

14.1.4 SetPaneConfiguration

This command changes the Configuration of a single pane. A display device shall support the modification of a single pane configuration through this command.

Table 203: SetPaneConfiguration

SetPaneConfiguration	Request-Response
Message name	**Description**
SetPaneConfigurationRequest	*This message contains the token of the video output and the new PaneConfiguration .* tt:ReferenceToken **VideoOutput**[1][1] tt:PaneConfiguration **PaneConfiguration**[1][1]
SetPaneConfigurationResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoPane	*The requested pane does not exist*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*
env:Sender ter:InvalidArgVal ter:InvalidConfig	*The configuration is not possible to set*

14.1.5 CreatePaneConfiguration

This command creates a pane configuration. A display device that supports dynamic creation of panes shall support the creation of a pane configuration through this command.

Table 204: CreatePaneConfiguration

CreatePaneConfiguration	Request-Response
Message name	**Description**
CreatePaneConfigurationRequest	*This message contains the token of the video output and the new PaneConfiguration* tt:ReferenceToken **VideoOutput**[1][1] tt:PaneConfiguration **PaneConfiguration**[1][1]
CreatePaneConfigurationResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:MaxNumberOfPane	*The maximum number of panes is reached*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*
env:Sender ter:InvalidArgVal ter:InvalidConfig	*The configuration is not possible to set*

14.1.6 DeletePaneConfiguration

This command deletes a pane configuration. A display device that supports dynamic deletion of panes shall support the deletion of a pane configuration through this command.

Table 205: DeletePaneConfiguration

DeletePaneConfiguration	Request-Response
Message name	**Description**
DeletePaneConfigurationRequest	*This message contains the token of the video output and the new PaneConfiguration.* tt:ReferenceToken **VideoOutput**[1][1] tt:ReferenceToken **PaneToken**[1][1]
DeletePaneConfigurationResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:FixedPane	*It is not possible to delete this pane configuration*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output does not exist*
env:Sender ter:InvalidArgVal ter:NoPane	*The requested pane configuration does not exist.*

14.2 Layout

The Layout assigns a pane configuration to a certain area of the display. The layout settings directly affect a specific video output. The layout consists of a list of PaneConfigurations and their associated display areas. If the device supports overlapping panes then the order the panes are displayed on the monitor is defined by the order of the PaneConfigurations in the list. The first Pane in the list is the one that is displayed in the foreground.

A device MAY either provide a fixed number of supported layouts or it is possible to configure the layout free.

14.2.1 GetLayout

This command returns the current layout of a video output. A display device shall support the retrieval of its layout through this command.

Table 206: GetLayout

GetLayout	Request-Response
Message name	**Description**
GetLayoutRequest	*Contains the VideoOutputToken of the output the display is connected to.* tt:ReferenceToken **VideoOutput**[1][1]
GetLayoutResponse	*Contains the current layout of the VideoOutput* tt:Layout **Layout**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output indicated with **VideoOutput** does not exist.*

14.2.2 SetLayout

The SetLayout operation can be used to change the layout of a display (e.g. change from single view to split screen view). A display device shall support the change of layout through this command.

Table 207: SetLayout

SetLayout	Request-Response
Message name	**Description**
SetLayoutRequest	*This message contains the token of the video output and the modified layout* tt:ReferenceToken **VideoOutput** [1][1] tt:Layout **Layout**[1][1]
SetLayoutResponse	*This message is an empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidLayout	*It is not possible to set the layout*
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested Video Output indicated with **VideoOutput** does not exist.*

14.3 Display Options

The Display Options contain the supported layouts (LayoutOptions) and the decoding and encoding capabilities (CodingCapabilities) of the device. The GetDisplayOptions command returns both, Layout and Coding Capabilities, of a VideoOutput.

LayoutOptions

The LayoutOptions describe the fixed and predefined layouts of a device. If the device does not offer fixed layouts and allows setting the layout free this element is empty.

Coding Capabilities

The network video display is able to decode audio and video streams using suitable decoding algorithms. The network video display supports any audio and video decoders, bitrates and resolution according to the manufacturer‘s choice.

In order to ensure interoperability between the different devices, this specification mandates the following decoder profiles:

The NVD shall support JPEG QVGA.

The NVD shall support G.711µ Law (if it supports audio)

These are the same codecs that are mandatory for the NVT.

There are no parameters to configure a decoder; a decoder shall decode all content (according to its capabilities) it receives. In case of decoding errors the decoder should try to request a synchronization point and try to continue decoding. It shall generate an event as defined in section 14.4.

The CodingCapabilities Element gives an indication about the decoding and encoding capabilities of the device.

14.3.1 GetDisplayOptions

This command lists layout and coding capabilities of a video output. A display device shall support the retrieval of its DisplayOptions through this command.

Table 208: GetDisplayOptions

GetDisplayOptions	Request-Response
Message name	**Description**
GetDisplayOptionsRequest	*Contains a* **PaneToken** *that indicates what media profile to delete.* tt:ReferenceToken **VideoOutput[**1][1]
GetDisplayOptionsResponse	*This message contains the token of the video output that the options are intended for.* tt:LayoutOptions **LayoutOptions**[0][1] tt:CodingCapabilities **CodingCapabilities**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoVideoOutput	*The requested VideoOutputToken does not exist*

14.4 Events

The display service shall dispatch events through the event service.

14.4.1 Decoding error event

The device shall be capable of generating the following event whenever it receives a bitstream that it is not able to decode.

There are several reasons why a decoder is not able to decode the bitstream. The following error codes are defined and shall be used by the device to inform the client about decoder errors:

1) “unsupported codec or unsupported codec profile” – The device is not able to decode the bitstream, because the codec or the profile is not supported by the device. The client should the try to reconfigure the transmitter according to the CodingCapabilities of the device.

2) “packet error” – there are missing or unexpected packets in the bitstream.

Other vendor specific codes are also allowed.

```
Topic: tns1:VideoDecoder/DecodingError
<tt:MessageDescription IsProperty="false">
      <tt:Source>
            <tt:SimpleItemDescription Name="VideoOutputToken"
      Type="tt:ReferenceToken"/>
      </tt:Source>
      <tt:Data>
          <tt:SimpleItemDescription Name="PaneReference"
      Type="tt:ReferenceToken"/>
          <tt:SimpleItemDescription Name=”Error” Type=”xs:string”
      minOccurs=”0”/>
      </tt:Data>
</tt:MessageDescription>
```

14.5 Service specific fault codes

Table 209 lists the display service specific fault codes. Additionally, each command can also generate a generic fault, see Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 209: Service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Sender	ter:InvalidArgVal	The Video Output	The requested Video

	ter:NoVideoOutput	does not exist	Output does not exist
env:Sender	ter.InvalidArgVal ter:FixPane	The PaneConfiguration is fixed.	It is not possible to delete this pane configuration
env:Sender	ter:InvalidArgVal ter:MaximumNumberOfPanes	The maximum number of panes is reached.	It is not possible to create a Pane because the maximum number of panes is reached
env:Sender	ter:InvalidArgVal ter:NoPane	The PaneConfiguration does not exist	The requested Pane Configuration does not exist.
envSe:nder	ter:InvalidArgVal ter:InvalidConfig	It is not possible to set the configuration	The requested configuration is not supported by the device
env:Sender	ter:Action ter:InvalidLayout	It is not possible to set the layout	The requested layout is not supported by the device

15 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. A device shall provide an event services as defined in [ONVIF Event WSDL]. Both an NVT and NVC shall support [WS-Addressing] for event services.

Event Handling in this standard is based on the [WS-BaseNotification] and [WS-Topics] specifications. This standard requires the implementation of the basic notification interface as described in section 15.1, which conforms completely to the [WS-BaseNotification] specification. In addition, the device shall implement the Real-time Pull-Point Notification Interface and the Notification Streaming Interface as introduced in sections 15.2 and 15.3, respectively.

This standard introduces notification message extensions that allow a client to track object properties (such as video analytics object properties) through events. Properties are defined in Section 15.4.

The description of event payload and their filtering within subscriptions is discussed in Section 15.5. Section 15.6 describes how Synchronization Point can be requested by clients using one of the three Notification Interfaces. Section 15.7 describes the integration of Topics. Section 15.9 discusses the handling of faults.

The last section demonstrates in detail 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.

15.1 Basic Notification Interface

Section 15.1.1 briefly introduces the Basic Notification Interface of the [WS-BaseNotification] specification. Section 15.1.2 summarizes the mandatory and the optional interfaces of the [WS-BaseNotification] specification.

15.1.1 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 21. 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.

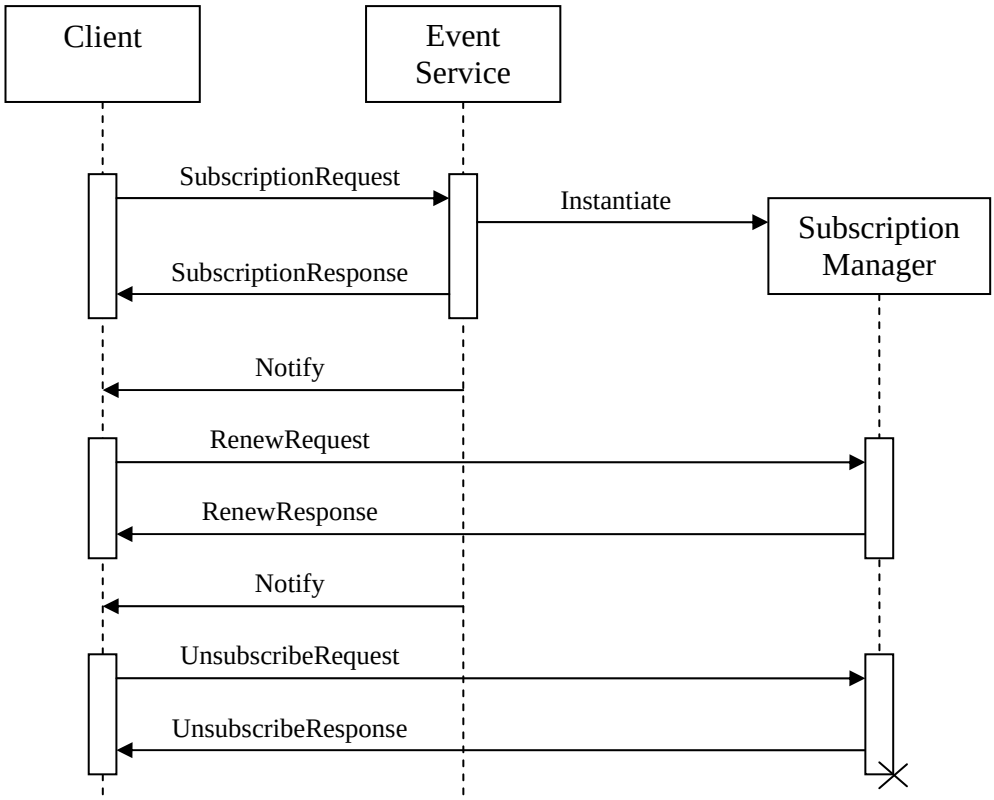


Figure 21: Sequence diagram for the Base Notification Interface

15.1.2 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]. As a result, the NotificationProducer Resource Properties are OPTIONAL (see Section 15.5). The device shall support TopicExpression and MessageContent filters with at least the dialects described in Sections 15.5.5 and 15.7.3. 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, an ONVIF compliant device shall list them in SubscribeResponses. 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.

15.2 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 PullPointSubscription with the CreatePullPointSubscriptionRequest message. The request contains a detailed description of the Subscription. The ConsumerReference shall be omitted, in contrast to the subscription of the Basic Notification Interface (see Section 15.1).

2) The device evaluates the Subscription and returns either a CreatePullPointSubscriptionResponse when the Subscription is accepted or one of the Fault codes.

3) If the Subscription is accepted, the response contains a WS-EndpointReference to a SubscriptionManager. This WS-Endpoint shall provide a PullMessages operation, which is used by the client to retrieve Notifications and by the Base Subscription Manager Interface described in the [WS-BaseNotification] specification. The Base Subscription Manager Interface consists of PullMessages, Renew and Unsubscribe operations. The sequence diagram of the interaction is shown in Figure 22. The PullMessagesRequest contains Timeout and MessageLimit parameters.

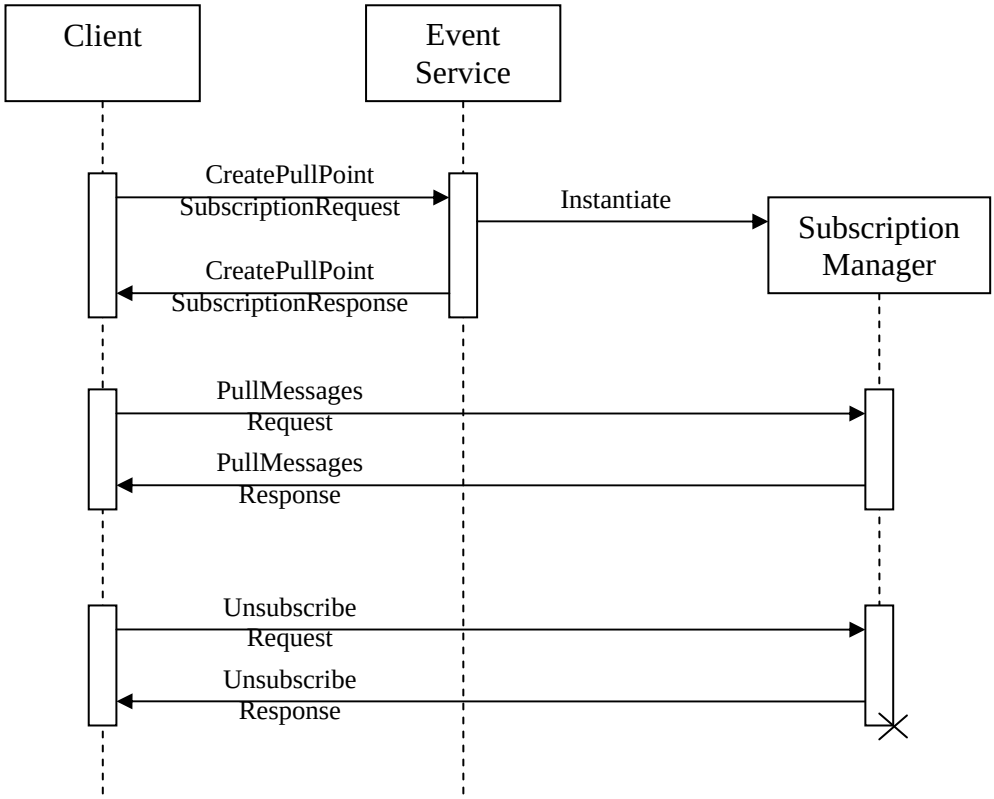


Figure 22: Sequence diagram for the Real-time Pull-Point Notification Interface.

4) The device shall immediately respond with notifications that have been aggregated on behalf of the client. If there are no aggregated notifications, the device waits with it's response until either a notification is produced for the client or the specified Timeout is 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.

5) If neither a termination time nor a relative termination time is set in the CreatePullPointSubscriptionRequest, each PullMessagesRequest shall be interpreted as a keep-alive for the corresponding PullPointSubscription. The termination time is recomputed according to the relative termination time if available or according to a device internal default value. To inform the client about the updated termination time, the PullMessagesReponse shall contain the CurrentTime and TerminationTime elements. When the PullMessagesRequest is used as keep-alive for the corresponding PullPointSubscription, the RenewRequest, defined by the [WS-BaseNotification], need not be called by a client. Nevertheless, the device shall support it for the PullPointSubscription.

15.2.1 Create pull point subscription

The device shall provide the following CreatePullPointSubscription command:

Table 210: CreatePullPointSubscription command

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

15.2.2 Pull messages

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

Table 211: PullMessages command

PullMessages	Request-response
Message name	**Description**
PullMessagesRequest	*This message shall be addressed to a SubscriptionManager in order to pull notifications:* xs:duration **Timeout** [1][1] xs:int **MessageLimit** [1][1]
PullMessagesResponse	*The response contains a list of notifications together with an updated TerminationTime for the SubscriptionManager:* xs:dateTime **CurrentTime** [1][1] xs:dateTime **TerminationTime** [1][1] wsnt:NotificationMessageHolderType **NotificationMessage** [0][unbounded]
PullMessagesFaultResponse	*Either Timeout or MessageLimit exceed the upper limit supported by the device. The Fault Message shall contain the upper limits for both parameters.* xs:duration **MaxTimeout**[1][1] xs:int **MaxMessageLimit**[1][1]

Fault codes	Description
	No specific fault codes.

15.3 Notification Streaming Interface

Section 11.10 describes the creation, deletion and modification of metadata configurations. Certain metadata configurations can contain multiple subscriptions whose structure is the same as that for a notification subscription. When a metadata configuration containing subscriptions has been assigned to a profile, a client uses that profile to get an RTP stream that includes the configured notifications as metadata. The notification streaming via RTP shall be implemented by an ONVIF compliant device.

The [WS-BaseNotification] defines the element `wsnt:NotificationMessage` to pack the Message Payload, the Topic and the ProducerReference. The structure of this message is the same as that for direct notification requests (the format is described in Section 15.5). Multiple instances of the `wsnt:NotificationMessage` elements can be placed within a metadata document introduced in the Real-time Viewing section.

There is no explicit SubscriptionReference with streaming notifications. Therefore, the `wsnt:NotificationMessage` shall NOT contain the SubscriptionReference element.

15.4 Properties

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

When a client subscribes to a topic representing a certain property, the device shall provide notifications informing the client of all objects with the requested property, which are alive at the time of the subscription. An client *can* also request the values of all currently alive properties the client has subscribed to at any time by asking for a synchronization point (see section 15.6).

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 15.5 explains the structure of events and properties in detail.

15.4.1 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.

15.5 Notification Structure

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

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

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

This corresponds to the following XML structure:

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

where the wsnt:Message element contains the actual notification payload. The XML type of the Message element can be specified within a TopicTree definition (see Section 15.7).

Section 15.5.1 gives an overview of the information a client retrieves through notifications. Section 15.5.2 gives a detailed formatting of the Message payload, and Section 15.5.4 introduces a description language for the Message payload. Section 15.5.5 defines the grammar used in a subscription to filter notifications by their Message content.

15.5.1 Notification information

A notification answers at least the following questions:

When did it happen?

Who produced the event?

What happened?

The “when” question is answered by adding a time attribute to the Message element of the NotificationMessage. 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 multiple parameters or none may be needed to identify the component uniquely. These parameters are placed as Items within the Source element of the Message container.

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

When the topic points to properties (see Section 15.4), the client uses the NotificationProducer, the Topic, the Source Items and optional Key Items (see Section 15.5) in order to identify the property. These values shall result in a unique identifier.

15.5.1.1 Event Example

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” identifies uniquely 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 the information which preset it is. Thereby, the Preset is identified by a PresetToken “Preset5” which is named “PresetName”.

15.5.2 Message Format

The Message element of the NotificationMessage is defined in [ONVIF Schema]. The definition is presented below[5]:

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

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

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

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

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

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

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

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

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

15.5.3 Property example, continued

The example in section 15.4.1 required an optional Key Item. 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.

15.5.4 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[6]:

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

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

The Name attribute of an Item shall be unique within all Items independent from the group (Source, Key, Data) they are coming from. The IsProperty attribute shall be set to true when the described Message relates to a property. If the Message, however, does not relate to a property, the Key group shall NOT be present. The Type attribute of a SimpleItemDescriptor shall match the SimpleElement definition of an XML schema. Similarly, the Type attribute of an ElementItemDescriptor shall match a global element declaration of an XML schema.

The location of all schema files used to describe Message payloads are listed in the GetEventPropertiesResponse message in Section 15.8.

15.5.4.1 Message Description Example

The following code is an example of a Message Description corresponding to the Property example of Section 15.5.3:

```
<tt:MessageDescription IsProperty="true">
```

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

```
  <tt:Source>
    <tt:SimpleItemDescriptionDescription Name="VideoSourceConfigurationToken"
                                Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescriptionDescription Name="VideoAnalyticsConfigurationToken"
                                Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescriptionDescription Name="Rule"
                                Type="xs:string"/>
  </tt:Source>
  <tt:Key>
    <tt:SimpleItemDescriptionDescription Name="ObjectId"
                                Type="tt:ObjectRefType"/>
  </tt:Key>
  <tt:Data>
    <tt:SimpleItemDescriptionDescription Name="IsInside"
                                Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

15.5.5 Message Content Filter

In the Subscription request, a client can filter notifications by TopicExpression (see Section 15.7.3) 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. An ONVIF compliant device shall implement the subset of XPath 1.0. 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"] )
```

15.6 Synchronization Point

Properties, introduced in section 15.4, inform a client about property creation, changes and deletion in a uniform way. When a client wants to synchronize its properties with the properties of the device, it can request a synchronization point which repeats the current status of all properties to which a client has subscribed. The PropertyOperation of all produced notifications is set to "Initialized" (see Section 15.5). The Synchronization Point is requested directly from the SubscriptionManager which was returned in either the SubscriptionResponse or in the CreatePullPointSubscriptionResponse. The property update is transmitted via the notification transportation of the notification interface. The following operation shall be provided by all Subscription Manager Endpoints:

Table 212: SetSynchronizationPoint command

SetSynchronizationPoint	Request-Response
Message name	**Description**
SetSynchronizationPoint-Request	*This message is empty.*

SetSynchronizationPoint-Response	*This message is empty.*
Fault codes	**Description**
	No command specific faults!

When a client uses the notification streaming interface, the client should use the SetSynchronizationPoint operation defined in the media service, see Section 11.18.

15.7 Topic Structure

This standard extends the Topic framework defined in the [WS-Topics] specification. Section 15.7.1 describes an ONVIF Topic Namespace, which should be taken as a basis for vendor specific topics. The Appendix 0 shows typical examples for such extensions. Section 15.7.2 defines an interface to topic properties. This interface shall be implemented by an ONVIF compliant device. Section 15.7.3 incorporates the Message Description Language defined in section 15.5.4 into the TopicSet structure. All topics grown from the ONVIF Topic Namespace describes the type of a topic according to section 15.7.3. Section 15.7.3 defines the Topic Expression Dialects which are supported by a device.

15.7.1 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.

The following root topics are defined in the ONVIF Namespace. All notifications referring to these topics shall use the Message Format as described in Section 15.5.2.

```
<wstop:TopicNamespace name="ONVIF"
 targetNamespace="http://www.onvif.org/ver10/topics" >
  <wstop:Topic name="Device"/>
  <wstop:Topic name="VideoSource"/>
  <wstop:Topic name="VideoEncoder"/>
  <wstop:Topic name="VideoAnalytics"/>
  <wstop:Topic name="RuleEngine"/>
  <wstop:Topic name="PTZController"/>
  <wstop:Topic name="AudioSource"/>
  <wstop:Topic name="AudioEncoder"/>
  <wstop:Topic name="UserAlarm"/>
  <wstop:Topic name="MediaControl"/>
  <wstop:Topic name="RecordingConfig"/>
  <wstop:Topic name="RecordingHistory"/>
  <wstop:Topic name="VideoOutput"/>
  <wstop:Topic name="AudioOutput"/>
  <wstop:Topic name="VideoDecoder"/>
  <wstop:Topic name="AudioDecoder"/>
  <wstop:Topic name=″Receiver″/>
</wstop:TopicNamespace>
```

15.7.2 Topic Type Information

The type information is added below a topic element by adding a MessageDescription element of type MessageDescriptionType defined in Section 15.5.4. Topic elements can be identified by the `wstop:topic attribute` with value `"true"`.

The following example demonstrates how Topics of a TopicSet are augmented with Message Descriptions:

```
<tns1:RuleEngine wstop:topic="true">
  <tns1:LineDetector wstop:topic="true">
    <tns1: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="tt:ObjectRefType"/>
        </tt:Data>
      </tt:MessageDescription>
    </tns1:Crossed>
  </tns1:LineDetector>
  <tns1:FieldDetector wstop:topic="true">
    <tns1: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="tt:ObjectRefType"/>
        </tt;Key>
        <tt:Data>
          <tt:SimpleItemDescription Name="IsInside" Type="xs:boolean"/>
        </tt:Data>
      </tt:MessageDescription>
    </tns1:ObjectsInside>
  </tns1:FieldDetector>
</tns1:RuleEngine>
```

15.7.3 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>
```

15.8 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 FilterDialects, Schema files and topics supported by the device.

Table 213: GetEventProperties command

GetEventProperties	Request-response
Message name	**Description**
GetEventPropertiesRequest	*This is an empty message.*
GetEventPropertiesResponse	xs:anyURI **TopicNamespaceLocatio**n [1][unbounded] xs:boolean **FixedTopicSet** [1][1] wstop:TopicSetType **TopicSet** [1][1] xs:anyURI **TopicExpressionDialect** [1][unbounded] xs:anyURI **MessageContentFilterDialect** [1][unbounded] xs:anyURI **ProducerPropertiesFilterDialect** [0][unbounded] xs:anyURI **MessageContentSchemaLocation** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

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 15.7.3):

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

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

The following MessageContentFilterDialects are mandatory for the an ONVIF compliant device(see Section 15.5.5):

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

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

The Message Content Description Language, introduced in Section 15.5.4, 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.

15.9 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.

15.10 Notification example

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

15.10.1 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>
```

15.10.2 GetEventPropertiesResponse

In this example, the device response uses the ONVIF topic namespace (the description can be downloaded from `http://www.onvif.org/onvif/ver10/topics/topicns.xml`). 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>
        <tns1:RuleEngine>
          <tns1:LineDetector>
            <tns1: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="tt:ObjectRefType"/>
                </tt:Data>
              </tt:MessageDescription>
            </tns1:Crossed>
          </tns1: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>
```

15.10.3 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/CreatePullPointSubscriptionReq
uest
    </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/ItemFilter">
          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>
```

15.10.4 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/CreatePullPointSubscription
Response
    </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>
```

15.10.5 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
</wsa:Action>
    <wsa:To>http://160.10.64.10/Subscription?Idx=0</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:PullMessages>
      <tet:Timeout>
      PT5S
      </tet:Timeout>
      <tet:MessageLimit>
        2
      </tet:MessageLimit>
    </tet:PullMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

15.10.6 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="2008-10-10T12:24:57.321">
              <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="2008-10-10T12:24:57.789">
              <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>
```

15.10.7 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>
```

15.10.8 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>
```

15.11 Service specific fault codes

The event service does not define any service specific faults except those defined in [WS-BaseNotification].

16 PTZ control

The PTZ control provides operations used to perform device pan, tilt and zoom control. A device with pan, tilt and zoom capability shall support the PTZ control. Similarly, a device with either just zoom or pan and tilt capability shall support the PTZ control. The PTZ control is defined in [ONVIF PTZ WSDL]. The mandatory operations are indicated under the command descriptions.

The PTZ control covers a wide category of camera devices. A dome model PTZ device is assumed to be able to change the viewing direction of the camera independently from the zoom.

PTZ movement is controlled using a coordinate system model. A device lists the set of coordinate systems it supports. This specification provides a set of generic Coordinate Spaces that is applicable for any PTZ device. It is possible to define further coordinate systems which are more appropriate for specific dome hardware. The PTZ control is applicable to the following devices:

- Dome or PTZ camera.
- Network video encoder with dome or PTZ camera connected via external serial port.
- Fixed megapixel camera with digital PTZ.
- Fixed camera with zoom.

The PTZ *control structure* consists of three major blocks:

- *PTZ Node* – A low-level PTZ entity that maps to the PTZ device and specifies its capabilities.
- *PTZ Configuration* – PTZ configuration of default coordinate systems and default speeds for a specific PTZ Node.
- *PTZ Control Operation* – Move, preset, and auxiliary operations.

The PTZ control is connected to a media profile by including the PTZ configuration into the profile *(*see Section 4.8.1) *and* all PTZ control operations are done by referring to a particular media profile.

The PTZ control does not provide operations to create or manipulate PTZ Nodes. For each available PTZ Node, the devide shall provide at least one PTZ Configuration assigned to this PTZ Node. This PTZ Configuration, then can be added to Media Profiles, which are used to control the dome. Each media profile contains no more than *one* PTZ configuration. The PTZ Configuration and the VideoSourceConfiguration belong together in the Media Profile because the VideoSourceConfiguration refers to the Camera which is controlled by the PTZ Configuration.

A PTZ-capable device shall provide at least one *ready-to-use-profile* including a PTZConfiguration that covers the most basic settings and a corresponding VideoSourceConfiguration as soon as the underlying PTZ device is ready to operate.

16.1 PTZ Model

The PTZ Model groups the possible movements of the PTZ unit into a Pan/Tilt component and into a Zoom component. To steer the PTZ unit, the service provides absolute move, relative move and continuous move operations. Different coordinate systems and units are used to feed these operations.

The PTZ service provides an AbsoluteMove operation to move the PTZ device to an absolute position. The service expects the absolute position as an argument referencing an absolute coordinate system. The speed of the Pan/Tilt movement and the Zoom movement can be specified optionally. Speed values are positive scalars and do not contain any directional information. It is not possible to specify speeds for Pan and Tilt separately without knowledge about the current position. This approach to specifying a desired position generally produces a non-smooth and non-intuitive action.

A RelativeMove operation is introduced by the PTZ service in order to steer the dome relative to the current position, but without the need to know the current position. The operation expects a positional translation as an argument referencing a relative coordinate system. This specification distinguishes between relative and absolute coordinate systems, since there are cases where no absolute coordinate system exists for a well-defined relative coordinate system. An optional speed argument can be added to the RelativeMove operation with the same meaning as for the AbsoluteMove operation.

Finally, the PTZ device can be moved continuously via the ContinuousMove command in a certain direction with a certain speed. Thereby, a velocity vector represents both, the direction and the speed information. The latter is expressed by the length of the vector.

The Pan/Tilt and Zoom coordinates can be uniquely specified by augmenting the coordinates with appropriate Space URIs. A Space URI uniquely represents the underlying coordinate system. Section 16.8 defines a standard set of coordinate systems. A PTZ Node shall implement these coordinate systems if the corresponding type of movement is supported by the PTZ Node. In many cases, the Pan/Tilt position is represented by pan and tilt angles in a spherical coordinate system. A digital PTZ, operating on a fixed megapixel camera, may express the camera's viewing direction by a pixel position on a static projection plane. Therefore, different coordinate systems are needed in this case in order to capture the physical or virtual movements of the PTZ device. These and other additional coordinate systems are defined in a separate document, *ONVIF PTZ Coordinate Spaces*. Optionally, the PTZ Node may define its own device specific coordinate systems to enable NVCs to take advantage of the specific properties of this PTZ Node.

The PTZ Node description retrieved via the GetNode or GetNodes operation contains all coordinate systems supported by a specific PTZ Node. Each coordinate system belongs to one of the following groups:

- AbsolutePanTiltPositionSpace
- RelativePanTiltTranslationSpace
- ContinuousPanTiltVelocitySpace
- PanTiltSpeedSpace
- AbsoluteZoomPositionSpace
- RelativeZoomTranslationSpace
- ContinuousZoomVelocitySpace

- ZoomSpeedSpace

If the PTZ Node does not support the coordinate systems of a certain group, the corresponding move operation will not be available for this PTZ Node. For instance, if the list does not contain an AbsolutePanTiltPositionSpace, the AbsoluteMove operation shall fail when an absolute Pan/Tilt position is specified. The corresponding command section describes those spaces that are required for a specific move command.

16.2 PTZ Node

A PTZ-capable device can have multiple PTZ Nodes. The PTZ Nodes may represent mechanical PTZ drivers, uploaded PTZ drivers or digital PTZ drivers. PTZ Nodes are the lowest level entities in the PTZ control API and reflect the supported PTZ capabilities. The PTZ Node is referenced either by its name or by its reference token. The PTZ Service does not provide operations to create or manipulate PTZ Nodes.

The following properties shall be provided for all PTZ Nodes:

- Token – A unique identifier that is used to reference PTZ Nodes.
- Name – A name given by the installer.
- SupportedPTZSpaces – A list of Coordinate Systems available for the PTZ Node. For each Coordinate System, the PTZ Node shall specify its allowed range.
- MaximumNumberOfPresets – All preset operations shall be available for this PTZ Node if one preset is supported.
- HomeSupported – A boolean operator specifying the availability of a home position. If set to true, the Home Position Operations shall be available for this PTZ Node.
- AuxiliaryCommands – A list of supported Auxiliary commands. If the list is not empty, the Auxiliary Operations shall be available for this PTZ Node.

16.2.1 GetNodes

A PTZ-capable device shall implement this operation and return all PTZ Nodes available on the device.

Table 214: GetNodes command

GetNodes	Request-Response
Message name	**Description**
GetNodesRequest	*This is an empty message.*
GetNodesResponse	*The response message contains a list of the existing PTZ Nodes on the device.* tt:PTZNode **PTZNode**[0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported by the device.*

16.2.2 GetNode

A PTZ-capable device shall implement the GetNode operation and return the properties of the requested PTZ Node, if it exists. Otherwise, the device shall respond with an appropriate Fault message.

Table 215: GetNode command

GetNode	Request-Response
Message name	**Description**
GetNodeRequest	*This message contains a reference to the requested PTZNode* tt:ReferenceToken **NodeToken**[1][1]
GetNodeResponse	*The PTZNode response message contains the requested PTZNode.* tt:PTZNode **PTZNode**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoEntity	*No such PTZNode on the device*
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.3 PTZ Configuration

The PTZConfiguration contains a reference to the PTZ Node in which it belongs. This reference cannot be changed by an NVC.

The following elements are part of the PTZ Configuration:

- PTZNodeToken – A mandatory reference to the PTZ Node that the PTZ Configuration belongs to.
- DefaultAbsolutePanTiltPositionSpace – If the PTZ Node supports absolute Pan/Tilt movements, it shall specify one Absolute Pan/Tilt Position Space as default.
- DefaultRelativePanTiltTranslationSpace – If the PTZ Node supports relative Pan/Tilt movements, it shall specify one RelativePan/Tilt Translation Space as default.
- DefaultContinuousPanTiltVelocitySpace – If the PTZ Node supports continuous Pan/Tilt movements, it shall specify one Continuous Pan/Tilt Velocity Space as default.
- DefaultPanTiltSpeedSpace – If the PTZ Node supports absolute or relative movements, it shall specify one Pan/Tilt Speed Space as default.
- DefaultAbsoluteZoomPositionSpace – If the PTZ Node supports absolute zoom movements, it shall specify one Absolute Zoom Position Space as default.
- DefaultRelativeZoomTranslationSpace – If the PTZ Node supports relative zoom movements, it shall specify one Relative Zoom Translation Space as default.

- DefaultContinuousZoomVelocitySpace – If the PTZ Node supports continuous zoom movements, it shall specify one Continuous Zoom Velocity Space as default.
- DefaultZoomSpeedSpace – If the PTZ Node supports either absolute or relative movements, it shall specify one Zoom Speed Space as default.
- DefaultPTZSpeed – If the PTZ Node supports absolute or relative PTZ movements, it shall specify corresponding default Pan/Tilt and Zoom speeds.
- DefaultPTZTimeout – If the PTZ Node supports continuous movements, it shall specify a default timeout, after which the movement stops.
- PanTiltLimits – The Pan/Tilt limits element should be present for a PTZ Node that supports an absolute Pan/Tilt. If the element is present it signals the support for configurable Pan/Tilt limits. If limits are enabled, the Pan/Tilt movements shall always stay within the specified range. The Pan/Tilt limits are disabled by setting the limits to –INF or +INF.
- ZoomLimits – The Zoom limits element should be present for a PTZ Node that supports absolute zoom. If the element is present it signals the supports for configurable Zoom limits. If limits are enabled the zoom movements shall always stay within the specified range. The Zoom limits are disabled by settings the limits to -INF and +INF.

The default Position/Translation/Velocity Spaces are introduced to allow NVCs sending move requests without the need to specify a certain coordinate system. The default Speeds are introduced to control the speed of move requests (absolute, relative, preset), where no explicit speed has been set.

The allowed pan and tilt range for Pan/Tilt Limits is defined by a two-dimensional space range that is mapped to a specific Absolute Pan/Tilt Position Space. At least one Pan/Tilt Position Space is required by the PTZNode to support Pan/Tilt limits. The limits apply to all supported absolute, relative and continuous Pan/Tilt movements. The limits shall be checked within the coordinate system for which the limits have been specified. That means that even if movements are specified in a different coordinate system, the requested movements shall be transformed to the coordinate system of the limits where the limits can be checked. When a relative or continuous movements is specified, which would leave the specified limits, the PTZ unit has to move along the specified limits. The Zoom Limits have to be interpreted accordingly.

16.3.1 GetConfigurations

A PTZ-capable device shall return all available PTZConfigurations through the GetConfigurations operation.

Table 216: GetConfigurations command

GetConfigurations	Request-Response
Message name	**Description**
GetConfigurations	*This is an empty message.*
GetConfigurationsResponse	*The response contains all existing PTZConfigurations on the device.* tt:PTZConfiguration **PTZConfiguration** [0][unbounded]

Fault codes	Description
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.3.2 GetConfiguration

A PTZ-capable device shall return the requested PTZ Configuration, if it exists, through the GetConfiguration operation.

Table 217: GetConfiguration command

GetConfiguration	Request-Response
Message name	**Description**
GetConfigurationRequest	*This message contains a reference to the requested PTZConfiguration.* tt:ReferenceToken **ConfigurationToken**[1][1]
GetConfigurationResponse	*The response contains the requested PTZConfiguration* tt:PTZConfiguration **PTZConfiguration** [1][1]
Fault codes	**Description**
env:Sender ter: InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported by the device.*

16.3.3 GetConfigurationOptions

A PTZ-capable device shall implement the GetConfigurationOptions operation. It returns the list of supported coordinate systems including their range limitations. Therefore, the options MAY differ depending on whether the PTZ Configuration is assigned to a Profile containing a Video Source Configuration. In that case, the options may additionally contain coordinate systems referring to the image coordinate system described by the Video Source Configuration. Each listed coordinate system belongs to one of the groups listed in Section 16.1. If the PTZ Node supports continuous movements, it shall return a Timeout Range within which Timeouts are accepted by the PTZ Node.

Table 218: GetConfigurationOptions command

GetConfigurationOptions	Request-Response
Message name	**Description**
GetConfigurationOptions-Request	*This message contains a token to a PTZ configuration.* ***ConfigurationToken*** *specifies an existing configuration that the options are intended for.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetConfigurationOptions-Response	*This message contains the PTZ configuration options.*

	tt:PTZConfigurationOptions **PTZConfigurationOptions**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration does not exist.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.3.4 SetConfiguration

A PTZ-capable device shall implement the SetConfiguration operation. The ForcePersistence flag indicates if the changes remain after reboot of the device.

Table 219: SetConfiguration command

SetConfiguration	Request-Response
Message name	**Description**
SetConfigurationRequest	*The **PTZConfiguration** element contains the modified PTZ configuration. The configuration shall exist in the device.* *The **ForcePersistence** element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:PTZConfiguration **PTZConfiguration**[1][1] xs:boolean **ForcePersistence**[1][1]
SetConfigurationResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The configuration does not exist.*
env:Sender ter:InvalidArgVal ter:ConfigModify	*The configuration parameters are not possible to set.*
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflict with other uses of the configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.4 Move Operations

This section describes three operations to move the PTZ unit absolutely, relatively or continuously. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

All Move commands are non-blocking, meaning they do not wait until the requested move operation has finished. The last move operation can be overwritten by sending another move request.

16.4.1 AbsoluteMove

If a PTZ Node supports absolute Pan/Tilt or absolute Zoom movements, it shall support the AbsoluteMove operation. The Position argument of this command specifies the absolute position to which the PTZ Unit moves. It splits into an optional Pan/Tilt element and an optional Zoom element. If the Pan/Tilt position is omitted, the current Pan/Tilt movement shall NOT be affected by this command. The same holds for the zoom position.

The spaces referenced within the Position shall be absolute position spaces supported by the PTZ Node. If the Space information is omitted, the corresponding default spaces of the PTZ configuration, a part of the specified Media Profile, is used. A device may support absolute Pan/Tilt movements, absolute Zoom movements or no absolute movements by providing only absolute position spaces for the supported cases.

An existing Speed argument overrides the DefaultSpeed of the corresponding PTZ configuration during movement to the requested position. If spaces are referenced within the Speed argument, they shall be Speed Spaces supported by the PTZ Node.

The operation shall fail if the requested absolute position is not reachable.

Table 220: AbsoluteMove command

AbsoluteMove	Request-Response
Message name	**Description**
AbsoluteMoveRequest	*This message contains a reference to the media profile, a **Position** vector specifying the absolute target position and an optional **Speed**.* tt:ReferenceToken **ProfileToken** [1][1] tt:PTZVector **Position** [1][1] tt:PTZSpeed **Speed** [0][1]
AbsoluteMoveResponse	*This is an empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env:Sender ter:InvalidArgVal ter:SpaceNotSupported	*A space is referenced in an argument which is not supported by the PTZ Node.*
env:Sender ter:InvalidArgVal ter:InvalidPosition	*The requested position is out of bounds.*
env:Sender ter:InvalidArgVal	*The requested speed is out of bounds.*

ter:InvalidSpeed	
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.4.2 RelativeMove

If a PTZ Node supports relative Pan/Tilt or relative Zoom movements, then it shall support the RelativeMove operation. The Translation argument of this operation specifies the difference from the current position to the position to which the PTZ device is instructed to move. The operation is split into an optional Pan/Tilt element and an optional Zoom element. If the Pan/Tilt element is omitted, the current Pan/Tilt movement shall NOT be affected by this command. The same holds for the zoom element.

The spaces referenced within the Translation element shall be Translation spaces supported by the PTZ Node. If the Space information is omitted for the Translation argument, the corresponding default spaces of the PTZ configuration, which is part of the specified Media Profile, is used. A device may support relative Pan/Tilt movements, relative Zoom movements or no relative movements by providing only translation spaces for the supported cases.

An existing Speed argument overrides the DefaultSpeed of the corresponding PTZ configuration during movement by the requested translation. If spaces are referenced within the Speed argument, they shall be Speed Spaces supported by the PTZ Node.

The command can be used to stop the PTZ Unit at its current position by sending zero values for Pan/Tilt and Zoom. Stopping shall have the very same effect independent of the relative space referenced.

If the requested translation leads to an absolute position which cannot be reached, the PTZ Node shall move to a reachable position along the border of valid positions.

Table 221: RelativeMove command

RelativeMove	Request-Response
Message name	**Description**
RelativeMoveRequest	*This message contains a reference to the media profile, a positional **Translation** relative to the current position and an optional **Speed** parameter.* tt:ReferenceToken **ProfileToken** [1][1] tt:PTZVector **Translation** [1][1] tt:PTZSpeed **Speed** [0][1]
RelativeMoveResponse	*This is an empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*

env:Sender ter:InvalidArgVal ter:SpaceNotSupported	*A space is referenced in an argument which is not supported by the PTZ Node.*
env:Sender ter:InvalidArgVal ter:InvalidTranslation	*The requested translation is out of bounds.*
env:Sender ter:InvalidArgVal ter:InvalidSpeed	*The requested speed is out of bounds.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.4.3 ContinuousMove

A PTZ-capable device shall support continuous movements. The Velocity argument of this command specifies a signed speed value for the Pan, Tilt and Zoom. The combined Pan/Tilt element is optional and the Zoom element itself is optional. If the Pan/Tilt element is omitted, the current Pan/Tilt movement shall NOT be affected by this command. The same holds for the Zoom element. The spaces referenced within the Velocity element shall be Velocity spaces supported by the PTZ Node. If the Space information is omitted for the Velocity argument, the corresponding default spaces of the PTZ configuration belonging to the specified Media Profile is used. A device MAY support continuous Pan/Tilt movements and/or continuous Zoom movements by providing only velocity spaces for the supported cases.

An existing Timeout argument overrides the DefaultPTZTimeout parameter of the corresponding PTZ configuration for this Move operation. The Timeout parameter specifies how long the PTZ Node continues to move.

The command can be used to stop the PTZ device at its current position by sending zero values for the Pan/Tilt and Zoom parameters. Stopping shall have the same effect independent of the velocity space referenced. This command has the same effect on a continuous move as the stop command specified in Section 16.4.4.

If the requested velocity leads to absolute positions which cannot be reached, the PTZ Node shall move to a reachable position along the border of its range. A typical application of the Continuous Move operation is controlling PTZ via joystick.

Table 222: ContinuousMove command

ContinuousMove	Request-Response
Message name	**Description**
ContinuousMoveRequest	*This message contains a reference to the media profile, a **Velocity** vector specifying the velocity of pan, tilt and zoom, and an optional **Timeout** parameter.* tt:ReferenceToken **ProfileToken** [1][1] tt:PTZSpeed **Velocity** [1][1] xs:duration **Timeout** [0][1]
ContinuousMoveResponse	*This is an empty message.*
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env:Sender ter:InvalidArgVal ter:SpaceNotSupported	*A space is referenced in an argument which is not supported by the PTZ Node.*
env:Sender ter:InvalidArgVal ter:TimeoutNotSupported	*The specified timeout argument is not within the supported timeout range.*
env:Sender ter:InvalidArgVal ter:InvalidVelocity	*The requested velocity is out of bounds.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.4.4 Stop

A PTZ-capable device shall support the Stop operation. If no Stop filter arguments are present, this command stops all ongoing pan, tilt and zoom movements. The Stop operation can be filtered to stop a specific movement by setting the corresponding stop argument.

Table 223: Stop (PTZ) command

Stop	Request-Response
Message name	**Description**
StopRequest	*This message contains a reference to the MediaProfile and parameters that indicate what should be stopped.* tt:ReferenceToken **ProfileToken**[1][1] xs:boolean **PanTilt** [0][1] xs:boolean **Zoom**0][1]
StopResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.4.5 GetStatus

A PTZ-capable device shall be able to report its PTZ status through the GetStatus command. The PTZ Status contains the following information:

- Position (optional) – Specifies the absolute position of the PTZ unit together with the Space references. The default absolute spaces of the corresponding PTZ configuration shall be referenced within the Position element.
- MoveStatus (optional) – Indicates if the Pan/Tilt/Zoom device unit is currently moving, idle or in an unknown state.
- Error (optional) – States a current PTZ error.
- UTC Time – Specifies the UTC time when this status was generated.

Table 224: GetStatus (PTZ) command

GetStatus	Request-Response
Message name	**Description**
GetStatusRequest	*This message contains a reference to the media profile where the PTZStatus should be requested.* tt:ReferenceToken **ProfileToken**[1][1]
GetStatusResponse	*This message contains the PTZStatus for the requested MediaProfile.* tt:PTZStatus **PTZStatus**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env:Receiver ter:Action ter:NoStatus	*No PTZ status is available in the requested Media Profile.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.5 Preset operations

This section describes operations that manage the presets of a PTZ Node. These operations shall be implemented for PTZ Nodes supporting presets. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

16.5.1 SetPreset

The SetPreset command saves the *current* device position parameters so that the device can move to the saved preset position through the GotoPreset operation.

In order to create a new preset, the SetPresetRequest contains no PresetToken. If creation is successful, the Response contains the PresetToken which uniquely identifies the Preset. An existing Preset can be overwritten by specifying the PresetToken of the corresponding Preset. In both cases (overwriting or creation) an optional PresetName can be specified. The operation fails if the PTZ device is moving during the SetPreset operation.

The device MAY internally save additional states such as imaging properties in the PTZ Preset which then should be recalled in the GotoPreset operation.

Table 225: SetPreset command

SetPreset	Request-Response
Message name	**Description**
SetPresetRequest	*This message contains a reference to the MediaProfile and the requested name or token for the preset.* tt:ReferenceToken **ProfileToken**[1][1] tt:ReferenceToken **PresetToken**[0][1] xs:string **PresetName**[0][1]
SetPresetResponse	*This message contains a reference to the Preset which has been set.* tt:ReferenceToken **PresetToken**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:PresetExist	*The requested name already exist for another preset.*
env:Sender ter:InvalidArgVal ter:InvalidPresetName	*The PresetName is either too long or contains invalid characters.*
env:Receiver ter:Action ter:MovingPTZ	*Preset cannot be set while PTZ unit is moving.*
env:Receiver ter:Action ter:TooManyPresets	*Maximum number of Presets reached.*
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token* **ProfileToken** *does not exist.*
env:Sender ter:InvalidArgVal ter:NoToken	*The requested preset token does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.5.2 GetPresets

The GetPresets operation returns the saved Presets consisting of the following elements:

- Token – A unique identifier to reference the Preset.
- Name – An optional mnemonic name.
- PTZ Position – An optional absolute position. If the PTZ Node supports absolute Pan/Tilt position spaces, the Pan/Tilt position shall be specified. If the PTZ Node supports absolute zoom position spaces, the zoom position shall be specified.

Table 226: GetPresets command

GetPresets	Request-Response
Message name	**Description**
GetPresetsRequest	*This message contains a reference to the MediaProfile where the operation should take place.* tt:ReferenceToken **ProfileToken**[1][1]
GetPresetsResponse	*This message contains a list of presets which are available for the requested MediaProfile.* tt:PTZPreset **Preset**[0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.5.3 GotoPreset

The GotoPreset operation recalls a previously set Preset. If the speed parameter is omitted, the default speed of the corresponding PTZ Configuration shall be used. The speed parameter can only be specified when Speed Spaces are available for the PTZ Node. The GotoPreset command is a non-blocking operation and can be interrupted by other move commands.

Table 227: GotoPreset command

GotoPreset	Request-Response
Message name	**Description**
GotoPresetRequest	*This message contains a reference to the MediaProfile where the move to the preset identified by its token should take place.* tt:ReferenceToken **ProfileToken**[1][1] tt:ReferenceToken **PresetToken**[1][1] tt:PTZSpeed **Speed**[0][1]

GotoPresetResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoToken	*The requested preset token does not exist.*
env:Sender ter:InvalidArgVal ter:SpaceNotSupported	*A space is referenced in an argument which is not supported by the PTZ Node.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env:Sender ter:InvalidArgs ter:InvalidSpeed	*The requested speed is out of bounds.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.5.4 RemovePreset

The RemovePreset operation removes a previously set Preset.

Table 228: RemovePreset command

RemovePreset	Request-Response
Message name	**Description**
RemovePresetRequest	*This message contains a reference to the MediaProfile where the preset identified by the token should be removed.* tt:ReferenceToken **ProfileToken**[1][1] tt:ReferenceToken **PresetToken**[1][1]
RemovePresetResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoToken	*The requested preset token does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.6 Home Position operations

This section describes operations used to manage the Home Position of a PTZ Node. These operations shall be implemented for PTZ Nodes supporting home positions. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

The “home” position MAY be set by the SetHome operation or is a fix position of the PTZ unit.

16.6.1 GotoHomePosition

This operation moves the dome to its home position. If the speed parameter is omitted, the default speed of the corresponding PTZ Configuration shall be used. The speed parameter can only be specified when Speed Spaces are available for the PTZ Node.The command is non-blocking and can be interrupted by other move commands.

Table 229: GotoHomePosition command

GotoHomePosition	Request-Response
Message name	**Description**
GotoHomePositionRequest	*This message contains a reference to the MediaProfile where the operation should take place.* tt:ReferenceToken **ProfileToken**[1][1] tt:PTZSpeed **Speed**[0][1]
GotoHomePositionResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Receiver ter:Action ter:NoHomePosition	*No home position has been defined for this Profile.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.6.2 SetHomePosition

The SetHome operation saves the *current* position parameters as the home position, so that the GotoHome operation can request that the device move to the home position.

The SetHomePosition command shall return with a failure if the “home” position is fixed and cannot be overwritten. If the SetHomePosition is successful, it shall be possible to recall the Home Position with the GotoHomePosition command.

Table 230: SetHomePosition command

SetHomePosition	Request-Response
Message name	**Description**
SetHomePositionRequest	*This message contains a reference to the MediaProfile where the home position should be set.* tt:ReferenceToken **ProfileToken**[1][1]
SetHomePositionResponse	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env:Receiver ter:Action ter:CannotOverwriteHome	*The home position is fixed and cannot be overwritten.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.7 Auxiliary operations

This section describes operations to manage auxiliary commands of a PTZ Node, such as an Infrared (IR) lamp, a heater or a wiper.

These operations shall be implemented for PTZ nodes indicating auxiliary commands in the node properties. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

16.7.1 SendAuxiliaryCommand

This operation is used to call an auxiliary operation on the device. The supported commands can be retrieved via the PTZ Node properties. The AuxiliaryCommand should match the supported command listed in the PTZ Node; no other syntax is supported. If the PTZ Node lists the *irlampon* command, then the AuxiliaryCommand argument would be *irlampon.* The SendAuxiliaryCommand shall be implemented when the PTZ Node supports auxiliary commands.

Table 231: Send Auxiliary command

SendAuxiliaryCommand	Request-Response
Message name	**Description**
SendAuxiliaryCommandRequest	*This message contains a reference to the MediaProfile where the Auxiliary request should be done and the Auxiliary request data.* tt:ReferenceToken **ProfileToken**[1][1] tt:AuxiliaryData **AuxiliaryData**[1][1]

SendAuxiliaryCommandResponse	*The response contains the auxiliary response.* tt:AuxiliaryData **AuxiliaryResponse**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The requested profile token **ProfileToken** does not exist.*
env:Sender ter:InvalidArgVal ter:NoPTZProfile	*The requested profile token does not reference a PTZ configuration.*
env: Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported.*

16.8 Predefined PTZ spaces

Spaces are used to specify absolute, relative and continuous movements. Whereas absolute movements require an absolute position, relative movements are specified by a position translation. Continuous movements require the specification of a velocity (relative movement over time). For these three cases, different coordinate systems are used describing the desired movement. The Generic Spaces do not absolutely specify the underlying PTZ Model, so that it can be applied to any PTZ hardware. Additional Spaces are defined in the document *ONVIF PTZ Coordinate Spaces*.

16.8.1 Absolute Position Spaces

16.8.1.1 Generic Pan/Tilt Position Space

The Generic Pan/Tilt Position Space shall be provided by every PTZ Node that supports absolute Pan/Tilt, since it does not relate to a specific physical range. Instead, the range should be defined as the full range of the PTZ unit normalized to the range -1 to 1 resulting in the following space description:

```
<tt:AbsolutePanTiltPositionSpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
  <tt:YRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:YRange>
</tt:AbsolutePanTiltPositionSpace>
```

16.8.1.2 Generic Zoom Position Space

The Generic Zoom Position Space shall be provided by every PTZ Node that supports absolute Zoom, since it does not relate to a specific physical range. Instead, the range should be defined as the full range of the Zoom normalized to the range 0 (wide) to 1 (tele). There is no assumption about how the generic zoom range is mapped to magnification, FOV or other physical zoom dimension. This results in the following space description:

```
<tt:AbsoluteZoomPositionSpace>
  <tt:SpaceURI>
  http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>0.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
</tt:AbsoluteZoomPositionSpace>
```

16.8.2 Relative Translation Spaces

A Relative Pan/Tilt Translation Space moves the PTZ unit a certain translation in a certain direction without knowing the camera's current Pan/Tilt position.

16.8.2.1 Generic Pan/Tilt Translation Space

The Generic Pan/Tilt Translation Space shall be provided by every PTZ Node that supports relative Pan/Tilt, since it does not relate to a specific physical range. Instead, the range should be defined as the full positive and negative translation range of the PTZ unit normalized to the range -1 to 1, where positive translation would mean clockwise rotation or movement in right/up direction resulting in the following space description:

```
<tt:RelativePanTiltTranslationSpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/TranslationGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
  <tt:YRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:YRange>
</tt:RelativePanTiltTranslationSpace>
```

16.8.2.2 Generic Zoom Translation Space

The Generic Zoom Translation Space shall be provided by every PTZ Node that supports relative Zoom, since it does not relate to a specific physical range. Instead, the corresponding absolute range should be defined as the full positive and negative translation range of the Zoom normalized to the range -1 to1, where a positive translation maps to a movement in TELE direction. The translation is signed to indicate direction (negative is to wide, positive is to tele). There is no assumption about how the generic zoom range is mapped to magnification, FOV or other physical zoom dimension. This results in the following space description:

```
<tt:RelativeZoomTranslationSpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/ZoomSpaces/TranslationGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
</tt:RelativeZoomTranslationSpace>
```

16.8.3 Continuous Velocity Spaces

The Continuous Velocity Spaces are used to continuously move the PTZ unit in a certain direction.

16.8.3.1 Generic Pan/Tilt Velocity Space

The Generic Pan/Tilt Velocity Space shall be provided by every PTZ Node, since it does not relate to a specific physical range. Instead, the range should be defined as a range of the PTZ unit's speed normalized to the range -1 to 1, where a positive velocity would map to clockwise rotation or movement in the right/up direction. A signed speed can be independently specified for the pan and tilt component resulting in the following space description:

```
<tt:ContinuousPanTiltVelocitySpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
  <tt:YRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:YRange>
</tt:ContinuousPanTiltVelocitySpace>
```

16.8.3.2 Generic Zoom Velocity Space

The Generic Zoom Velocity Space specifies a zoom factor velocity without knowing the underlying physical model. The range should be normalized from -1 to 1, where a positive velocity would map to TELE direction. A Generic Zoom Velocity Space description resembles the following:

```
<tt:ContinuousZoomVelocitySpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>-1.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
</tt:ContinuousZoomVelocitySpace>
```

16.8.4 Speed Spaces

The Speed Spaces specify the speed for a Pan/Tilt and Zoom movement when moving to an absolute position or to a relative translation. In contrast to the Velocity Spaces, Speed Spaces do not contain any directional information. The Speed of a combined Pan/Tilt movement is represented by a single non-negative scalar value.

16.8.4.1 Generic Pan/Tilt Speed Space

The Generic Pan/Tilt Speed Space shall be provided by every PTZ Node that supports configurable speed for Pan/Tilt, since it does not relate to a specific physical range. Instead, the range should be defined as the full range of the Speed range normalized to the range 0 (stopped) to 1 (full speed). This results in the following space description:

```
<tt:PanTiltSpeedSpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/GenericSpeedSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>0.0</tt:Min>
    <tt:Max>1.0</tt:Max>
  </tt:XRange>
</tt:PanTiltSpeedSpace>
```

16.8.4.2 Generic Zoom Speed Space

The Generic Zoom Speed Space shall be provided by every PTZ Node that supports configurable speed for Zoom, since it does not relate to a specific physical range. Instead, the range should be defined as the full range of the Speed range normalized to the range 0 (stopped) to 1 (full speed). This results in the following space description:

```
<tt:ZoomSpeedSpace>
  <tt:SpaceURI>
http://www.onvif.org/ver10/tptz/ZoomSpaces/ZoomGenericSpeedSpace
  </tt:SpaceURI>
  <tt:XRange>
    <tt:Min>0.0</tt:Min>
  </tt:XRange>
</tt:ZoomSpeedSpace>
```

16.9 Service specific fault codes

Table 232 below lists the PTZ service specific fault codes. Each command can generate a generic fault, see Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic sub code is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 232: PTZspecific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:Action ter:MovingPTZ	Preset cannot be set	Preset cannot be set while the PTZ unit is moving.
env:Receiver	ter:Action ter:TooManyPresets	Number of presets limit reached	Maximum number of Presets reached.
env:Receiver	ter:ActionNotSupported ter:PTZNotSupported	PTZ not supported	PTZ is not supported by the device.
env:Sender	ter:InvalidArgVal ter:PresetExist	Token already exist	The requested name or token already exist for another preset.
env:Receiver	ter:Action ter:NoStatus	No PTZ status available	No PTZ status is available in the requested Media Profile.
env:Receiver	ter:Action ter:ConfigurationConflict	Conflict when using new settings	The new settings result in an inconsistent configuration.

env:Receiver	ter:Action ter:CannotOverwriteHome	Home position cannot be overwritten	The home position is fixed and cannot be overwritten.
env:Sender	ter:InvalidArgVal ter:NoEntity	No such PTZ node	No such PTZ Node on the device
env:Sender	ter:InvalidArgVal ter:NoConfig	No such configuration	No such configuration exist.
env:Sender	ter:InvalidArgVal ter:ConfigModify	The paramters could not be set	The configuration parameters are not possible to set.
env:Sender	ter:InvalidArgVal ter:InvalidPosition	Destination out of bounds	The requested destination is out of bounds.
env:Sender	ter:InvalidArgVal ter:InvalidTranslation	Translation out of bounds	The requested translation is out of bounds.
env:Sender	ter:InvalidArgVal ter:InvalidSpeed	Reqested speed out of bounds	The requested speed is out of bounds.
env:Sender	ter:InvalidArgVal ter:InvalidVelocity	Velocity out of bounds	The requested velocity is out of bounds.
env:Sender	ter:InvalidArgVal ter:InvalidPresetName	PresetName too long	The PresetName is either too long or contains invalid characters.
env:Sender	ter:InvalidArgVal ter:NoPTZProfile	Profile miss PTZ configuration	The requested profile token does not reference a PTZ configuration.
env:Sender	ter:InvalidArgVal ter:NoProfile	Profile token does not exist	The requested profile token **ProfileToken** does not exist.
env:Sender	ter:InvalidArgVal ter:TimeoutNotSupported	Timeout not supported	The specified timeout argument is not within the supported timeout range.
env:Sender	ter:InvalidArgVal ter:NoToken	Token does not exist.	The requested preset token does not exist

env:Receiver	ter:Action ter:NoHomePosition	No HomePosition	No home position has been defined for this Profile.
env:Sender	ter:InvalidArgVal ter:SpaceNotSupported	No such space	A space is referenced in an argument which is not supported by the PTZ Node.

17 Video analytics

Section 4.12 gives a general overview of the ONVIF video analytics architecture. This section covers the following main areas of this architecture:

- Analytics Module interface
- Scene description
- Rules interface
- Event interface

The event interface is handled through the Event service described in 15. 17.1 introduces the XML-based scene description, which can be streamed as metadata to clients via RTP (see 12.1.2.1.1 for more details). The media service provides operations to manage complete analytics configurations consisting of both the rule engine and the analytics engine configuration (see Section 11). The analytics service allows more fine-grained configuration of individual rules and individual analytics modules (see 17.2 and 17.3).

An device supporting analytics shall implement the Scene Description and Event Interface, as well as the Configuration of Analytics by the Media Service. If the device additionally supports a rule engine, responsible for analytics engine as defined by this standard, then it shall implement the Rules Analytics Modules Interface.

A complete video analytics configuration can be attached to a profile via the media service. A video analytics configuration becomes connected to a specific video source (see 11.9). The device shall ensure that a corresponding analytics engine starts operation when a client subscribes directly or indirectly for events produced by the analytics or rule engine or when a client requests the corresponding scene description stream.

17.1 Scene Description Interface

17.1.1 Overview

This specification defines the XML schema that shall be used to encode Scene Descriptions by a device. The scope of the Scene Description covers basic Scene Elements which can be displayed in a video overlay to the end-user as well as a framework for vendor-specific extensions. Appendix A.2 shows additional Scene Elements that may be used for processing vendor-specific rules.

The Video Analytics Engine is configured via Profiles of the MediaControl section. If Video Analytics are available in a Profile, a VideoSourceConfiguration and a VideoAnalyticsConfiguration shall be referenced in the Profile. The Video Analytics Engine then processes frames according to the referenced VideoSourceConfiguration.

17.1.2 Frame Related Content

The input of the Video Analytics Engine is images from a video source. The extracted scene elements are associated with the image from which they were extracted. An extracted scene is distinguished from the general description of the video source processed by the Video Analytics Engine (information such as video input line, video resolution, frame cropping, frame rate etc.), the temporal frame association within the input stream, and the spatial positioning of elements within a frame.

The linkage between a Video Source and a Video Analytics Component is part of the Media Control, allowing the Video Analytics to run on a cropped video source with a reduced frame rate.

The temporal and spatial relation of scene elements with respect to the selected video source is discussed in sections 17.1.2.1 and 17.1.2.2. The appearance and behaviour of tracked objects is discussed in section 17.1.3.1. Interactions between objects like splits and merges are described in section 17.1.3.2.

A PTZ device can put information about the Pan, Tilt and Zoom at the beginning of a frame, allowing a client to estimate the 3D coordinates of scene elements. Next, the image Coordinate System can be adapted with an optional Transformation Node which is described in the next subsection. Finally, multiple Object Descriptions can be placed and their association can be specified within an ObjectTree Node.Below, the definitions are included for convenience[7]:

```
<xs:complexType name="Frame">
  <xs:sequence>
    <xs:element name="PTZStatus" type="tt:PTZStatus"
         minOccurs="0"/>
    <xs:element name="Transformation" type="tt:Transformation"
         minOccurs="0"/>
    <xs:element name="Object" type="tt:Object" minOccurs="0"
         maxOccurs="unbounded"/>
    <xs:element name="ObjectTree" type="tt:ObjectTree" minOccurs="0"/>
    ...
  </xs:sequence>
  <xs:attribute name="UtcTime" type="xs:dateTime" use="required"/>
   ...
</xs:complexType>

<xs:element name="Frame" type="tt:Frame">
```

Subsection 17.1.2.1 describes how frames processed by the Video Analytics Algorithm are referenced within the Video Analytics stream.

17.1.2.1 Temporal Relation

Since multiple Scene Elements can be extracted from the same image, Scene Elements are listed below a Frame Node which establishes the link to a specific image from the video input. The Frame Node contains a MANDATORY UtcTime attribute. This UtcTime timestamp shall enable a client to map the Frame Node exactly to one video frame. For example,. the RTP timestamp of the corresponding encoded video frame shall result in the same UTC timestamp after conversion. The synchronization between Video and Metadata streams is further described in the Real-time Viewing Section 12.1.2.2.1.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  ...
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  ...
</tt:Frame>
```

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

17.1.2.2 Spatial Relation

Most Scene Elements refer to some part in an image from which information has been extracted. For instance, when tracking objects over time, their position within each frame shall be specified. These positions shall relate to a Coordinate System. The Default Coordinate System is shown in Figure 23. It maps onto the rectangle selected in the VideoSourceConfiguration of the corresponding Profile.

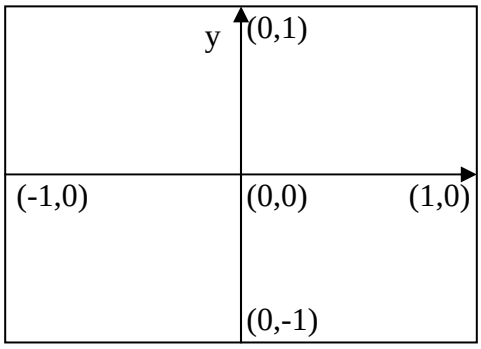


Figure 23: Default frame coordinate system

This specification allows modification of the Coordinate System for individual nodes of the XML tree. As a result, each Frame Node starts with the Default Coordinate System. Each Child Node inherits the most recent Coordinate System of its parent. A Transformation Node modifies the most recent Coordinate System of its parent. Coordinate specifications are always related to the most recent Coordinate System of the Parent Node.

The specification defines transformation nodes for scaling and translation. The Scene Description contains placeholders where these Transformation Nodes are placed[8].

```
<xs:complexType name="Transformation">
  <xs:sequence>
    <xs:element name="Translate" type="Vector" minOccurs="0"/>
    <xs:element name="Scale" type="Vector" minOccurs="0"/>
    ...
  </xs:sequence>
</xs:complexType>
```

It follows a mathematical description of coordinate systems and transformations. A coordinate system consists of a translational vector $t = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ and scaling $s = \begin{pmatrix} s_x \\ s_y \end{pmatrix}$. A point $p = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$ given with respect to this coordinate system is transformed into the corresponding point $q = \begin{pmatrix} q_x \\ q_y \end{pmatrix}$ of the default coordinate system by the following formula: $\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} p_x \cdot s_x + t_x \\ p_y \cdot s_y + t_y \end{pmatrix}$. Similarly, a vector ***v*** given with respect to the coordinate system is transformed into the corresponding vector ***w*** of the default coordinate system by: $\begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x \\ v_y \cdot s_y \end{pmatrix}$.

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

A Transformation Node has an optional scaling vector $u = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$ and an optional translational vector $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$. If the scaling is not specified, its default value $u = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ is assumed. Similarly, the default value for the translation is $v = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. The Transformation Node modifies the top-most Coordinate System in the following way:

$\begin{pmatrix} t'_x \\ t'_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x + t_x \\ v_y \cdot s_y + t_y \end{pmatrix}, \begin{pmatrix} s'_x \\ s'_y \end{pmatrix} = \begin{pmatrix} u_x \cdot s_x \\ u_y \cdot s_y \end{pmatrix}$, where $\begin{pmatrix} t'_x \\ t'_y \end{pmatrix}$ and $\begin{pmatrix} s'_x \\ s'_y \end{pmatrix}$ replace the top-most Coordinate System.

For example, the coordinates of the scene description are given in a frame coordinate system, where the lower-left corner has coordinates (0,0) and the upper-right corner coordinates (320,240). The Frame Node resembles the following code where the scaling is set to the doubled reciprocal of the frame width and the frame height:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.00625" y="0.00834"/>
  </tt:Transformation>
  ...
</tt:Frame>
```

17.1.3 Scene Elements

This section focuses on Scene Elements generated by object tracking algorithms and defines object handling and object shapes for them.

Frames where no objects have been detected can be skipped within the Scene Description to save bandwidth, as long as the last frame in the Scene Description is empty as well. It is RECOMMENDED that the device regularly sends the Scene Description even if it is empty, in order to indicate that the analytics engine is operational. The device shall send a Scene Description if a SynchronizationPoint is requested for the corresponding stream.

When the receiver of a Scene Description receives an empty frame, the receiver should assume that all subsequent frames are empty as well until the next non-empty frame is received. When the last received frame is non-empty, the receiver should assume that a description of the next processed frame will be transmitted.

17.1.3.1 Objects

Objects are identified via their Object ID. Features relating to one particular object are collected in an Object Node with the corresponding Object ID as an attribute. Associations of objects, like Object Renaming, Object Splits, Object Merges and Object Deletions are expressed in a separate ObjectTree node. An Object ID is implicitly created with the first appearance of the Object ID within an Object Node[9].

```
<xs:complexType name="ObjectId">
  <xs:attribute name="ObjectId" type="xs:int"/>
</xs:complexType>
```

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

```
<xs:complexType name="Object">
  <xs:complexContent>
    <xs:extension base="ObjectId">
      <xs:sequence>
        <xs:element name="Appearance" type="Appearance" minOccurs="0"/>
        <xs:element name="Behaviour" type="Behaviour" minOccurs="0"/>
        ...
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The Object Node has two placeholders for Appearance and Behaviour information. The Appearance Node starts with an optional Transformation Node which can be used to change from a frame-centric coordinate system to an object-centric coordinate system. Next, the Shape of an Object can be specified. If an object is detected in a frame, the Shape information should be present in the Appearance description. The video analytics algorithm MAY add Object Nodes for currently not visible Objects, if it is able to infer information for this object otherwise. In such cases, the Shape description MAY be omitted.

Other object features like colour and object class can be added to the Appearance Node. This specification focuses on the Shape Descriptors (see section 17.1.3.3). The definition of colour and object class can be found in Appendix B.1.

This specification defines two standard Behaviours for Objects. When an Object stops moving, it can be marked as either Removed or Idle. These behaviours shall be listed as Child Nodes of the Behaviour Node of an Object. The presence of a Removed or Idle Node does not automatically delete the corresponding Object ID, making it possible to reuse the same Object ID when the object starts moving again.

An object marked with the Removed Behaviour specifies the place from where the real object was removed. The marker should not be used as the Behaviour of the removed object. It is possible to detect the removal although the action of taking away the object was not detected.

Objects previously in motion can be marked as Idle to indicate that the object stopped moving. As long as such objects don't change, they will not be listed in the Scene Description anymore. When an Idle object appears again in the Scene Description, the Idle flag is removed automatically.

Example:

```
...
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.421">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
```

```
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
    <tt:Behaviour>
      <tt:Idle/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.621">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="25.0" top="30.0" right="105.0" bottom="80.0"/>
        <tt:CenterOfGravity x="65.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
...
<tt:Frame UtcTime="2008-10-10T12:24:57.721">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1.0"/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="19">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
      </tt:Shape>
    </tt:Appearance>
    <tt:Behaviour>
      <tt:Removed/>
    </tt:Behaviour>
  </tt:Object>
</tt:Frame>
```

17.1.3.2 Object Tree

When two objects come too close to each other, such that the Video Analytics can no longer track them individually, an Object Merge should be signalled by adding a Merge Node to the ObjectTree Node of the Frame Node. The Merge Node contains a From Node listing the merging ObjectIds and a To Node containing the ObjectId. The merged Object is used in future frames as the tracking ID. If the Video Analytics Algorithm detects that one object is occluding the others and is able to track this object further, the occluding object should be put in the To Node.

The separation of objects is indicated by a Split Node. In this case, the From Node contains a single ObjectId representing the object which is split in the current frame. The objects separating from this split object are listed in the To Node. The ObjectId of the From Node can reappear in the To Node, if this object did occlude the others and the Video Analytics Algorithm was able to track this object during the occlusion.

An Object does not need to be involved in a merge operation in order to be part of a split operation. For example, if an object is moving together with a person, and the person leaves the object somewhere, the object might be detected the first time by the Video Analytics when the person moves away from the object left behind. In such cases, the first appearance of the object can be combined with a Split operation.

When a merged object reappears as an Object Node in a later frame without a split indication, then this object is implicitly split. The Video Analytics Algorithm, however, could not determine where the split object came from.

A Video Analytics Algorithm can track and remember a limited number of objects. In order to indicate that a certain Object has been removed from the memory of the algorithm and therefore never appear again, the SceneDescription can contain a Delete Node within the ObjectTree Node.

If the Video Analytics Algorithm can not decide during a Split operation the identity of an object, it should use a new ObjectId. When the algorithm has collected sufficient evidence for the identity of this object, it can change the ObjectId via the Rename operation. The Rename operation can also be used when an object reenters the scene and the true identity is discovered after some time.

A deleted ObjectId shall NOT be reused within the Scene Description until the ObjectId container has wrapped around.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:Object ObjectId="17">
    ...
  </tt:Object>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.421">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:ObjectTree>
    <tt:Merge>
      <tt:From ObjectId="12"/>
      <tt:From ObjectId="17"/>
      <tt:To ObjectId="12"/>
    </tt:Merge>
  </tt:ObjectTree>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.521">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
</tt:Frame>

<tt:Frame UtcTime="2008-10-10T12:24:57.621">
  <tt:Object ObjectId="12">
    ...
  </tt:Object>
  <tt:Object ObjectId="17">
    ...
  </tt:Object>
  <tt:ObjectTree>
    <tt:Split>
      <tt:From ObjectId="12"/>
      <tt:To ObjectId="17"/>
```

```
      <tt:To ObjectId="12"/>
    </tt:Split>
  </tt:ObjectTree>
</tt:Frame>
```

17.1.3.3 Shape descriptor

Shape information shall be placed below the optional Shape Node of in an Object Appearance Node. If present, the Shape Node holds information where the Object under consideration has been detected in the specified frame. A Shape Node shall at least contain two Nodes representing the Bounding Box and the Center Of Gravity of the detected object.

The coarse Bounding Box is further refined with additional Child Nodes, each representing a Shape Primitive. If multiple Shape Primitives are present, their union defines the Object's Shape. In this specification, a generic Polygon Descriptor is provided.

Polygons that describe the shape of an object shall be simple polygons defined by a list of Points.

Two consecutive Points (where the last point is connected with the first one) in the list define a line segment. The order of the Points shall be chosen such that the enclosed Object region can be found on the left-hand side all line segments. The polyline defined by the list of Points shall NOT be self-intersecting.

Example:

```
<tt:Frame UtcTime="2008-10-10T12:24:57.321">
  <tt:Transformation>
    <tt:Translate x="-1.0" y="-1".0/>
    <tt:Scale x="0.003125" y="0.00416667"/>
  </tt:Transformation>
  <tt:Object ObjectId="12">
    <tt:Appearance>
      <tt:Shape>
        <tt:BoundingBox left="20.0" top="30.0" right="100.0" bottom="80.0"/>
        <tt:CenterOfGravity x="60.0" y="50.0"/>
        <tt:Polygon>
          <tt:Point x="20.0" y="30.0"/>
          <tt:Point x="100.0" y="30.0"/>
          <tt:Point x="100.0" y="80.0"/>
          <tt:Point x="20.0" y="80.0"/>
        </tt:Polygon>
      </tt:Shape>
    </tt:Appearance>
  </tt:Object>
</tt:Frame>
```

17.2 Rule interface

The Video Analytics Configuration consists of two parts (see Section 11.9). The first part configures the Video Analytics Engine creating the SceneDescription. The second part configures the Rule Engine. For the second part, a XML structure is introduced in Section 17.2.1 to communicate the configuration of Rules. Section 17.2.2 specifies a language to describe the configuration of a specific Rule type. Section 17.2.3 defines two standard Rules that should be supported by a device implementing a Rule Engine. Section 17.2.4 introduces operations to manage rules. If the device supports a Rule Engine, it shall implement the complete Rule Interface.

17.2.1 Rule representation

The configuration of a rule has two required attributes: one specifies the Name and the other specifies the Type of the Rule. The different configuration parameters are listed below the Parameters element of the Rule element. Each Parameter is either a SimpleItem or an

ElementItem (compare with message payload in Section 15). The Name attribute of each Item shall be unique within the parameter list. SimpleItems have an additional Value attribute containing the value of the parameter. The value of ElementItems is given by the child element of the ElementItem. It is RECOMMENDED to represent as many parameters as possible by SimpleItems.

The following example shows a complete Video Analytics Configuration containing two Rules:

```
<tt:VideoAnalyticsConfiguration>
  <tt:AnalyticsEngineConfiguration>
    ...
  </tt:AnalyticsEngineConfiguration>
  <tt:RuleEngineConfiguration>
    <tt:Rule Name="MyLineDetector" Type="tt:LineDetector">
      <tt:Parameters>
        <tt:SimpleItem Name="Direction" Value="Any"/>
        <tt:ElementItem Name="Segments">
          <tt:Polyline>
            <tt:Point x="10.0"  y="50.0"/>
            <tt:Point x="100.0" y="50.0"/>
          </tt:Polyline>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:Rule>
    <tt:Rule Name="MyFieldDetector" Type="tt:FieldDetector">
      <tt:Parameters>
        <tt:ElementItem Name="Field">
          <tt:Polygon>
            <tt:Point x="10.0"  y="50.0"/>
            <tt:Point x="100.0" y="50.0"/>
            <tt:Point x="100.0" y="150.0"/>
          </tt:Polygon>
        </tt:ElementItem>
      </tt:Parameters>
    </tt:Rule>
  </tt:RuleEngineConfiguration>
</tt:VideoAnalyticsConfiguration>
```

17.2.2 Rule description language

The description of a Rule contains the type information of all parameters belonging to a certain Rule Type and the description of the output produced by such a rule. The output of the Rule Engine is Events which can either be used in an Event Engine or be subscribed to by a client.

The parameters of a certain Rule Type are listed below the ParameterDescription element. All parameters are either Simple or ElementItems and can be described by either a SimpleItemDescription or an ElementItemDescription. Both ItemDescriptions contain a Name attribute to identify the parameter and a Type attribute to reference a specific XML schema type. In case of the SimpleItemDescription, the Type attribute shall reference a SimpleType schema definition. In case of the ElementItemDescription, the Type attribute shall reference a global element declaration of an XML schema.

The output produced by this Rule Type is described in multiple MessageDescription elements. Each MessageDescription contains a description of the Message Payload according to the Message Description Language detailed in Section 15. Additionally, the MessageDescription shall contain a ParentTopic element naming the Topic a client has to subscribe to in order to receive this specific output. The Topic shall be specified as a Concrete Topic Expression.

Section 17.2.3 demonstrates the usage of the Rule Description Language on two standard rules. Below, the definitions are included for convenience[10]:

```
<xs:element name="RuleDescription" type="tt:ConfigDescription"/>

<xs:complexType name="ConfigDescription">
  <xs:sequence>
    <xs:element name="ParameterDescription"
                type="tt:ItemListDescription"/>
    <xs:element name="Messages" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:complexContent>
          <xs:extension base="tt:MessageDescription">
            <xs:sequence>
              <xs:element name="ParentTopic" type="xs:string"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      </xs:element>
    ...
  </xs:sequence>
  <xs:attribute name="Name" type="xs:string" use="required"/>
</xs:complexType>

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

17.2.3 Standard Rules

The following standard rules apply to static cameras. In case of a PTZ device, image-based rules should contain an additional ElementItem. The ElementItem identifies the position of the device for which the rule has been setup. The corresponding ElementItemDescription resembles the following:

```
    <tt:ElementItemDescription Name="PTZStatus" Type="tt:PTZStatusType">
```

17.2.3.1 LineDetector

The LineDetector is defined by a non-intersecting simple polyline. If an Object crosses the polyline in the specified direction, the Rule Engine sends a Crossed event containing the name of the LineDetector and a reference to the object which has crossed the line. As directions, one can select between Left, Right, and Any, where directions Left and Right refer to the direction walking along the line from the first point to the second point and are the prohibited directions.

---

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

The LineDetector resembles the following code using the Rule Description Language, detailed in the previous section:

```
<tt:RuleDescription Name="tt:LineDetector">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="Direction" Type="tt:Direction"/>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
  </tt:Parameters>
  <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="tt:ObjectId"/>
    </tt:Data>
    <tt:ParentTopic>tns1:RuleEngine/LineDetector/Crossed</tt:ParentTopic>
  </tt:MessageDescription>
</tt:RuleDescription>
```

The code above defines two parameters, Segments and Direction, and produces one Event attached to the topic tns1:RuleEngine/LineDetector/Crossed.

17.2.3.2 FieldDetector

A FieldDetector is defined by a simple non-intersecting polygon. The FieldDetector determines if each object in the scene inside or outside the polygon. This information is put into a property.

The FieldDetector resembles the following code, using the Rule Description Language detailed in the previous section:

```
<tt:RuleDescription Name="tt:FieldDetector">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Field" Type="tt:Polygon"/>
  </tt:Parameters>
  <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="tt:ObjectIdType"/>
    </tt:Key>
    <tt:Data>
      <tt:SimpleItemDescription Name="IsInside" Type="xs:boolean"/>
    </tt:Data >
    <tt:ParentTopic>
      tns1:RuleEngine/FieldDetector/ObjectsInside
    </tt:ParentTopic>
  </tt:MessageDescription>
</tt:RuleDescription>
```

From the Inside property, a client can derive the Entering and the Leaving parameters of the detector. A client can simulate Entering and Leaving events by adding a MessageContent Filter to the subscription, which lets only ObjectsInside messages pass, where the IsInside Item is set to true resp. false.

17.2.4 Operations on rules

If the device supports a Rule Engine as defined by ONVIF, then it shall implement the following operations to manage rules. The Create/Delete/Modify operations are atomic, meaning that either all modifications can be processed or the complete operation shall fail.

17.2.4.1 Get Supported rules

The device shall indicate the rules it supports by implementing the subsequent operation. It returns a list of Rule Descriptions according to the Rule Description Language described in Section 17.2.2. Additionally, it contains a list of URLs that provide the location of the schema files. These schema files describe the types and elements used in the Rule Descriptions. If rule descriptions reference types or elements of the ONVIF schema file, the ONVIF schema file shall be explicitly listed.

Table 233: GetSupportedRules command

GetSupportedRules	Request-response
Message name	**Description**
GetSupportedRulesRequest	*The request message contains the VideoAnalyticsConfigurationToken for which the supported rules should be listed.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetSupportedRulesResponse	*The response contains the supported rules.* tt: SupportedRules **SupportedRules** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

17.2.4.2 Get Rules

The following operation retrieves the currently installed Rules:

Table 234: GetRules command

GetRules	Request-response
Message name	**Description**
GetRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the rules should be reported.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetRulesResponse	*The response is a list of installed rules for the specified configuration.* tt:Config **Rule** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*

17.2.4.3 Create rules

The following operation adds Rules to a VideoAnalyticsConfiguration. If all rules can not be created as requested, the device responds with a fault message.

Table 235: CreateRules command

CreateRules	Request-response
Message name	**Description**
CreateRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken to which the listed Rules should be added*. tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **Rule** [1][unbounded]
CreateRulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidRule	*The suggested rules configuration is not valid on the device.*
env:Sender ter:InvalidArgVal ter:RuleAlreadyExistent	*The same rule name exists already in the configuration.*
enc:Receiver ter:Action ter:TooManyRules	*There is not enough space in the device to add the rules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot create the rules without creating a conflicting configuration.*

17.2.4.4 Modify Rules

The following operation modifies Multiple Rules. If all rules can not be modified as requested, the device responds with a fault message.

Table 236: ModifyRules command

ModifyRules	Request-response
Message name	**Description**
ModifyRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the listed Rules should be modified.* tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **Rule**[1][unbounded]
ModifyRulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*

env:Sender ter:InvalidArgVal ter:InvalidRule	*The suggested rules configuration is not valid on the device.*
env:Sender ter:InvalidArgs ter:RuleNotExistent	*The rule name or names do not exist.*
enc:Receiver ter:Action ter:TooManyRules	*There is not enough space in the device to add the rules to the configuration.*
env:Receiver ter:Action ter:ConflictingConfig	*The device cannot modify the rules without creating a conflicting configuration.*

17.2.4.5 Delete Rules

The following operation deletes Multiple Rules. If all rules can not be deleted as requested, the device responds with a fault message.

Table 237: DeleteRules command

DeleteRules	Request-response
Message name	**Description**
DeleteRulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken from which the listed Rules should be removed.* tt:ReferenceToken **ConfigurationToken** [1][1] xs:string **RuleName** [1][unbounded]
DeleteRulesResponse	*The response is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Receiver ter:Action ter:ConflictingConfig	*The device cannot delete the rules without creating a conflicting configuration.*
env:Sender ter:InvalidArgs ter:RuleNotExistent	*The rule name or names do not exist.*

17.3 Analytics Modules Interface

The Video Analytics Configuration consists of two parts (see Section 11.9). The first part configures the Video Analytics Engine creating the SceneDescription. The second part configures the Rule Engine. Section 17.3.1 defines an XML structure for the first part that communicates the configuration of Analytics Modules. Section 17.3.2 defines the language that describes the configuration of a specific Analytics Module. Section 17.3.3 defines the operations required by the Analytics Modules Interface. If the device supports an Analytics Engine as defined by ONVIF, it shall implement the complete Analytics Modules Interface.

17.3.1 Analytics module configuration

The Analytics Module Configuration is identical to the Rule Configuration, described in Section 17.2.1. The following example shows a possible configuration of a vendor-specific

ObjectTracker. This tracker allows configuration of the minimum and maximum object size with respect to the processed frame geometry.

```
<tt:VideoAnalyticsConfig>
  <tt:AnalyticsEngineConfig>
    <tt:AnalyticsModule Name="MyObjectTracker" Type="nn:ObjectTracker">
      <tt:Parameters>
        <tt:SimpleItem Name="MinObjectWidth" Value="0.01"/>
        <tt:SimpleItem Name="MinObjectHeight" Value="0.01"/>
        <tt:SimpleItem Name="MaxObjectWidth" Value="0.5"/>
        <tt:SimpleItem Name="MaxObjectHeight" Value="0.5"/>
      </tt:Parameters>
    </tt:AnalyticsModule>
  </tt:AnalyticsEngineConfig>
  <tt:RuleEngineConfig>
    ...
  </tt:RuleEngineConfig>
</tt:VideoAnalyticsConfig>
```

17.3.2 Analytics Module Description Language

The Analytics Module reuses the Rule Description Language, described in Section 17.2.2. The following AnalyticsModuleDescription element replaces the RuleDescription element:

```
<xs:element name="AnalyticsModuleDescription"
            type="tt:ConfigDescription"/>
```

Similar to rules, Analytics Modules produce Events and shall be listed within the Analytics Module Description. The subsequent description corresponds to the example of the previous section. The example module produces a SceneTooCrowded Event when the scene becomes too complex for the module.

```
<tt:AnalyticsModuleDescription Name="nn:ObjectTracker">
  <tt:Parameters>
    <tt:SimpleItemDescription Name="MinObjectWidth"  Type="xs:float"/>
    <tt:SimpleItemDescription Name="MinObjectHeight" Type="xs:float"/>
    <tt:SimpleItemDescription Name="MaxObjectWidth"  Type="xs:float"/>
    <tt:SimpleItemDescription Name="MaxObjectHeight" Type="xs:float"/>
  </tt:Parameters>
  <tt:MessageDescription>
    <tt:Source>
      <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                                Type="tt:ReferenceToken"/>
      <tt:SimpleItemDescription Name="AnalyticsModule" Type="xs:string"/>
    </tt:Source>
    <tt:ParentTopic>
      tns1:VideoAnalytics/nn:ObjectTracker/SceneTooCrowded
    </tt:ParentTopic>
  </tt:MessageDescription>
</tt:AnalyticsModuleDescription>
```

17.3.3 Operations on Analytics Modules

If the device supports an analytics engine as defined by ONVIF, it shall support the subsequent operations to manage analytics modules. The Create/Delete/Modify operations shall be atomic, all modifications can be processed or the complete operation shall fail.

17.3.3.1 GetSupportedAnalyticsModules

The device indicates the analytics modules it supports by implementing the GetSupportedAnalysticsModule operation. It returns a list of Analytics Modules according to the Analytics Module Description Language, described in Section 17.2.2. Additionally, it contains a list of URLs that provide the location of the schema files. These schema files describe the types and elements used in the Analytics Module Descriptions. If the analytics

module descriptions reference types or elements of the ONVIF schema file, the ONVIFschema file shall be explicitly listed.

Table 238: GetSupportedAnalyticsModules command

GetSupportedAnalyticsModules	Request-response
Message name	**Description**
GetSupportedAnalyticsModulesRequest	*The request message contains the VideoAnalyticsConfigurationToken for which the supported analytics modules should be listed.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetSupportedAnalyticsModulesResponse	*The response contains the supported analytics modules.* **SupportedAnalyticsModules** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

17.3.3.2 GetAnalytics Modules

The following operation retrieves the currently installed Analytics Modules:

Table 239: GetAnalyticsModules command

GetAnalyticsModules	Request-response
Message name	**Description**
GetAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the analytics modules should be reported.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsModulesResponse	*The response is a list of installed analytics modules for the specified configuration.* tt:Config **AnalyticsModule** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*

17.3.3.3 CreateAnalytics Modules

The following operation adds Analytics Modules to a VideoAnalyticsConfiguration. If all Analytics Modules can not be created as requested, the device responds with a fault message.

Table 240: CreateAnalyticsModules command.

CreateAnalyticsModules	Request-response

Message name	Description
CreateAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken to which the listed Analytics Modules should be added.* tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **AnalyticsModule** [1][unbounded]
CreateAnalyticsModulesResponse	This is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgs ter:NameAlreadyExistent	*The same analytics module name exists already in the configuration.*
enc:Receiver ter:Action ter:TooManyModules	*There is not enough space in the device to add the analytics modules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot create the analytics modules without creating a conflicting configuration.*
env:Sender ter:InvalidArgVal ter:InvalidModule	*The suggested module configuration is not valid on the device.*

17.3.3.4 ModifyAnalytics Modules

The following operation modifies multiple Analytics Modules. If all analytics modules can not be modified as requested, the device respond with a fault message.

Table 241: ModifyAnalyticsModules command

ModifyAnalyticsModules	Request-response
Message name	**Description**
ModifyAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken for which the listed analytics modules should be modified.* tt:ReferenceToken **ConfigurationToken** [1][1] tt:Config **AnalyticsModule** [1][unbounded]
ModifyAnalyticsModulesResponse	The response is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Sender ter:InvalidArgs ter:NameNotExistent	*The analytics module with the requested name does not exist.*
enc:Receiver ter:Action ter:TooManyModules	*There is not enough space in the device to add the analytics modules to the configuration.*

env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot modify the analytics modules without creating a conflicting configuration.*
env:Sender ter:InvalidArgVal ter:InvalidModule	*The suggested module configuration is not valid on the device.*

17.3.3.5 DeleteAnalytics Modules

The following operation deletes multiple Analytics Modules. If all analytics modules can not be deleted as requested, the device responds with a fault message.

Table 242: DeleteAnalyticsModules command

DeleteAnalyticsModules	Request-response
Message name	**Description**
DeleteAnalyticsModulesRequest	*The request message specifies the VideoAnalyticsConfigurationToken from which the listed Analytics Modules should be removed.* tt:Reference**Token ConfigurationToken** [1][1] xs:string **AnalyticsModuleName** [1][unbounded]
DeleteAnalyticsModulesResponse	The response is an empty message.
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*The VideoAnalyticsConfiguration does not exist.*
env:Receiver ter:Action ter:ConfigurationConflict	*The device cannot delete the analytics modules without creating a conflicting configuration.*
env:Sender ter:InvalidArgs ter:NameNotExistent	*The analytics module with the requested name does not exist.*

17.4 Service-specific fault codes

Table 243 below lists the analytics service-specific fault codes. Each command can also generate a generic fault. Refer to Table 6.

The specific faults are defined as subcode of a generic fault, see Section 5.11.2.1. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 243: The analytics-specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:Action ter:TooManyRules	No more space available.	There is not enough space in the device to add the rules to the configuration.
env:Receiver	ter:Action	No more space	There is not enough space in the device to add the analytics

	ter:TooManyModules	available.	modules to the configuration.
env:Receiver	ter:Action ter:ConfigurationConflict	Conflict when using new settings	The new settings result in an inconsistent configuration.
env:Sender	ter:InvalidArgVal ter:NoConfig	No such configuration	The requested VideoAnalyticsConfiguration does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidRule	The rule is invalid.	The suggested rule configuration is not valid.
env:Sender	ter:InvalidArgVal ter:InvalidModule	The module is invalid	The suggested analytics module configuration is not valid on the device.
env:Sender	ter:InvalidArgVal ter:RuleAlreadyExistent	The rule exists	The same rule name exists already in the configuration.
env:Sender	ter:InvalidArgs ter:RuleNotExistent	The rule does not exist	The rule name or names do not exist.
env:Sender	ter:InvalidArgs ter:NameAlreadyExistent	The name exists	The same analytics module name exists already in the configuration.
env:Sender	ter:InvalidArgs ter:NameNotExistent	The name does not exist	The analytics module with the requested name does not exist.

18 Analytics device

The analytics device service has to be used for stand alone analytics devices which perform evaluation processes on media streams or metadata enhanced media streams. It may be used for other entities as well. Evaluations may involve more than one media stream or metadata enhanced media stream at a time.

The analytics device service receives media streams or metadata enhanced media streams from live-generating or storing devices. It could comprise decoder capabilities if analysis is being performed on uncompressed data. A metadata enhanced media stream describes any stream containing media data and assigned metadata.

The Analytics Device Service is being used by clients to configure properties and functionality of a stand alone analytics device or other analytic operations on an entity providing this service.

Backchannel capabilities are not provided by stand alone analytics devices.

The Analytics Device Service relies on the receiver service for receiving the data from other devices through receiver objects identified by ReceiverTokens. Mechanisms have to be provided to assign different tracks in the received RTSP stream to the appropriate AnalyticsEngine.

Changes of e.g. camera parameters while analysis is being performed may influence results of the analysis. Therefore, input parameter changes have to be reflected in the AnalyticsEngineInput structure.

18.1 Overview

The central element in the configuration of an Analytics Device Service is the AnalyticsEngineControl. It comprises necessary tokens and descriptions for the service as well as the possibility of activation/deactivation for the particular AnalyticsEngineControl.

An AnalyticsEngine could be either a single algorithm or a complete application, e.g. lost baggage. Several parameter sets (VideoAnalyticsConfiguration) can exist in parallel for an AnalyticsEngine to allow for switching between e.g. day and night configurations. Additionally, a structure is provided (AnalyticsEngineInputInfo) to describe input configuration requirements for the particular AnalyticsEngine.

In order to enable adaptation of the AnalyticsEngine to different input data the description of the input being feed into the AnalyticsEngine has to be provided in the AnalyticsEngineInput element.

All structures have to exist at least once after boot of the NVA entity and could be filled in with default values where appropriate.

18.2 Analytics Engine Input

The AnalyticsEngineInput structure describes the video and metadata input provided to a particular AnalyticsEngine. If more than one input source is being used there has to be an AnalyticsEngineInput element for each of the sources.

SourceIdentification: identifies the source the input is coming from (e.g. identification of the camera cluster, the particular camera and the profile being used)

VideoSource: information about the video source, in particular about the compression parameters being used

MetadataInput: describes the source metadata provisioning to be used for analysis

18.2.1 GetAnalyticsEngineInputs

This operation lists all available analytics engine inputs for the device. The Analytics Device Service shall support the listing of available analytics engine inputs through the GetAnalyticsEngineInputs command.

Table 244: GetAnalyticsEngineInputs command

GetAnalyticsEngineInputs	Request-Response
Message name	**Description**
GetAnalyticsEngineInputsRequest	*This is an empty message.*
GetAnalyticsEngineInputsResponse	*Contains a list of structures describing available AnalyticsEngineInputs.* tt:AnalyticsEngineInput **Configuration** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

18.2.2 GetAnalyticsEngineInput

The GetAnalyticsEngineInput command fetches the input configuration if the analytics engine input configuration token is known. An Analytics Device Service shall support the listing of an analytics engine input configuration through the GetAnalyticsEngineInput command.

Table 245: GetAnalyticsEngineInput command

GetAnalyticsEngineInput	Request-Response
Message name	**Description**
GetAnalyticsEngineInputRequest	*Contains the token of an existing analytics engine input configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsEngineInputResponse	*Contains the requested analytics engine input configuration.* tt:AnalyticsEngineInput **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.2.3 SetAnalyticsEngineInput

This command changes the analytics engine input configuration. An Analytics Device Service shall support the modification of its analytics engine input configuration through this command.

Table 246: SetAnalyticsEngineInput command

SetAnalyticsEngineInput	Request-Response
Message name	**Description**
SetAnalyticsEngineInput - Request	*The Configuration shall be the new configuration. The ForcePersistence element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AnalyticsEngineInput **Configuration**[1][1] xs:boolean **ForcePersistence** [1][1]
SetAnalyticsEngineInputResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:invalidConfig	*The configuration is not possible to set*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.2.4 CreateAnalyticsEngineInputs

This command generates analytics engine input configurations. An Analytics Device Service shall support the generation of analytics engine input configurations through this command.

Table 247: CreateAnalyticsEngineInputs command

CreateAnalyticsEngineInputs	Request-Response
Message name	**Description**
CreateAnalyticsEngineInputsRequest	*The Configuration shall be the new configuration. The ForcePersistence element determines if the configuration shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AnalyticsEngineInput **Configuration**[1][unbounded] xs:boolean **ForcePersistence** [1][unbounded]
CreateAnalyticsEngineInputsResponse	*Contains the configurations including generated tokens.* tt:AnalyticsEngineInput **Configuration**[1][unbounded]
Fault codes	**Description**

env:Sender ter:InvalidArgVal ter:invalidConfig	*The configurations are not possible to set*
env:Receiver ter:Action ter:MaxAnalyticsEngineInput	*The maximum number of supported AnalyticsEngineInput objects has been reached.*

18.2.5 DeleteAnalyticsEngineInputs

This command deletes analytics engine input configurations. An Analytics Device Service shall support the deletion of analytics engine input configurations through this command.

Table 248: DeleteAnalyticsEngineInputs command

DeleteAnalyticsEngineInputs	Request-Response
Message name	**Description**
DeleteAnalyticsEngineInputsRequest	*Contains ConfigurationTokens identifying the AnalyticsEngineInputs to be deleted.* tt:ReferenceToken **ConfigurationToken** [1][unbounded]
DeleteAnalyticsEngineInputsResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAnalyticsEngineInput	*The requested AnalyticsEngineInput indicated with **ConfigurationToken** does not exist.*
env:Sender ter:Action ter:CannotDeleteEngineInput	*It is not possible to delete a specified AnalyticsEngineInput.*

18.3 Video Analytics Configuration

18.3.1 GetVideoAnalyticsConfiguration

The GetVideoAnalyticsConfiguration command fetches the video analytics configuration if the video analytics configuration token is known. An Analytics Device Service shall support the listing of video analytics configuration through the GetVideoAnalyticsConfiguration command. All suitable video analytics configuration token can be found within available AnalyticsEngine configurations.

Table 249: GetVideoAnalyticsConfiguration command

GetVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
GetVideoAnalyticsConfigurationRequest	*Contains the token of an existing video analytics configuration.* tt:ReferenceToken **ConfigurationToken** [1][1]

GetVideoAnalyticsConfigurationResponse	*Contains the requested video analytics configuration.* tt:VideoAnalyticsConfiguration **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.3.2 SetVideoAnalyticsConfiguration

This command changes the video analytics configuration. An Analytics Device Service shall support the modification of its analytics engine configuration through this command. If the SetVideoAnalyticsConfiguration command is being received by the Analytics Device Service the changes shall be applied also to the affected configuration if it is in active use.

Table 250: SetVideoAnalyticsConfiguration command

SetVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
SetVideoAnalyticsConfiguration – Request	*The Configuration shall be the new configuration.* *The ForcePersistence element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:VideoAnalyticsConfiguration **Configuration**[1][1] xs:boolean **ForcePersistence** [1][1]
SetVideoAnalyticsConfigurationResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:invalidConfig	*The configuration is not possible to set*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.4 Analytics Engines

The structure returned by the commands defined herein contains a list of available VideoAnalyticsConfiguration for the particular AnalyticsEngine together with appropriate AnalyticsEngineInputInfo elements for each VideoAnalyticsConfiguration.

VideoAnalyticsConfiguration: description of configuration possibilities of the analytics engine

AnalyticsEngineInputInfo: information about input requirements of the analytics engine

18.4.1 GetAnalyticsEngines

This operation lists all available analytics engines for the device. The Analytics Device Service shall support the listing of available analytics engines through the GetAnalyticsEngines command.

Table 251: GetAnalyticsEngines command

GetAnalyticsEngines	Request-Response
Message name	**Description**
GetAnalyticsEnginesRequest	*This is an empty message.*
GetAnalyticsEnginesResponse	*Contains a list of structures describing available AnalyticsEngines.* tt:AnalyticsEngine **Configuration** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

18.4.2 GetAnalyticsEngine

The GetAnalyticsEngine command fetches the analytics engine if the analytics engine token is known. An Analytics Device Service shall support the listing of an analytics engine configuration through the GetAnalyticsEngine command.

Table 252: GetAnalyticsEngine command

GetAnalyticsEngine	Request-Response
Message name	**Description**
GetAnalyticsEngineRequest	*Contains the token of an existing analytics engine.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsEngineResponse	*Contains the requested AnalyticsEngine configuration.* tt:AnalyticsEngine **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.5 Analytics Engine Control

The AnalyticsEngineControl structure shall be used to exercise control through the commands defined in the following.

Name: friendly description

EngineToken: Token of the analytics engine (AnalyticsEngine) being controlled

EngineConfigToken: Token of the analytics engine configuration (VideoAnalyticsConfiguration) in effect

InputToken: Tokens of the input (AnalyticsEngineInput) configuration applied

ReceiverToken: Tokens of the receiver providing media input data. The order of ReceiverToken shall exactly match the order of InputToken.

Multicast: parameter for multicast used to configure and control multicast of the metadata stream

Subscription: Description of Topics the controlled engine is reacting on

Mode: indicating the actual status for the controlled analysis (shall be either "Idle" or "Active")

18.5.1 GetAnalyticsEngineControls

This operation lists all available analytics engine controls for the device. The Analytics Device Service shall support the listing of available analytics engine controls through the GetAnalyticsEngineControls command.

Table 253: GetAnalyticsEngineControls command

GetAnalyticsEngineControls	Request-Response
Message name	**Description**
GetAnalyticsEngineControlsRequest	*This is an empty message.*
GetAnalyticsEngineControlsResponse	*Contains a list of structures describing available AnalyticsEngineControls.* tt:AnalyticsEngineControl **AnalyticsEngineControls** [1][unbounded]
Fault codes	**Description**
	No command specific faults!

18.5.2 GetAnalyticsEngineControl

The GetAnalyticsEngineControl command fetches the analytics engine control if the analytics engine control token is known. An Analytics Device Service shall support the listing of analytics engine control configuration through the GetAnalyticsEngineControl command.

Table 254: GetAnalyticsEngineControl command

GetAnalyticsEngineControl	Request-Response
Message name	**Description**
GetAnalyticsEngineControlRequest	*Contains the token of an existing AnalyticsEngineControl.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsEngineControlResponse	*Contains the requested AnalyticsEngineControl configuration.* tt:AnalyticsEngineControl **Configuration** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.5.3 SetAnalyticsEngineControl

This command changes the AnalyticsEngineControl configuration. An Analytics Device Service shall support the modification of its analytics engine control configuration through this command.

Table 255: SetAnalyticsEngineControl command

SetAnalyticsEngineControl	Request-Response
Message name	**Description**
SetAnalyticsEngineControlRequest	*The Configuration shall be the new configuration.* *The ForcePersistence element determines if the configuration changes shall be stored and remain after reboot. If true, changes shall be persistent. If false, changes MAY revert to previous values after reboot.* tt:AnalyticsEngineControl **Configuration**[1][1] xs:boolean **ForcePersistence** [1][1]
SetAnalyticsEngineControlResponse	*This message is empty*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:invalidConfig	*The configuration is not possible to set*
env:Sender ter:InvalidArgVal ter:NoConfig	*The requested configuration indicated with **ConfigurationToken** does not exist.*

18.5.4 CreateAnalyticsEngineControl

CreateAnalyticsEngineControl shall create a new control object. Mode shall be set to "idle". To change the mode to "active" the SetAnalyticsEngineControl command can be used. An Analytics Device Service shall support the creation of control objects through this command.

Table 256: CreateAnalyticsEngineControl command

CreateAnalyticsEngineControl	Request-Response
Message name	**Description**
CreateAnalyticsEngineControlRequest	*The Configuration shall be the new configuration.* . tt:AnalyticsEngineControl **Configuration**[1][1]
CreateAnalyticsEngineControlResponse	*Contains the configuration including the generated token.* tt:AnalyticsEngineControl **Configuration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:AnalyticsEngineControlExists	*An AnalyticsEngineControl with the token **ConfigurationToken** already exists.*
env:Receiver ter:Action ter:MaxAnalyticsEngineControl	*The maximum number of supported AnalyticsEngineControl objects has been reached.*
env:Sender ter:InvalidArgVal ter:invalidConfig	*The configuration is not possible to set*

18.5.5 DeleteAnalyticsEngineControl

DeleteAnalyticsEngineControl shall delete a control object. An Analytics Device Service shall support the deletion of control objects through this command.

Table 257: DeleteAnalyticsEngineControl command

DeleteAnalyticsEngineControl	Request-Response
Message name	**Description**
DeleteAnalyticsEngineControlRequest	*Contains the **ConfigurationToken** of the AnalyticsEngineControl to be deleted.* tt:ReferenceToken **ConfigurationToken** [1][1]
DeleteAnalyticsEngineControlResponse	*This message is empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAnalyticsEngineControl	*The requested AnalyticsEngineControl indicated with **ConfigurationToken** does not exist.*
env:Sender ter:Action ter:CannotDeleteControl	*It is not possible to delete the specified AnalyticsEngineControl.*

18.6 GetAnalyticsState

GetAnalyticsState returns status information of the referenced AnalyticsEngineControl object. The structure AnalyticsStateInformation is expandable. The expansion shall be used to convey additional state information about substructures. E.g. an AnalyticsEngine is composed of different analytics algorithms for which state information should be provided. The state element of AnalyticsStateInformation always holds an aggregated state of all substructures.

An Analytics Device Service shall support state information provisioning through this command.

ConfigurationToken shall be the identification of the AnalyticsEngineControl for which the state information is requested

State shall hold the aggregated state over all substructures of AnalyticsStateInformation If state equals “Error”, the Error may be filled in with an implementation defined value. A device shall apply the following rules to compute aggregate state:

Idle	The state of all substructures is “Idle”
PartiallyActive	At least one of the substructures has state “Active”, all other substructures have state "Idle".
Active	The state of all substructures is “Active”
Error	At least one of the substructures has state “Error”

Error, if present, shall hold an implementation defined string value that describes the error.

Table 258: GetAnalyticsState

GetAnalyticsState	Request-Response
Message name	**Description**
GetAnalyticsStateRequest	*Contains the **ConfigurationToken** of the AnalyticsEngineControl for which to get the state.* tt:ReferenceToken **ConfigurationToken** [1][1]
GetAnalyticsStateResponse	*The **State** shall hold the state of the AnalyticsEngineControl.* tt:AnalyicsStateInformation **State**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAnalyticsEngineControl	*The ConfigurationToken does not reference an existing **AnalyticsEngineControl**.*

18.7 Output streaming configuration

An Analytics Device Service provides a real-time streaming interface as specified in in the chapter “Real time streaming”. by acting as an RTSP server. Instead of the token identifying the profile being used in a Media Profile, the token identifying the AnalyticsEngineControl will be used on an Analytics Device Service.

18.7.1 Request stream URI

This operation requests a URI that can be used to initiate a live stream using RTSP as the control protocol. The URI is valid only as it is specified in the response or until the Analytics Engine Control is reconfigured. The Analytics Device Service shall support the retrieval of a stream URI for a specific analytics engine control through the GetAnalyticsDeviceStreamUri command.

Table 259: GetAnalyticsDeviceStreamUri command

GetAnalyticsDeviceStreamUri	Request-Response
Message name	**Description**
GetAnalyticsDeviceStreamUriRequest	*The **StreamSetup** element contains two parts. StreamType defines if a unicast or multicast media stream is requested. Transport specifies a chain of transport protocols defining the tunneling of the media stream over different network protocols.* *The **AnalyticsEngineControlToken** element shall indicate the analytics engine control to use.* tt:StreamSetup **StreamSetup** [1][1] tt:ReferenceToken **AnalyticsEngineControlToken** [1][1]
GetAnalyticsDeviceStreamUriResponse	*Contains the **Uri** to be used for requesting the media stream* xs:anyURI **Uri** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoAnalyticsEngineControl	*The requested configuration indicated with **AnalyticsEngineControlToken** does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidStreamSetup	*Specification of StreamType or Transport part in **StreamSetup** is not supported.*
env:Sender ter:OperationProhibited ter:StreamConflict	*Specification of StreamType or Transport part in **StreamSetup** causes conflict with other streams.*

19 Recording control

19.1 Introduction

The recording service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes creation and deletion of recordings and tracks, as well as locking and unlocking ranges of recordings and deletion of recorded data.

Recording jobs transfer data from a recording source to a recording. A recording source can be a receiver object created with the receiver service, or it can be a media profile that encodes data on a local device. The media profile could be used as a source on a camera with embedded storage.

The term *recording* is used in this specification to denote a container for a set of audio, video and metadata tracks. A recording could hold any number of tracks. A track is viewed as an infinite timeline that holds data at certain times.

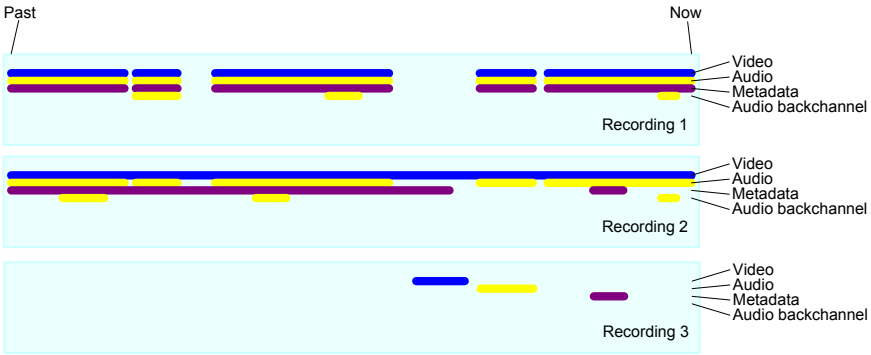


Figure 24: Example of recordings and tracks

The figure shows three recordings, each with a video, a metadata and two audio tracks. Here second audio track is used for storing the audio backchannel.

At a minimum, a recording shall be capable of holding three tracks, one for audio, one for video and one for metadata. Some implementations of the recording service may support multiple tracks of each type.

To save data to a recording, a client first creates a recording and ensures that the recording has the necessary tracks. Then the client creates a recording job that pulls data from one or more sources and stores the data to the tracks in the recording.

Clients may set up multiple recording jobs that all record into the same recording. If multiple recording jobs are active, the device uses a priority scheme to select between the tracks defined in the recording jobs. Clients may change the mode of recording jobs at any time, thereby providing means to implement features like alarm recording or manual recording.

The recording job relies on the receiver service for receiving the data from other devices through receiver objects identified by ReceiverTokens

For the cases where a client uses a receiver object with a single recording job, the recording service can auto create and auto delete receiver objects. Autocreation is signalled with the AutoCreateReceiver flag in the recording job configuration structure. Receiver objects created this way shall be automatically deleted when no recording job uses them anymore. A receiver object that is automatically created shall have all its fields set to empty values. The client should configure the receiver object after it has created the recording job.

The ONVIF view of recordings is a logical one which is independent of the way recordings are physically stored on disk. For instance, some camera implementations realise alarm recording by creating a distinct file on a FAT file system for each alarm that occurs. Although each file could be represented as a different ONVIF recording, the intent of the model in this standard is that all these files are aggregated into a single recording. By searching for the "DataPresent" event with the FindEvents method of the search service, a client can locate the times at which video started to be recorded and where video stopped being recorded.

If Metadata is recorded, the metadata can also hold all the events generated by the data source (see section 15 on event handling and section 11.10 for the MetadataConfiguration object). In addition, a device also conceptually record ONVIF defined historical events (see Recording Event Descriptions in the search service), this includes information like start and end of a recorded data range. A device may also conceptually record vendor specific historical events. Events generated by the device are not inserted in existing metadata tracks of recordings. The FindEvents method in the search service can find all the recorded events. Many device implementations will automatically delete the oldest recorded data from storage in order to free up space for new recordings. Locks provide a mechanism to allow a user to select ranges of data. A range of data that is locked does not get deleted automatically. Support for locks is reserved for future versions of the specification.

19.2 General Requirements

All the objects created within the recording service shall be persistent – i.e. they shall survive a power cycle. Likewise, all the configuration data in the objects shall be persistent.

19.3 Data structures

19.3.1 RecordingConfiguration

The RecordingConfiguration structure shall be used to configure recordings through CreateRecordings and Get/SetRecordingConfiguration.

MaximumRetentionTime specifies the maximum time that data in the any track within the recording shall be stored. The device shall delete any data older than the maximum retention time. Such data shall not be accessible anymore. If the MaximumRetentionPeriod is set to 0, the device shall not limit the retention time of stored data, except by resource constraints. Whatever the value of MaximumRetentionTime, the device may automatically delete recordings to free up storage space for new recordings.

None of the other fields defined in this structure shall be used by the device. Instead, it simply stores this information, and it shall return it through the *GetRecordingConfiguration* and *GetRecordingInformation* (see 20.5) methods.

19.3.2 TrackConfiguration

The TrackConfiguration structure shall be used to configure tracks using CreateTrack and Get/SetTrackConfiguration

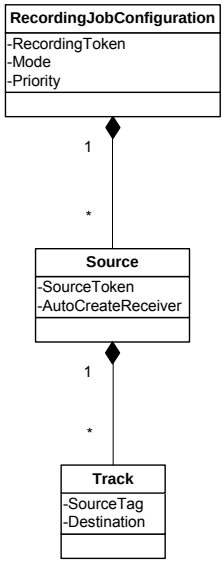
The TrackConfiguration contains the following fields:

The **TrackType** defines the data type of the track. It shall be equal to the strings "Video", "Audio" or "Metadata". The track shall only be able to hold data of that type.

None of the other fields defined in this structure shall be used by the device. Instead, it simply stores this information, and it shall return it through the *GetTrackConfiguration* and *GetRecordingInformation* (see 20.5) methods.

19.3.3 RecordingJobConfiguration

The RecordingJobConfiguration structure shall hold the configuration for a recording job. As a UML diagram, the RecordingJobConfiguration can be viewed as:



The RecordingJobConfiguration holds the following fields:

RecordingToken: Identifies the recording to which this job shall store the received data.

Mode: The mode of the job. If it is idle, nothing shall happen. If it is active, the device shall try to obtain data from the receivers. A client shall use GetRecordingJobState to determine if data transfer is really taking place. The only valid values for Mode shall be “Idle” and “Active”.

Priority: This shall be a positive number. If there are multiple recording jobs that store data to the same track, the device will only store the data for the recording job with the highest priority. The priority is specified per recording job, but the device shall determine the priority of each track individually. If there are two recording jobs with the same priority, the device shall record the data corresponding to the recording job that was activated the latest.

The value 0 indicates the lowest priority. Higher values shall indicate a higher priority.

SourceToken: This field shall be a reference to the source of the data. The type of the source is determined by the attribute Type in the SourceToken structure. If Type is http://www.onvif.org/ver10/schema/Receiver, the token is a ReceiverReference. In this case the device shall receive the data over the network. If Type is http://www.onvif.org/ver10/schema/Profile, the token identifies a media profile, instructing the device to obtain data from a profile that exists on the local device.

If the **SourceToken** is omitted, **AutoCreateRecevier** shall be true.

AutoCreateReceiver: If this field is TRUE, and if the **SourceToken** is omitted, the device shall create a receiver object (through the receiver service) and assign the ReceiverReference to the **SourceToken** field. When retrieving the RecordingJobConfiguration from the device, the **AutoCreateReceiver** field shall never be present.

SourceTag: If the received RTSP stream contains multiple tracks of the same type, the **SourceTag** differentiates between those Tracks.

Destination: The destination is the tracktoken of the track to which the device shall store the received data.

19.4 CreateRecording

CreateRecording shall create a new recording. The new recording shall be created with one video, one audio and one metadata track.

This method is optional. It shall be available if the Recording/DynamicRecordings capability is TRUE.

Table 260: CreateRecording command

CreateRecording	
Message name	**Description**
CreateRecordingRequest	*Contains the initial configuration for the recording* tt:RecordingConfiguration **RecordingConfiguration**[1][1]
CreateRecordingResponse	*Returns the reference to the created recroding* tt:RecordingReference **RecordingToken**[1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:MaxRecordings	*The device cannot create a new recording because it already has the maximum number of recordings that it supports.*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The RecordConfiguration is invalid.*
env:Receiver ter:ActionNotSupported ter:NotImplemented	*This optinal method is not implemented*

When successfully completed, CreateRecording shall have created three tracks with the following configurations:

TrackToken	TrackType
VIDEO001	Video
AUDIO001	Audio
META001	Metadata

All TrackConfigurations shall have the MaximumRetentionTime set to 0 (unlimited), and the Description set to the empty string.

19.5 DeleteRecording

DeleteRecording shall delete a recording object. Whenever a recording is deleted, the device shall delete all the tracks that are part of the recording, and it shall delete all the Recording Jobs that record into the recording. For each deleted recording job, the device shall also delete all the receiver objects associated with the recording job that are automatically created using the AutoCreateReceiver field of the recording job configuration structure and are not used in any other recording job.

This method is optional. It shall be available if the Recording/DynamicRecordings capability is TRUE.

Table 261: DeleteRecording command

DeleteRecording	
Message name	**Description**
DeleteRecordingRequest	*Identifies the recording that shall be deleted* tt:RecordingReference **RecordingToken**[1][1]
DeleteRecordingResponse	*This message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an exiting recording*
env:Receiver ter: ActionNotSupported ter:NotImplemented	*The device cannot delete recordings*
env:Receiver ter:Action ter:CannotDelete	*This specific recording cannot be deleted*

19.6 GetRecordings

GetRecordings shall return a description of all the recordings in the device. This description shall include a list of all the tracks for each recording.

Table 262: GetRecordings command

GetRecordings	
Message name	**Description**
GetRecordingsRequest	*This shall be an empty message*
GetRecordingsResponse	*The **RecordingItem** identifies a recording and its current configuration* tt:GetRecordingsResponseItem **RecordingItem**[0][unbounded]
Fault codes	**Description**
No command specific faults	

19.7 SetRecordingConfiguration

SetRecordingConfiguration shall change the configuration of a recording

Table 263: SetRecordingConfiguration command

SetRecordingConfiguration	
Message name	**Description**
SetRecordingConfigurationRequest	*The **RecordingToken** shall identify the recording that shall be changed. The* **RecordingConfiguration** *shall be the new configuration for that recording* tt:RecordingReference **RecordingToken**[1][1] tt:RecordingConfiguration **RecordingConfiguration**[1][1]
SetRecordingConfigurationResponse	This message shall be empty.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter: BadConfiguration	*The configuration is invalid.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

19.8 GetRecordingConfiguration

GetRecordingConfiguration shall retrieve the recording configuration for a recording

Table 264: GetRecordingConfiguration command

GetRecordingConfiguration	
Message name	**Description**
GetRecordingConfigurationRequest	*The **RecordingToken** shall identify the recording for which the configuration shall be retrieved.* tt:RecordingReference **RecordingToken**[1][1]
GetRecordingConfigurationResponse	*The* **RecordingConfiguration** *shall be the current configuration for the specified recording* tt:RecordingConfiguration **RecordingConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

19.9 CreateTrack

This method shall create a new track within a recording.

This method is optional. It shall be available if the Recording/DynamicTracks capability is TRUE.

Table 265: CreateTrack command

CreateTrack

Message name	Description
CreateTrackRequest	*The **RecordingToken** shall identify the recording to which a track shall be added. The **TrackConfiguration** shall provide the configuration for the new track.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackConfiguration **TrackConfiguration**[1][1]
CreateTrackResponse	*The **TrackToken** shall identify the newly created track. The **TrackToken** shall be unique within the recoding to which the new track belongs.* tt:TrackReference **TrackToken**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*
env:Receiver ter:Action ter:MaxTracks	*The new track cannot be created because the maximum number of tracks that the device supports for this recording has been reached.*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The TrackConfiguration is invalid.*
env:Receiver ter:ActionNotSupported ter:NotImplemented	*This optinal method is not implemented*

A TrackToken in itself does not uniquely identify a specific track. Tracks within different recordings may have the same TrackToken.

19.10 DeleteTrack

DeleteTrack shall remove a track from a recording. All the data in the track shall be deleted.

This method is optional. It shall be available if the Recording/DynamicTracks capability is TRUE.

Table 266: DeleteTrack command

DeleteTrack	
Message name	**Description**
DeleteTrackRequest	*The **RecordingToken** shall identify the recording from which to delete the track. The **TrackToken** identifies the track to delete.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1]
DeleteTrackResponse	*This message shall be empty.*
Fault codes	**Description**
env:Receiver ter:ActionNotSupported	*The device does not implement the DeleteTrack method.*

ter:NotImplemented	
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*
env:Receiver ter:Action ter:CannotDelete	*This specific track cannot be deleted*

19.11 GetTrackConfiguration

GetTrackConfiguration shall retrieve the configuration for a specific track.

Table 267: GetTrackConfiguration command

GetTrackConfiguration	
Message name	**Description**
GetTrackConfigurationRequest	*The **RecordingToken** and **TrackToken** shall identify the recording from which to get the track configuration.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1]
GetTrackConfigurationResponse	tt:TrackConfiguration **TrackConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

19.12 SetTrackConfiguration

SetTrackConfiguration shall change the configuration of a track.

Table 268: SetTrackConfiguration command

SetTrackConfiguration	
Message name	**Description**
SetTrackConfigurationRequest	*The **RecordingToken** and **TrackToken** shall identify the track for which to set the track configuration. The **TrackConfiguration** is the new configuration for the track.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1] tt:TrackConfiguration **TrackConfiguration**[1][1]
SetTrackConfigurationResponse	This message shall be empty.

Fault codes	Description
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The contents of the configuration object are invalid.*

19.13 CreateRecordingJob

CreateRecordingJob shall create a new recording job.

Table 269: CreateRecordingJob command

CreateRecordingJob	
Message name	**Description**
CreateRecordingJobRequest	***JobConfiguration*** *shall hold the configuration for the new recording job.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
CreateRecordingJobResponse	*The* ***JobToken*** *shall identify the created recording job. The* ***JobConfiguration*** *structure shall be the configuration as it is used by the device. This may be different from the JobConfiguration passed to CreateRecordingJob.* tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:MaxRecordingJobs	*The maximum number of recording jobs that the device can handle has been reached.*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The contents of the JobConfiguration are invalid.*
env:Receiver ter:Action ter:MaxReceivers	*If the AutoCreateReceivers flag is TRUE, this error can be returned if the receiver service cannot create a new receiver.*

The **JobConfiguration** returned from CreateRecordingJob shall be identical to the **JobConfiguration** passed into CreateRecordingJob, except for the ReceiverToken and the AutoCreateReceiver. In the returned structure, the ReceiverToken shall be present and valid and the AutoCreateReceiver field shall be omitted.

19.14 DeleteRecordingJob

DeleteRecordingJob removes a recording job. It shall also implicitly delete all the receiver objects associated with the recording job that are automatically created using the

AutoCreateReceiver field of the recording job configuration structure and are not used in any other recording job.

Table 270: DeleteRecordingJob command

DeleteRecordingJob	
Message name	**Description**
DeleteRecordingJobRequest	*The **JobToken** shall identify the recording job that shall be deleted.* tt:RecordingJobReference **JobToken**[1][1]
DeleteRecordingJobResponse	*The message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

19.15 GetRecordingJobs

GetRecordingJobs shall return a list of all the recording jobs in the device.

Table 271: GetRecordingJobs command

GetRecordingJobs	
Message name	**Description**
GetRecordingJobsRequest	*This message shall be empty.*
GetRecordingJobsResponse	*The **JobItem** identifies a job in the device and holds its current configuration.* tt:GetRecordingJobsResponseItem **JobItem**[0][unbounded]
Fault codes	**Description**
No command specific faults	

19.16 SetRecordingJobConfiguration

SetRecordingJobConfiguration shall change the configuration for a recording job.

Table 272: SetRecordingJobConfiguration command

SetRecordingJobConfiguration	
Message name	**Description**
SetRecordingJobConfigurationRequest	*The JobConfiguration returned from SetRecordingJobConfiguration shall be identical to the JobConfiguration passed into SetRecordingJobConfiguration, except for the ReceiverToken and the AutoCreateReceiver. In the returned structure, the ReceiverToken shall be present and valid and the AutoCreateReceiver field shall be omitted.*

	tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobConfiguration **JobConfiguration**[1][1]
SetRecordingJobConfigurationResponse	*The **JobConfiguration** structure shall be the configuration as it is used by the device. This may be different from the JobConfiguration passed to CreateRecordingJob.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The contents of the JobConfiguration are invalid.*
env:Receiver ter:Action ter:MaxReceivers	*If the AutoCreateReceivers flag is TRUE, this error can be returned if the receiver service cannot create a new receiver.*

SetRecordingJobConfiguration shall implicitly delete any receiver objects that were created automatically if they are no longer used as a result of changing the recording job configuration.

19.17 GetRecordingJobConfiguration

GetRecordingJobConfiguration shall return the current configuration for a recording job.

Table 273: GetRecordingJobConfiguration command

GetRecordingJobConfiguration	
Message name	**Description**
GetRecordingJobConfiguration Request	*The **JobToken** shall identify the recording job for which to retrieve the configuration.* tt:RecordingJobReference **JobToken**[1][1]
GetRecordingJobConfiguration Response	*The **JobConfiguration** shall hold the current configuration of the recording job.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

19.18 SetRecordingJobMode

SetRecordingJobMode shall change the mode of the recording job. Using this method shall be equivalent to retrieving the recording job configuration, and writing it back with a different mode.

Table 274: SetRecordingJobMode command

SetRecordingJobMode

Message name	Description
SetRecordingJobModeRequest	*The **JobToken** shall identify the recording job for which to change the recording mode. The **mode** shall be the new mode for the recording job.* tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobMode **Mode**[1][1]
SetRecordingJobModeResponse	*This message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*
env:Sender ter:InvalidArgVal ter:BadMode	*The Mode is invalid.*

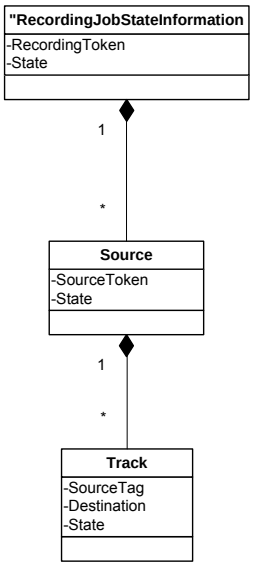
19.19 GetRecordingJobState

GetRecordingJobState returns the state of a recording job. It includes an aggregated state, and state for each track of the recording job.

Table 275: GetRecordingJobState command

GetRecordingJobState	
Message name	**Description**
GetRecordingJobState Request	*The **JobToken** shall identify the recording job for which to get the state.* tt:RecordingJobReference **JobToken**[1][1]
GetRecordingJobState Response	*The **State** shall hold the state of the recording job.* tt:RecordingJobStateInformation **State**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

The UML representation of the RecordingJobStateInformation structure is:



RecordingToken shall be the identification of the recording that the recording job records to.

State (as part of RecordingJobStateInformation) shall hold the aggregated state over the whole RecordingJobInformation structure.

SourceToken shall identify the data source of the recording job.

State (as part of RecordingJobStateSource) shall hold the aggregated state over all substructures of RecordingJobStateSource.

SourceTag shall identify the track of the data source that provides the data.

Destination shall indicate the destination track

State (as part of RecordingJobTrackState) shall provide the job state of the track. The valid values of state shall be “Idle”, “Active” and “Error”. If state equals “Error”, the Error field may be filled in with an implementation defined value.

Error, if present, shall hold an implementation defined string value that describes the error. The string should be in the English language.

A device shall apply the following rules to compute aggregate state

Idle	All state values in sub-nodes are “idle”
PartiallyActive	The state of some sub-nodes are “active” and some sub-nodes are “idle”
Active	The state of all sub-nodes is “Active”
Error	At least one of the sub-nodes has state “Error”

19.20 Events

The recording service shall dispatch events through the event service. It shall be capable of generating the events listed in this chapter whenever the condition that fires the event occurs.

Some of these events are similar to the automatically generated events that can be searched for by the FindEvents method in the search service. See section 20.17.

19.20.1 Recording job state changes

If the a state field of the RecordingJobStateInformation structure changes, the device shall send the event:

```
Topic: tns1:RecordingConfig/JobState
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="xs:String"/>
    <tt:ElementItemDescription Name="Information"
Type="tt:RecordingJobStateInformation"/>
  </tt:Data>
</tt:MessageDescription>
```

19.20.2 Configuration changes

If the configuration of a recording is changed, the device shall send the event:

```
Topic: tns1:RecordingConfig/RecordingConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:RecordingConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

If the configuration of a track is changed, the device shall send the event:

```
Topic: tns1:RecordingConfig/TrackConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:TrackConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

If the configuration of a recording job is changed, the device shall send the event:

```
Topic: tns1:RecordingConfig/RecordingJobConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration"
Type="tt:RecordingJobConfiguration"/>
```

```
  </tt:Data>
</tt:MessageDescription>
```

19.20.3 Data deletion

Whenever data is deleted, the device shall send the event:

```
Topic: tns1:RecordingConfig/DeleteTrackData
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="StartTime" Type="xsDateTime"/>
    <tt:SimpleItemDescription Name="EndTime" Type="xsDateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

19.20.4 Recording and track creation and deletion

Whenever a recording is created, the device shall send the event:

```
Topic: tns1:RecordingConfig/CreateRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a recording is deleted, the device shall send the event:

```
Topic: tns1:RecordingConfig/DeleteRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a track is created, the device shall send the event:

```
Topic: tns1:RecordingConfig/CreateTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a track is deleted, the device shall send the event:

```
Topic: tns1:RecordingConfig/DeleteTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

19.21 Examples

19.21.1 Example 1: setup recording of a single camera

There are two steps involved. The first step is to configure the NVS

```
; Create recording (this implicitly creates an A, V and M track)
RecordToken = CreateRecording(RecordConfiguration)

; The tracktokens are predefined. We don't have to find them on the device
TrackToken1 = "VIDEO001"
TrackToken2 = "AUDIO001"
TrackToken3 = "META001"

; Create a recording job, assume that we set mode to idle, auto create
receiver
JobToken, ActualJobConfig = CreateRecordingJob(JobConfiguration)

; Configure the receiver
ConfigureReceiver(ActualJobConfiguration.ReceiverToken,
ReceiverConfiguration)
```

This completes the configuration step.

Finally, to really start recording, some entity calls

```
; Activate the recording job
SetRecordingJobMode(JobToken, Active)
```

to make the job active. This will cause the NVS to set up an RTSP connection with the device.

Therefore, to start and stop recording, all that is needed is to call SetRecordingJobMode on pre-configured recording jobs. And since the embedded configuration objects are persistent, the configuration cycle only needs to be done once.

19.21.2 Example 2: Record multiple streams from one camera to a single recording

This example is very similar to example 1. The jobconfiguration will hold references to two receiver objects. Each receiver object is configured to receive from the same device, but from a different stream.

```
; Create recording (this implicitly creates an A, V and M track)
RecordToken = CreateRecording(RecordConfiguration)

; The tracktokens are predefined. We don't have to find them on the device
TrackToken1 = "VIDEO001"
TrackToken2 = "AUDIO001"
TrackToken3 = "META001"

; Create three additional tracks
TrackToken4 = CreateTrack(RecordToken, AudioConfig)
TrackToken5 = CreateTrack(RecordToken, VideoConfig)
TrackToken6 = CreateTrack(RecordToken, MetadataConfig)


; Create a recording job, assume that we set mode to idle,auto create two
receivers
JobToken, ActualJobConfiguration = CreateRecordingJob(JobConfiguration)


; Configure the receivers
ConfigureReceiver(ActualJobConfiguration.ReceiverToken[1],
                                                Receiver1Configuration)
ConfigureReceiver(ActualJobConfiguration.ReceiverToken[2],
                                                Receiver2Configuration)
```

To really start recording, some entity calls

```
; Activate the recording job
SetRecordingJobMode(JobToken, Active)
```

20 Recording Search

20.1 Introduction

The search service provides a number of operations for finding data of interest within a set of recordings. The most common way of doing this would be to search for events that are either included in the metadata track of a recording, or are otherwise associated with a recording in the device (see Recording Events below).

GetRecordingSummary returns a summary for all recording, and can be used to provide the scale of a timeline.

GetRecordingInformation returns information about a single recording, such as start time and current status.

GetMediaAttributes returns the media attributes of a recording at a specific point in time.

The actual search is done by coupled find and result operations. Each find operation initiates a search session. The client can then acquire the results from the search session in increments, or all at once, depending on implementation and the scale of the search. There are four pairs of search operations for Recordings, Recording Events, PTZ Positions and Metadata.

GetSearchState returns the state of a search session.

EndSearch ends a search session, halting search and returning any blocking result operations.

20.2 Concepts

20.2.1 Search Direction

Search is performed from a start point on the time line, towards an end point. If the end point is prior to the start point, search will be performed backwards. This can be useful if only the newest matching event is of interest, or if it is otherwise convenient to get the results in newest to oldest order.

If no end point is specified, the search will always be performed forward in time from the start point.

20.2.2 Recording Event

Describes a discrete event related to the recording. It is represented as a notification message, but this does not necessarily mean it has been recorded as a notification. Recording events can either be notifications included in a recorded metadata track, it can be created by the recording device as a result of an internal event or mechanism, or it can be inserted by a client using a WebService request or a metadata stream. However the recording event has been created and associated with a particular recording, this specification makes no implications on how it is stored internally on a device, only how it should be represented in the interface.

However created, recording events are always treated as notifications in regards to search filters and results returned. Each recording event has a notification topic as defined in 15.7 of the core specification. Predefined recording events are described in section 20.17.

To communicate the original state of property events, virtual start state events can be returned in a search result containing the value of one or more properties at the start point of the search interval. Such start state events are virtual events in the sense that they are created on the fly by the server, rather than being collected from recorded data. If the client indicates that such events are desired by setting the appropriate flag, virtual events matching the topics defined in the search filter should be returned for any recording in the search scope.

20.2.3 Search Session

A search session is started asynchronously by a Find-operation and is identified by a search token unique for that session. Results are returned in increments using GetResult-operations referring to the session created by the Find-operation. The search can be terminated in three ways:

- KeepAlive time expires – If no request from a client has been made that refers to a particular session within the specified time interval, it will terminate.
- A GetResult method returns the last data for the search session by setting the search state in its result to “Completed”.
- EndSearch – The client explicitly ends a session.

Ending a session will cancel an ongoing search, immediately return and make further requests to the same session result in an error message. A device shall not reuse search token immediately as it would confuse clients unaware that a session had ended.

20.2.4 Search Scope

The scope contains a number of optional elements, together limiting the set of data to look into when performing searches.

20.2.4.1 Included data

Optionally, the client can define sources and recordings to search in by specifying lists of tokens for each type. If several types are given, the union of the specified tokens shall be used. If there are no sources or recordings tokens specified, all recordings shall be included. The scope is further refined by the Recording Information Filter. However, if recordings are specified the filters will only be applied to that subset of recordings.

20.2.4.2 Recording Information Filter

Rather than specifying a list of recording tokens, the recordings can be filtered by an XPath filter operating on the RecordingInformation structure. This allows the client to filter on all elements present in the RecordingInformation structure, using comparisons according to the XPath dialect defined in section 20.18. If a recording information filter is supplied, only recordings matching the filter shall be part of the scope.

Example of a filter that includes only recordings containing audio in the search scope:

```
boolean(//Tracks[TrackType = “Audio”])
```

20.2.5 Search Filters

Search filters are specific for the type of search operation. See FindEvents, FindPTZPosition, FindMetadata respectively. They all act on the recordings defined by the scope.

20.3 Data Structures

20.3.1 RecordingInformation Structure

RecordingInformation contains information about a recording, the tracks it consists of and the source.

- RecordingToken – a unique identifier of the recording.
- EarliestRecording – the date and time of the oldest data in the recording
- LatestRecording – the date and time of the newest data in the recording.
- Content – informative description of content.
- RecordingStatus – current status of recording, can be any of: Initiated, Recording, Stopped, Removing, Removed.
- RecordingSourceInformation – a structure containing information about the source of the recording.
- TrackInformation – a list of track information structures.

20.3.2 RecordingSourceInformation Structure

Contains information about the source of a recording.

- SourceId – an identifier for the source chosen by the client that creates the recording. This identifier is opaque to the NVS. Clients may use any type of URI for this field..
- Name – informative name of the source.
- Location – informative description of the location of the source.
- Description – informative description of the source.
- Address – informative URI of the source.

20.3.3 TrackInformation Structure

Contains information about a single track in a recording.

- TrackToken – an identifier of the track. The TrackToken is unique between all TrackTokens used within a recording.
- TrackType – identifies the type of track (video, audio or metadata)
- Description – informative description of the track.
- DataFrom – The date and time of the oldest recorded data in the track.

- DataTo – The date and time of the newest recorded data in the track.

20.3.4 SearchState Enumeration

The search state can be one of the following

- Queued – meaning that the search has not yet begun.
- Searching – meaning that the search is under way, and new results can be produced.
- Completed – meaning that the search is completed and no new results will be produced.

20.3.5 MediaAttributes Structure

The MediaAttributes contains information about the media tracks of a particular recording for a particular time frame. The time frame can be a single point in time, in which case the *From* and *Until* elements are identical.

- RecordingToken – A reference to the recording that this structure concerns.
- From – a point in time from when the attributes are valid for the recording.
- Until – a point in time until when the specified attributes are valid for the recording.
- VideoAttributes - A set of video attributes, describing the data of a recorded video track.
- AudioAttributes - A set of audio attributes, describing the data of a recorded audio track.
- MetadataAttributes - A set of attributes, describing the possible metadata content of a recorded metadata track.

20.3.6 FindEventResult Structure

- RecordingToken – identifying the recording containing the found event.
- TrackToken – identifying the track containing the found event.
- Time – the date and time of the found event.
- Event – the event message found.
- StartStateEvent – if true, indicates the event represents the start state of one or more properties in the recording.

20.3.7 FindPTZPositionResult Structure

- RecordingToken – identifying the recording containing the matching position.
- TrackToken – identifying the track containing the matching position.
- Time – the date and time of the matching position.

- Position – the matching PTZ vector.

20.3.8 PTZPositionFilter Structure

Contains the necessary elements to define what PTZ positions to search for. The PTZ vectors shall be in the same coordinate space as the PTZ coordinates stored in the recording.

- MinPosition - The lower boundary of the PTZ volume to look for.
- MaxPosition - The upper boundary of the PTZ volume to look for.
- EnterOrExit - If true, search for when entering or exiting the specified PTZ volume.

20.3.9 MetadataFilter Structure

Contains an XPath expression to be applied to the MetadataStream structure.

Example of an expression searching for objects overlapping the lower right quadrant of the scene:

```
boolean(//Object/Appearance/Shape/BoundingBox[@right > ″0.5″])
and boolean(//Object/Appearance/Shape/BoundingBox[@bottom
> ″0.5″])
```

20.3.10 FindMetadataResult Structure

- RecordingToken – identifying the recording containing the matching metadata.
- TrackToken – identifying the track containing the matching metadata.
- Time – the date and time of the matching metadata.

20.4 GetRecordingSummary

GetRecordingSummary is used to get a summary description of all recorded data. This operation is mandatory to support for a device implementing the recording search service.

Table 276: GetRecordingSummary command

GetRecordingSummary	Request-Response
Message name	**Description**
GetRecordingSummaryRequest	This shall be the empty message
GetRecordingSummaryResponse	Returns a structure containing: *DataFrom* specifying the first time when there is recorded data on the device; *DataUntil* specifying the last point in time where there is data recorded on the device; the estimated total number of recordings and tracks. tt:RecordingSummary summary[1][1]
Fault codes	**Description**

	No command specific error codes

20.5 GetRecordingInformation

Returns information about a single *Recording* specified by a *RecordingToken.* This operation is mandatory to support for a device implementing the recording search service.

Table 277: GetRecordingInformation command

GetRecordingInformation	Request-Response
Message name	**Description**
GetRecordingInformationRequest	*Request description* tt:ReferenceToken **RecordingToken** [1][1]
GetRecordingInformationResponse	*Response description* tt:RecordingInformation **RecordingInformation** [1][1]
Fault codes	**Description**
env:Sender ter: InvalidArgVal ter: InvalidToken	*The RecordingToken is not valid.*

20.6 GetMediaAttributes

Returns a set of media attributes for all tracks of the specified recordings at a specified point in time. Clients using this operation shall be able to use it as a non blocking operation. A device shall set the starttime and endtime of the MediaAttributes structure to equal values if calculating this range would causes this operation to block. See MediaAttributes structure for more information. This operation is mandatory to support for a device implementing the recording search service.

Table 278: GetMediaAttributes command

GetMediaAttributes	Request-Response
Message name	**Description**
GetMediaAttributesRequest	*RecordingTokens* is a list of references to the recordings to query. *Time* is the point in time where for which information is requested. tt:ReferenceToken **RecordingTokens** [0][unbounded] xs:dateTime **Time**[1][1]
GetMediaAttributesResponse	Contains a *MediaAttributes* structure for each RecordingToken specified in the request. tt:MediaAttributes **MediaAttributes** [0][unbounded]

Fault codes	Description
env:Sender ter:InvalidArgVal ter:InvalidToken	*The RecordingToken is not valid.*

20.7 FindRecordings

FindRecordings starts a search session, looking for recordings that matches the scope (See 20.2.4) defined in the request. Results from the search are acquired using the GetRecordingSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- The entire time range from *StartPoint* to *EndPoint* has been searched through.
- The total number of matches has been found, defined by the *MaxMatches* parameter.
- The session has been ended by a client EndSession request.
- The session has been ended because *KeepAliveTime* since the last request related to this session has expired.

The order of the results is undefined, to allow the device to return results in any order they are found. This operation is mandatory to support for a device implementing the recording search service.

Table 279: FindRecordings command

FindRecordings	Request-Response
Message name	**Description**
FindRecordingsRequest	*Scope* defines the dataset to consider for this search. The search ends after *MaxMatches*. *KeepAliveTime* is the session timeout after each request concerning this session. tt:SearchScope**Scope** [1][1] xs:int **MaxMatches** [0][1] xs:duration **KeepAliveTime** [1][1]
FindRecordingsResponse	Returns the *SearchToken* identifying the search session created by this request. tt:JobToken **SearchToken** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:ResourceProblem	*Device is unable to create a new search session.*

20.8 GetRecordingSearchResults

GetRecordingSearchResults acquires the results from a recording search session previously initiated by a FindRecordings operation. The response shall not include results already returned in previous requests for the same session. If *MaxResults* is specified, the response shall not contain more than *MaxResults* results.

GetRecordingSearchResults shall block until:

- *MaxResults* results are available for the response if *MaxResults* is specified.
- *MinResults* results are available for the response if *MinResults* is specified.
- *WaitTime* has expired.
- Search is completed or stopped.

This operation is mandatory to support for a device implementing the recording search service.

Table 280: GetRecordingSearchResults command

GetRecordingSearchResults	Request-Response
Message name	**Description**
GetRecordingSearchResultsRequest	*SearchToken* specifies the search session. *MinResults* specifies the minimum number of results that should be returned. If the total number of results is lower than *MinResults* in a completed search, all results should be returned. *MaxResults* specifies the maximum number of results to return. *WaitTime* defines the maximum time to block, waiting for results. tt:JobToken **SearchToken** [1][1] xs:int **MinResults** [0][1] xs:int **MaxResults** [0][1] xs:duration **WaitTime** [0][1]
GetRecordingSearchResultsResponse	Returns a structure containing the current *SearchState* and a list of *RecordingInformation* structures. tt:FindRecordingResultList ResultList[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.9 FindEvents

FindEvents starts a search session, looking for recording events (see 20.2.2) in the *scope* (See 20.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetEventSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- The entire time range from *StartPoint* to *EndPoint* has been searched through.
- The total number of matches has been found, defined by the *MaxMatches* parameter.
- The session has been ended by a client EndSession request.
- The session has been ended because *KeepAliveTime* since the last request related to this session has expired.

Results shall be ordered by time, ascending in case of forward search, or descending in case of backward search. This operation is mandatory to support for a device implementing the recording search service.

Table 281: FindEvents command

FindEvents	Request-Response
Message name	**Description**
FindEventsRequest	*StartPoint* is the point of time where the search will start. *EndPoint* is The point of time where the search will stop. This can be a time before the *StartPoint*, in which case the search is performed backwards in time. If *EndPoint* is omitted, search will go forward from the *StartPoint*. *Scope* defines the dataset to consider for this search. *SearchFilter* contains the topic and message filter needed to define what events to search for. By setting the *IncludeStartState* to true, the client indicates that virtual events at the time of *StartPoint* should be returned to represent the state in the recording. The search ends after *MaxMatches. KeepAliveTime* is the session timeout after each request concerning this session. xs:dateTime **StartPoint** [1][1] xs:dateTime **EndPoint** [0][1] tt:SearchScope**Scope** [1][1] tt:EventFilter **SearchFilter** [1][1] xs:boolean **IncludeStartState** [1][1] xs:int **MaxMatches** [0][1] xs:duration **KeepAliveTime** [1][1]
FindEventsResponse	Returns the *SearchToken* identifying the search session created by this request. tt:JobToken **SearchToken** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:ResourceProblem	*Device is unable to create a new search session.*

20.10 GetEventSearchResults

GetEventSearchResults acquires the results from a recording event search session previously initiated by a FindEvents operation. The response shall not include results already returned in

previous requests for the same session. If *MaxResults* is specified, the response shall not contain more than *MaxResults* results.

GetEventSearchResults shall block until:

- *MaxResults* results are available for the response if *MaxResults* is specified.
- *MinResults* results are available for the response if *MinResults* is specified.
- *WaitTime* has expired.
- Search is completed or stopped.

This operation is mandatory to support for a device implementing the recording search service.

Table 282: GetEventSearchResults command

GetEventSearchResults	Request-Response
Message name	**Description**
GetEventSearchResultsRequest	*SearchToken* specifies the search session. *MinResults* specifies the minimum number of results that should be returned. *MaxResults* specifies the maximum number of results to return. *WaitTime* defines the maximum time to block, waiting for results. tt:JobToken **SearchToken** [1][1] xs:int **MinResults** [0][1] xs:int **MaxResults** [0][1] xs:duration **WaitTime** [0][1]
GetEventSearchResultsResponse	Returns a structure containing the current *SearchState* and a list of *FindEventResult* structures. tt:SearchState **SearchState** [1][1] tt:FindEventResult **FindEventResult** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.11 FindPTZPosition

FindPTZPosition starts a search session, looking for ptz positions in the *scope* (See 20.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetPTZPositionSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- The entire time range from *StartPoint* to *EndPoint* has been searched through.

- The total number of matches has been found, defined by the *MaxMatches* parameter.
- The session has been ended by a client EndSession request.
- The session has been ended because *KeepAliveTime* since the last request related to this session has expired.

This operation is mandatory to support whenever CanContainPTZ is true for any metadata track in any recording on the device.

Table 283: FindPTZPosition command

FindPTZPosition	Request-Response
Message name	**Description**
FindPTZPositionRequest	*StartPoint* is the point of time where the search will start. *EndPoint* is The point of time where the search will stop. This can be a time before the *StartPoint*, in which case the search is performed backwards in time. If *EndPoint* is omitted, search will go forward from the *StartPoint*. *Scope* defines the dataset to consider for this search. *SearchFilter* contains the search criteria needed to define the PTZ position to search for. The search ends after *MaxMatches. KeepAliveTime* is the session timeout after each request concerning this session. xs:dateTime **StartPoint** [1][1] xs:dateTime **EndPoint** [0][1] tt:SearchScope**Scope** [1][1] tt:PTZPositionFilter **SearchFilter** [1][1] xs:int **MaxMatches** [0][1] xs:duration **KeepAliveTime** [1][1]
FindPTZPositionResponse	Returns the *SearchToken* identifying the search session created by this request. tt:JobToken **SearchToken** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:ResourceProblem	*Device is unable to create a new search session.*
env:Receiver ter:ActionNotSupported ter:NotImplemented	*This optinal method is not implemented*

20.12 GetPTZPositionSearchResults

GetPTZPositionSearchResults acquires the results from a ptz position search session previously initiated by a FindPTZPosition operation. The response shall not include results already returned in previous requests for the same session. If *MaxResults* is specified, the response shall not contain more than *MaxResults* results.

GetPTZPositionSearchResults shall block until:

- *MaxResults* results are available for the response if *MaxResults* is specified.
- *MinResults* results are available for the response if *MinResults* is specified.
- *WaitTime* has expired.
- Search is completed or stopped.

This operation is mandatory to support whenever CanContainPTZ is true for any metadata track in any recording on the device.

Table 284: GetPTZPositionSearchResults command

GetPTZPositionSearchResults	Request-Response
Message name	**Description**
GetPTZPositionSearchResultsRequest	*SearchToken* specifies the search session. *MinResults* specifies the minimum number of results that should be returned. *MaxResults* specifies the maximum number of results to return. *WaitTime* defines the maximum time to block, waiting for results. tt:JobToken **SearchToken** [1][1] xs:int **MinResults** [0][1] xs:int **MaxResults** [0][1] xs:duration **WaitTime** [0][1]
GetPTZPositionSearchResultsResponse	Returns a structure containing the current *SearchState* and a list of *FindePTZPositionResult* structures. tt:SearchState **SearchState** [1][1] tt:FindPTZPositionResult **FindPTZPositionResult** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.13 FindMetadata

FindMetadata starts a search session, looking for metadata in the scope (See 20.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetMetadataSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- The entire time range from *StartPoint* to *EndPoint* has been searched through.
- The total number of matches has been found, defined by the *MaxMatches* parameter.
- The session has been ended by a client EndSession request.

- The session has been ended because *KeepAliveTime* since the last request related to this session has expired.

This operation is mandatory to support if the MetaDataSearch capability is set to true in the SearchCapabilities structure return by the GetCapabilities command in the Device service.

Table 285: FindMetadata command

FindMetadata	Request-Response
Message name	**Description**
FindMetadataRequest	*StartPoint* is the point of time where the search will start. *EndPoint* is The point of time where the search will stop. This can be a time before the *StartPoint*, in which case the search is performed backwards in time. If *EndPoint* is omitted, search will go forward from the *StartPoint*. *Scope* defines the dataset to consider for this search. *SearchFilter* contains the search criteria needed to define the metadata to search for. The search ends after *MaxMatches. KeepAliveTime* is the session timeout after each request concerning this session. xs:dateTime **StartPoint** [1][1] xs:dateTime **EndPoint** [0][1] tt:SearchScope**Scope** [1][1] tt:MetadataFilter **SearchFilter** [1][1] xs:int **MaxMatches** [0][1] xs:duration **KeepAliveTime** [1][1]
FindMetadataResponse	Returns the *SearchToken* identifying the search session created by this request. tt:JobToken **SearchToken** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:ResourceProblem	*Device is unable to create a new search session.*

20.14 GetMetadataSearchResults

GetMetadataSearchResults acquires the results from a recording search session previously initiated by a FindMetadata operation. The response shall not include results already returned in previous requests for the same session. If *MaxResults* is specified, the response shall not contain more than *MaxResults* results.

GetMetadataSearchResults shall block until:

- *MaxResults* results are available for the response if *MaxResults* is specified.
- *MinResults* results are available for the response if *MinResults* is specified.
- *WaitTime* has expired.
- Search is completed or stopped.

This operation is mandatory to support if the MetaDataSearch capability is set to true in the SearchCapabilities structure return by the GetCapabilities command in the Device service.

Table 286: GetMetadataSearchResults command

GetMetadataSearchResults	Request-Response
Message name	**Description**
GetMetadataSearchResults Request	*SearchToken* specifies the search session. *MinResults* specifies the minimum number of results that should be returned. *MaxResults* specifies the maximum number of results to return. *WaitTime* defines the maximum time to block, waiting for results. tt:JobToken **SearchToken** [1][1] xs:int **MinResults** [0][1] xs:int **MaxResults** [0][1] xs:duration **WaitTime** [0][1]
GetMetadataSearchResults Response	Returns a structure containing the current *SearchState* and a list of *FindMetadataResult* structures. tt:SearchState **SearchState** [1][1] tt:FindMetadataResult **FindMetadataResult** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.15 GetSearchState

GetSearchState returns the current state of the specified search session. Queued, Searching or Completed. This operation is mandatory to support for a device implementing the recording search service.

Table 287: GetSearchState command

GetSearchState	Request-Response
Message name	**Description**
GetSearchStateRequest	*SearchToken* specifies the search session. tt:JobToken **SearchToken** [1][1]
GetSearchStateResponse	Returns the current state of the search session. tt:SearchState **State** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.16 EndSearch

EndSearch stops and ongoing search session, causing any blocking result request to return and the *SearchToken* to become invalid. If the search was interrupted before completion, the point in time that the search had reached shall be returned. If the search had not yet begun, the *StartPoint* shall be returned. If the search was completed the original *EndPoint* supplied by the Find operation shall be returned. This operation is mandatory to support for a device implementing the recording search service.

Table 288: EndSearch command

EndSearch	Request-Response
Message name	**Description**
EndSearchRequest	*SearchToken* specifies the search session. tt:JobToken **SearchToken** [1][1]
EndSearchResponse	Returns the point in time where the search was at when ended. xs:dateTime **EndPoint** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidToken	*The search token is invalid.*

20.17 Recording Event Descriptions

A device shall generate the following events with the corresponding event message descriptions. A device supporting the recording search service shall record these notification messages so that clients can use FindEvents to search for these messages. All recording events that are generated by the device and inserted into the recording history shall have a root topic of tns1:RecordingHistory.

```
Topic: tns1:RecordingHistory/Recording/State
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsRecording" Type="tt:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

This message is sent whenever a client starts or stops recording for a specific recording. At start recording, IsRecording shall be set to true. At stop recording, IsRecording shall be set to false.

Topic: tns1:RecordingHistory/Track/State

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
```

```
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsDataPresent" Type="tt:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

This message signals when the data for a track is present. When the data becomes present, a message with IsDataPresent set to TRUE shall be sent. When the data becomes unavailable, The message with IsDataPresent set to FALSE shall be sent.

An NVS MAY generate the following events. If the NVS supports these events, it shall always automatically records these notification messages so that clients can always use FindEventfor these messages.

Topic: tns1:RecordingHistory/Track/VideoParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="VideoEncoding"
Type="tt:VideoEncoding"/>
    <tt:SimpleItemDescription Name="VideoWidth" Type="xs:int"/>
    <tt:SimpleItemDescription Name="VideoHeight" Type="xs:int"/>
    <tt:SimpleItemDescription Name="tt:RateControl"
Type="VideoRateControl"/>
  </tt:Data>
</tt:MessageDescription>
```

Topic: tns1:RecordingHistory/Track/AudioParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="AudioEncoding"
Type="tt:AudioEncoding"/>
    <tt:SimpleItemDescription Name="AudioSampleRate" Type="xs:int"/>
    <tt:SimpleItemDescription Name="AudioBitrate" Type="xs:int"/>
  </tt:Data>
</tt:MessageDescription>
```

The NVS shall send either message (depending on the track data type) whenever any of these properties change.

20.18 XPath dialect

This section defines the XPATH dialect that a device that realises the search service shall implement to parse the XPath strings that are passed to the methods of the search service.

```
Dialect=http://www.onvif.org/ver10/tse/searchFilter

[1] Expression ::= BoolExpr | Expression 'and' Expression
      | Expression 'or' Expression | '('Expression')' |
'not''('Expression')'
[2] BoolExpr ::= 'boolean''('PathExpr')' | 'contains''(' ElementPath ','
'"' String '"' ')'
[3] PathExpr ::= '//SimpleItem' NodeTest | '//ElementItem' NodeTest |
      ElementTest
[4] NodeTest ::= '['AttrExpr']'
[5] AttrExpr ::= NameComp | ValueComp | AttrExpr 'and' AttrExpr | AttrExpr
'or' AttrExpr | 'not''(AttrExpr')'
[6] NameComp ::= NameAttr'='"'String'"'
[7] ValueComp ::= ValueAttr Operator '"'String'"'
[8] Operator ::= '=' | '!=' | '<' | '<=' | '>' | '>='
[9] NameAttr ::= '@Name'
[10] ValueAttr ::= '@Value'
[11] ElementTest ::= '/' ElementPath '['NodeComp']'
[12] ElementPath ::= ElementName ElementName*
[13] ElementName ::= '/' String
[14] NodeComp ::= NodeName Operator '"' String '"'
[15] NodeName ::= '@' String | String
```

Example of an XPath expression used to find recordings from the basement where there is at least one track containing video:

```
boolean(//Source[Location = “Basement”]) and
boolean(//Tracks[TrackType = “Video”])
```

21 Replay Control

This section describes the use of RTSP for replaying recorded streams, and defines a service for mapping replay endpoints to URI for use in RTSP.

21.1 Use of RTSP

The replay protocol is based on RTSP [RFC 2326]. However because RTSP does not directly support many of the features required by CCTV applications, this standard defines several extensions to the protocol; these are detailed below.

This standard makes the following stipulations on the usage of RTSP:

1. RTP/RTSP/HTTP/TCP shall be supported by the server. This is the same transport protocol as a device that implements media streaming through the media service shall support, and the same requirements shall apply to replay streaming.
2. The server shall support the unicast RTP/UDP transport for streaming.
3. Clients should use a TCP-based transport for replay, in order to achieve reliable delivery of media packets.
4. The server MAY elect not to send RTCP packets during replay. In typical usage RTCP packets are not required, because usually a reliable transport will be used, and because absolute time information is sent within the stream, making the timing information in RTCP sender reports redundant.

21.1.1 RTSP describe

The SDP returned by the RTSP describe command shall include the TrackReference for each track of the recording to allow a client to map the tracks presented in the SDP to tracks of the recording. The tag shall use the following format:

a:x-onvif-track:<TrackReference>

For example:

```
NVS – NVT:  DESCRIBE rtsp://192.168.0.1 RTSP/1.0
                  CSeq: 1
                  User-Agent: ONVIF Rtsp client
                  Accept: application/sdp

NVT – NVS:  RTSP/1.0 200 OK
                  CSeq: 1
                  Content-Type: application/sdp
                  Content-Length: xxx

v=0
                  o= 2890842807 IN IP4 192.168.0.1
                  m=video 0 RTP/AVP 26
                        a=control:rtsp://192.168.0.1/video
                  a=x-onvif-track:VIDEO001
                  m=audio 0 RTP/AVP 98
                  a=control:rtsp://192.168.0.1/audio
                  a=x-onvif-track:AUDIO001
```

21.2 RTP header extension

In order to allow clients to report a stable and accurate timestamp for each frame played back regardless of the direction of playback, it is necessary to associate an absolute timestamp with each packet, or each group of packets with the same RTP timestamp (e.g. a video frame).

This is achieved using an RTP header extension containing an NTP timestamp and some additional information also useful for replay.

The replay mechanism uses the extension ID 0xABAC for the replay extension.

Below shows the general form of an RTP packet containing this extension:

Table 289: RTP packet layout

<table>
<tr><td>V=2</td><td>P</td><td>X=1</td><td>CC</td><td>M</td><td>PT</td><td>sequence number</td></tr>
<tr><td colspan="7">timestamp</td></tr>
<tr><td colspan="7">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="5">0xABAC</td><td colspan="2">length=3</td></tr>
<tr><td colspan="7">NTP timestamp…</td></tr>
<tr><td colspan="7">...NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="2">mbz</td><td colspan="2">CSeq</td><td>padding</td></tr>
<tr><td colspan="7">payload…</td></tr>
</table>

The fields of this extension are as follows:

- NTP timestamp. An NTP [RFC 1305] timestamp indicating the absolute UTC time associated with the access unit.
- C: 1 bit. Indicates that this access unit is a synchronization point or “clean point”, e.g. the start of an intra-coded frame in the case of video streams.
- E: 1 bit. Indicates the end of a contiguous section of recording. The last access unit in each track before a recording gap, or at the end of available footage, shall have this bit set. When replaying in reverse, the E flag shall be set on the last frame at the end of the contiguous section of recording.
- D: 1 bit. Indicates that this access unit follows a discontinuity in transmission. It is primarily used during reverse replay; the first packet of each GOP has the D bit set since it does not chronologically follow the previous packet in the data stream (see section 21.5).
- mbz: This field is reserved for future use and must be zero.
- CSeq: 1 byte. This is the low-order byte of the CSeq value used in the RTSP PLAY command that was used to initiate transmission. When a client sends multiple, consecutive PLAY commands, this value may be used to determine where the data from each new PLAY command begins.

The replay header extension shall be present in the first packet of every access unit (e.g. video frame). It MAY NOT be present in subsequent packets of an access unit.

21.2.1 NTP Timestamps

The NTP timestamps in the RTP extension header shall increase monotonically over successive packets within a single RTP stream. They should correspond to wallclock time as measured at the original transmitter of the stream, adjusted if necessary to preserve monotonicity.

21.2.2 Compatibility with the JPEG header extension

The replay header extension may co-exist with the header extension used by the JPEG RTP profile; this is necessary to allow replay of JPEG streams that use this extension. The JPEG extension is simply appended to the replay extension; its presence is indicated by an RTP header extension length field with a value greater than 3, and by the extension start codes of 0xFFD8 or 0xFFFF at the start of the fourth word of the extension content.

The following illustrates a JPEG packet that uses both extensions:

Table 290: RTP packet with JPEG header layout

<table>
<tr><td>V=2</td><td>P</td><td>X=1</td><td>CC</td><td>M</td><td>PT</td><td>sequence number</td></tr>
<tr><td colspan="7">timestamp</td></tr>
<tr><td colspan="7">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="5">0xABAC</td><td colspan="2">length=N+3</td></tr>
<tr><td colspan="7">NTP timestamp…</td></tr>
<tr><td colspan="7">...NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="2">mbz</td><td>CSeq</td><td>padding</td></tr>
<tr><td colspan="5">0xFFD8</td><td colspan="2">jpeglength=N</td></tr>
<tr><td colspan="7">extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length</td></tr>
<tr><td colspan="7">payload…</td></tr>
</table>

21.3 RTSP Feature Tag

The Replay Service uses the “onvif-replay” feature tag to indicate that it supports the RTSP extensions described in this standard. This allows clients to query the server’s support for these extensions using the Require header as described in [RFC 2326] section 12.3.1 .

Example:

```
C->S:   SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0
        CSeq: 302
        Require: onvif-replay

S->C:   RTSP/1.0 551 Option not supported
        CSeq: 302
        Unsupported: onvif-replay
```

The Replay Server shall accept a SETUP command that includes a Require header containing the onvif-replay feature tag.

21.4 Initiating Playback

Playback is initiated by means of the RTSP PLAY method. For example:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
```

ONVIF devices MAY support reverse playback. Reverse playback is indicated using the Scale header field with a negative value. For example to play in reverse without no data loss a value of –1.0 would be used.

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Scale: -1.0
```

If a device supports reverse playback it shall accept a Scale header with a value of –1.0. A device MAY accept other values for the Scale parameter. Unless the Rate-Control header is set to “no” (see below), the Scale parameter is used in the manner described in [RFC 2326]. If Rate-Control is set to “no”, the Scale parameter, if it is present, shall be either 1.0 or –1.0, to indicate forward or reverse playback respectively. If it is not present, forward playback is assumed.

21.4.1 Range header field

The Range field shall be expressed using absolute times only; the other formats defined by [RFC 2326] shall NOT be used by ONVIF replay clients. Servers may choose to support other formats also. Absolute times are expressed using the *utc-range* from [RFC 2326].

Either open or closed ranges may be used. In the case of a closed range, the range is increasing (end time later than start time) for forward playback and decreasing for reverse playback. The direction of the range shall correspond to the value of the Scale header.

In all cases, the first point of the range indicates the starting point for replay.

Examples:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-20090615T115000
Rate-Control: no
```

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T115000.440Z-20090615T114900
Rate-Control: no
Scale: -1.0
```

21.4.2 Rate-Control header field

This specification introduces the Rate-Control header field, which may be either “yes” or “no”. If the field is not present, “yes” is assumed, and the stream is delivered in real time using standard RTP timing mechanisms. If this field is “no”, the stream is delivered as fast as possible, using only the flow control provided by the transport to limit the delivery rate.

The important difference between these two modes is that with “Rate-Control=yes”, the server is in control of the playback speed, whereas with “Rate-Control=no” the client is in control of the playback speed. Rate-controlled replay will typically only be used by non-ONVIF specific clients as they will not specify “Rate-Control=no”.

When replaying multiple tracks of a single recording, started by a single RTSP PLAY command and not using rate-control, the data from the tracks should be multiplexed in time in the same order as they were recorded.

21.4.3 Frames header field

The Frames header field may be used to reduce the number of frames that are transmitted, for example to lower bandwidth or processing load. Three modes are possible:

1. Intra frames only. This is indicated using the value “intra”, optionally followed by a minimum interval between successive intra frames in the stream. The latter can be used to limit the number of frames received even in the presence of “I-frame storms” caused by many receivers requesting frequent I-frames.
2. Intra frames and predicted frames only. This is indicated using the value “predicted”. This value can be used to eliminate B-frames if the stream includes them.
3. All frames. This is the default.

Examples:

To request intra frames only:

```
Frames: intra
```

To request intra frames with a minimum interval of 4000 milliseconds:

```
Frames: intra/4000
```

To request intra frames and predicted frames only:

```
Frames: predicted
```

To request all frames (note that it is not necessary to explicitly specify this mode but the example is included for completeness):

```
Frames: all
```

The interval argument used with the “intra” option refers to the recording timeline, not playback time; thus for any given interval the same frames are played regardless of playback speed. The interval argument shall NOT be present unless the Frames option is “intra”.

The server shall support the Frames header field. This does not preclude the use of the Scale header field as an alternative means of limiting the data rate. The implementation of the Scale header field may vary between different server implementations, as stated by [RFC 2326].

21.4.4 Synchronization points

The transmitted video stream shall begin at a synchronization point (see section 11.18). The rules for choosing the starting frame are as follows:

- If the requested start time is within a section of recorded footage, the stream starts with the first clean point at or before the requested start time. This is the case regardless of playback direction.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the forwards direction, the stream starts with the first clean point in the section following the requested start time.
- If the requested start time is within a gap in recorded footage and playback is being initiated in the reverse direction, the stream starts with the last clean point in the section preceding the requested start time.

21.5 Reverse replay

Reverse replay is initiated using the Scale header field with a negative value as described above.

21.5.1 Packet transmission order

The order in which video packets are transmitted during reverse replay is based on GOPs, where a GOP consists of a clean point followed by a sequence of non-cleanpoint packets.

During reverse playback, GOPs are sent in reverse order, but packets within a GOP are sent in forward order. The first packet of each GOP shall have the “discontinuity” bit set in its RTP extension header. The last packet of a GOP immediately following a gap (or the beginning of available footage) shall have the E bit set in its RTP extension header.

When transmitting only key frames, or when the codec is not motion-based (e.g. JPEG), a GOP is considered to consist of a single frame, but may still be composed of multiple packets. In this case the packets within each frame are again sent in forward order, while the frames themselves are sent in reverse order.

Audio and metadata streams MAY be transmitted in an order mirroring that of the video stream. Thus packets from these streams are sent in forward playback order until the occurrence of a packet (generally a video packet) with the D bit set in the extension header, at which point they jump back to a point before the discontinuity.

21.5.2 RTP sequence numbers

The RTP sequence numbers of packets transmitted during reverse playback shall increment monotonically *in the order of delivery*, not in the intended order of playback.

21.5.3 RTP timestamps

The use of RTP timestamps depends on the value of the Rate-Control header. If the value of this header is “no” (i.e. the client controls playback speed), the RTP timestamps are derived from the original sampling times of the recorded frames. If the Rate-Control header is not present or has the value “yes” (i.e. the server controls playback speed), the RTP timestamps correspond to playback timing as described in [RFC 2326] Appendix B.

If Rate-Control is “no”, the RTP timestamps of packets transmitted during reverse playback shall be the same as they would be if those same packets were being transmitted in the forwards direction. Unlike the sequence numbers, the RTP timestamps correspond to the original recording order, not the delivery order. The server MAY use the same RTP timestamps that were originally received when the stream was recorded.

This means that successive RTP packets of a single GOP will always have increasing RTP timestamps (see transmission order above), but that the timestamp on index frames of successively received GOPs will decrease during reverse replay.

If Rate-Control is “yes”, the RTP timestamps of packets transmitted during reverse playback shall indicate the times at which each frame should be rendered at the client. Thus successive packets of a single GOP will have *decreasing* RTP timestamps (since the first one delivered should be played last), and the timestamps on index frames will *increase*. In this mode the interval between successive timestamps depends on the values of the Speed and Scale headers, as described in [RFC 2326] Appendix B

21.6 RTSP Keepalive

When rate control is disabled and the RTP stream is tunneled through the RTSP connection (i.e. using the RTP/RTSP/TCP or RTP/RTSP/HTTP/TCP transports), the client shall not send SET_PARAMETER requests and the server shall not time out the connection in the absence of these requests. This is because the client may be unable to receive the responses to these requests, for example if replay is paused.

On the other hand, either the server or client may enable TCP keepalive on the connection in order to determine if the other endpoint has become unresponsive..

21.7 Currently recording footage

If the client commences playback from the current real world time or shortly before it, it can end up playing footage in real time as it is being recorded. In this event the server simply continues to send stream data to the client as it receives it.

Note that the E bit is not set on access units currently being recorded even though each access unit sent to the replay client will typically be the last one known to the server. If recording stops however, the E bit is set on the last access unit of the recording.

21.8 End of footage

If playback reaches a point after which there is no further data in one or more of the streams being sent, it stops transmitting data but does not enter the “paused” state. If the server resumes recording after this has happened, delivery will resume with the new data as it is received.

21.9 Go To Time

As stated in [RFC 2326] section 10.5, a PLAY command received when replay is already in progress will not take effect until the existing play operation has completed. This specification adds a new RTSP header, “Immediate”, which overrides this behaviour for the PLAY command that it is used with:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Immediate: yes
```

If the server receives a PLAY command with the Immediate header set to “yes”, it will immediately start playing from the new location, cancelling any existing PLAY command. The first packet sent from the new location shall have the D (discontinuity) bit set in its RTP extension header.

21.10 Use of RTCP

A server is not required to send RTCP packets. If it does send them, the following rules apply:

If Rate Control is enabled (see section 21.4.2), RTCP packets shall be constructed and transmitted as specified in [RFC 3550]. In particular, the NTP timestamp in a sender report indicates the current wallclock time, and is not related to the NTP timestamps embedded in the RTP extension headers in the data streams.

If Rate Control is not enabled, both the NTP timestamp and RTP timestamp in each sender report shall be set to zero.

21.11 Replay Service Commands

This section describes the web service commands provided by the Replay Service.

21.11.1 Request replay URI

GetReplayUri requests a URI that can be used to initiate playback of a recorded stream using RTSP as the control protocol. The URI is valid only as it is specified in the response. All implementations of the Replay Service shall support the GetReplayUri command.

Table 291: GetReplayUri command

GetReplayUri	Request-Response
Message name	**Description**
GetReplayUriRequest	*The **StreamSetup** element contains two parts. StreamType defines if a unicast or multicast media stream is requested. Transport specifies a chain of transport protocols defining the tunnelling of the media stream over different network protocols.* *The **RecordingToken** element indicates the recording to be streamed.* tt:StreamSetup **StreamSetup** [1][1] tt:ReferenceToken **RecordingToken** [1][1]
GetReplayUriResponse	*Contains the Uri to be used for requesting the media stream.* xs:anyURI **Uri** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	*The recording does not exist.*
env:Sender ter:InvalidArgVal ter:InvalidStreamSetup	*Specification of StreamType or Transport part in **StreamSetup** is not supported.*
env:Sender ter:OperationProhibited ter:StreamConflict	*Specification of StreamType or Transport part in **StreamSetup** causes conflict with other streams.*

21.11.2 ReplayConfiguration

The ReplayConfiguration structure contains the configuration of the replay service. The fields in the ReplayConfiguration structure are:

SessionTimeout: An RTSP session has a keep-alive time. It shall be refreshed regularly to prevent the session from timing out. If the session times out, it shall be torn down. The session timeout for replay follows the same rules as applies for live streaming using the media service and as discussed in chapter “Real-time streaming”.

21.11.3 SetReplayConfiguration

SetReplayConfiguration changes the configuration of the replay service. The replay service shall allow its configuration to be changed using this command.

Table 292: SetReplayConfiguration command

SetReplayConfiguration	Request-Response
Message name	**Description**
SetReplayConfigurationRequest	*The **Configuration** shall hold the new configuration for the replay service.* tt:ReplayConfiguration **Configuration**[1][1]
SetReplayConfigurationResponse	*This shall be the empty message*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:ConfigModify	*The values in the configuration cannot be set.*

21.11.4 GetReplayConfiguration

GetReplayConfiguration returns the current configuration of the replay service. The replay service shall allow its configuration to be retrieved using this command.

Table 293: GetReplayConfiguration command

GetReplayConfiguration	Request-Response
Message name	**Description**
GetReplayConfigurationRequest	*This shall be an empty message.*
GetReplayConfigurationResponse	*The **Configuration** shall holds the current configuration for the replay service.* tt:ReplayConfiguration **Configuration**[1][1]
Fault codes	**Description**
	No command specific error codes.

21.11.5 Service specific fault codes

Table 90 lists the replay service-specific fault codes. In addition, each command can also generate a generic fault, see ***[Table 5]***.

The specific faults are defined as sub code of a generic fault, see Section ***[Section 5.11.2.1]***. The parent generic subcode is the *subcode* at the top of each row below and the specific fault *subcode* is at the bottom of the cell.

Table 294: Replay service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Sender	ter:InvalidArgVal ter:NoProfile	Profile token does not exist	The requested profile token **ProfileToken** does not exist.
env:Sender	ter:InvalidArgVal ter:InvalidStreamSetup	Invalid Stream setup	Specification of StreamType or Transport part in **StreamSetup** is not supported.
env:Sender	ter:OperationProhibited ter:StreamConflict	Stream conflict	Specification of StreamType or Transport part in **StreamSetup** causes conflict with other streams.
env:Sender	ter:InvalidArgVal ter:ConfigModify	Parameters cannot be set	The configuration parameters cannot be set.

22 Security

As is true for all network-oriented information technology, security is a most important subject for network video communication. The security threat depends on the application. While some applications are most vulnerable to network based attacks, other applications are not at all sensitive. The cost for implementing security countermeasures varies depending on the type of attacks intended to prevent. These facts imply that we cannot list general security requirements on the network video product or system, but can try to find a reasonable level of security requirements for devices conformant to this specification and to define basic security mechanism that allows building secure network video systems.

The current specification defines security mechanisms on two communication levels:

- Transport level security
- Message level security

This specification adopts port-based authentication mechanism as follows.

- IEEE 802.1X

22.1 Transport level security

Transport *level* security protects the data transfer between the client and the server. Transport Layer Security (TLS) is regarded as a mature standard for encrypted transport connections to provide a basic level of communication security. The TLS protocol allows the configuration of a mutually authenticated transport session as well as preserving the confidentiality and the integrity protected transport.

A device conformant to this specification should support TLS 1.0 [RFC 2246] and related specifications. The device should support TLS 1.1 [RFC 4346]. The device MAY support TLS 1.2 [RFC 5246].

A device should support TLS for protection of all of the ONVIF services it provides. A device also should support TLS for protection of media streams for the RTP/RTSP/HTTPS tunnel option as defined in Section 11. This specification profiles a particular implementation of TLS and other relevant specifications that can be used with TLS.

A client should support TLS 1.0 [RFC 2246] and TLS 1.1 [RFC 4346]. The client MAY support TLS 1.2 [RFC 5246].

22.1.1 Supported cipher suites

A device that supports TLS shall support all of the following cipher suites [RFC 2246], [RFC 3268]:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

If a client supports TLS, then it shall support the following cipher suites:

- TLS_RSA_WITH_AES_128_CBC_SHA

- TLS_RSA_WITH_NULL_SHA

22.1.2 Server authentication

A device that supports TLS shall support server authentication using TLS. The device shall support processing of X.509 server certificates. The RSA key length shall be at least 1024 bits.

A client should support server authentication using TLS.

This specification does not provide a full server certificate generation and Certificate Authority (CA) model. However, device management commands for device certificate retrieval and download are defined in Section 8.4.

The details of the server private key or keys secure bootstrapping mechanisms are *outside the scope* of the current specification. However, commands for *on board* key generation are defined in Section 8.4.

22.1.3 Client authentication

A device that supports TLS should support client authentication. Client authentication can be enabled/disabled with a device management command as described in Section 8.4

A device that supports TLS shall include the RSA certificate type (rsa_sign, for example) in the certificate request [RFC 2246] for client certificates, and shall support verification of the RSA client certificate and signature.

A client should support client authentication. If client authentication is supported, the client shall support RSA client certificate and signature and shall use an RSA key length of at least 1024 bits.

The trusted CA bootstrapping mechanisms are *outside the scope* of the current specification. Future versions of the specification might define standardized bootstrapping mechanisms.

22.2 Message level security

TLS allows point-to-point confidentiality and integrity. Web Services, however, allow a more flexible communication pattern with intermediate nodes. In such situations TLS cannot provide end-to-end security. Furthermore, in order to implement user based access control on command level for Web Services, there is a need to verify the origin of each SOAP message. This can be provided through the WS-Security framework. ONVIF WS-security is profiled in Section 5.12.

22.3 IEEE 802.1X

IEEE 802.1X is an IEEE standard for port based network access control for the purpose of providing authentication and authorization of the devices attached to LAN ports. It makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases in which the authentication and authorization process fails.

This specification recommends the adoption of IEEE 802.1X for port based authentication for wireless networks. A device that supports IEEE 802.1X shall support EAP-PEAP/MSCHAPv2 type as a supported EAP method. The device MAY also support other EAP methods such as EAP-MD5, EAP-TLS and EAP-TTLS types.

This specification defines a set of commands to configure and manage the IEEE 802.1X configuration, please refer to section 8.4.7.

Annex A
(informative)

Notification topics

A.1 Media configuration topics

For the following entities of the Media Configuration, the ONVIF TopicNamespace provides the following topics:

```
tns1:MediaConfiguration/Profile
tns1:MediaConfiguration/VideoSourceConfiguration
tns1:MediaConfiguration/AudioSourceConfiguration
tns1:MediaConfiguration/VideoEncoderConfiguration
tns1:MediaConfiguration/AudioEncoderConfiguration
tns1:MediaConfiguration/VideoAnalyticsConfiguration
tns1:MediaConfiguration/PTZConfiguration
tns1:MediaConfiguration/MetaDataConfiguration
```

Each of these topics represents a property. A client subscribing to one of these topics will be notified about changes, creation and deletion of the corresponding entity.

The Message structures of the different topics are specified next using the MessageDescription Language introduced in Section 15.

A.1.1 Profile

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="ProfileToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:Profile"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.2 VideoSourceConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="VideoSourceConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:VideoSourceConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.3 AudioSourceConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="AudioSourceConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:AudioSourceConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.4 VideoEncoderConfiguration

```
<tt:MessageDescription iIsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="VideoEncoderConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:VideoEncoderConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.5 AudioEncoderConfiguration

```
<tt:MessageDescription iIsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="AudioEncoderConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:AudioEncoderConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.6 VideoAnalyticsConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="VideoAnalyticsConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:VideoAnalyticsConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.7 PTZConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="PTZConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:PTZConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.8 MetaDataConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="MetaDataConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Config"
                 Type="tt:MetaDataConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.9 Device management topics

The Device Topic contains the following Sub-topics defined in the ONVIF TopicNamespace:

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

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

Only the Relay defines a message payload. The other topics reply with an empty message.

A.1.10 Relay

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

A.1.11 PTZ Controller Topics

The PTZ service specifies handling of PTZ presets. Since the move operations are non-blocking, an NVC is not informed when the PTZ preset has been reached. Therefore, the following events are introduced which inform subscribers about the status of preset movements.

```
tns1:PTZController/PTZPresets/Invoked
tns1:PTZController/PTZPresets/Reached
tns1:PTZController/PTZPresets/Aborted
tns1:PTZController/PTZPresets/Left
```

The typical sequence of events is that first a NVC requests a certain Preset. When the device accepts this request, it will send out an Invoked event. The Invoked event has to follow either a Reached event or an Aborted event. The former is used when dome was able to reach the invoked preset position, the latter in any other case. A Reached event has to follow a Left event, as soon as the dome moves away from the preset position.

The Message structure of these events is given by the following Message Description (see chapter 12):

```
<tt:MessageDescription>
  <tt:Source>
    <tt:SimpleItemDescription Name="PTZConfigurationToken"
                  Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="PresetToken" Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="PresetName"  Type="tt:Name"/>
  </tt:Data>
</tt:MessageDescription>
```

Annex B
(informative)

Scene descriptions

B.1 Colour Descriptor

A Colour Descriptor is defined as an optional element of the Appearance Node of an Object Node. The Colour Descriptor is defined by a list of Colour Clusters, each consisting of a Colour Value, an optional weight and an optional covariance matrix. The Colour Descriptor does not specify, how the Colour Clusters are created. They can represent bins of a colour histogram or the result of a clustering algorithm.

Colours are represented by three-dimensional vectors. Additionally, the colourspace of each colour vector can be specified by a colourspace attribute. If the colourspace attribute is missing, the YCbCr colourspace is assumed. It refers to the 'sRGB' gamut with the RGB to YCbCr transformation as of ISO/IEC 10918-1 (Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines), a.k.a. JPEG. The Colourspace URI for the YCbCr colourspace is www.onvif.org/ver10/colorspace/YCbCr.

```
<xs:complexType name="ColorDescriptor">
  <xs:sequence>
    <xs:element name="ColorCluster" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Color" type="tt:Color"/>
          <xs:element name="Weight" type="xs:float" minOccurs="0"/>
          <xs:element name="Covariance" type="tt:ColorCovariance" minOccurs="0"/>
          ...
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Color">
  <xs:attribute name="X" type="xs:float" use="required"/>
  <xs:attribute name="Y" type="xs:float" use="required"/>
  <xs:attribute name="Z" type="xs:float" use="required" />
  <xs:attribute name="Colorspace" type="xs:anyURI"/>
</xs:complexType>

<xs:complexType name="ColorCovariance">
  <xs:attribute name="XX" type="xs:float" use="required"/>
  <xs:attribute name="YY" type="xs:float" use="required"/>
  <xs:attribute name="ZZ" type="xs:float" use="required" />
  <xs:attribute name="XY" type="xs:float"/>
  <xs:attribute name="XZ" type="xs:float"/>
  <xs:attribute name="YZ" type="xs:float" />
  <xs:attribute name="Colorspace" type="xs:anyURI"/>
</xs:complexType>
```

B.1.1 Class Descriptor

A Class Descriptor is defined as an optional element of the Appearance Node of an Object Node. The Class Descriptor is defined by a list of object classes together with a likelihood that the corresponding object belongs to this class. The sum of the likelihoods shall NOT exceed 1.

```
<xs:simpleType name="ClassType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Animal"/>
    <xs:enumeration value="Face"/>
    <xs:enumeration value="Human"/>
    <xs:enumeration value="Vehicle"/>
```

```
      <xs:enumeration value="Other"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="ClassDescriptor">
    <xs:sequence>
      <xs:element name="ClassCandidate" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
           <xs:sequence>
             <xs:element name="Type" type="tt:ClassType"/>
             <xs:element name="Likelihood" type="xs:float"/>
           </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```

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]

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* [URL: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 [URL: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.

URL: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.

URL: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.

URL: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.

URL: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

URL: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 Analytics WSDL] ONVIF Video Analytics Service WSDL, ver 2.0, 2010.

URL:http://www.onvif.org/onvif/ver10/analytics/wsdl/analytics.wsdl

[ONVIF DM WSDL] ONVIF Device Management Service WSDL, ver 2.0, 2010.

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

[ONVIF Event WSDL] ONVIF Event Service WSDL, ver 2.0, 2010.

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

[ONVIF Imaging WSDL] ONVIF Imaging Service WSDL, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/imaging/wsdl/imaging.wsdl
[ONVIF Media WSDL] ONVIF Media Service WSDL, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/media/wsdl/media.wsdl
[ONVIF PTZ WSDL] ONVIF PTZ Service WSDL, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/ptz/wsdl/ptz.wsdl
[ONVIF DP WSDL] ONVIF Remote Discovery Proxy Services WSDL, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/network/wsdl/remotediscovery.wsdl
[ONVIF Schema] ONVIF Schema, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/schema/onvif.xsd
[ONVIF Topic Namespace] ONVIF Topic Namespace XML, ver 2.0, 2010.
URL:http://www.onvif.org/onvif/ver10/topics/topicns.xml
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>