Open Network Video Interface Forum Core Specification

Version 1.0
November, 2008

Table of Contents

1 Scope 13

2 References 14

2.1 Normative references 14

2.2 Informative references 17

3 Terms and Definitions 19

3.1 Conventions 19

3.2 Definitions....19

3.3 Abbreviations 19

4 Specification overview 22

4.1 Web Services 22

4.2 IP configuration 23

4.3 Device discovery 23

4.4 Device management 24
4.4.1 Capabilities 24
4.4.2 Network....25
4.4.3 System 25
4.4.4 Security....26
4.4.5 Input/Output 26

4.5 Imaging configuration....26

4.6 Media configuration....26
4.6.1 Media profiles....27

4.7 Real-time streaming 30

4.8 Event handling 31

4.9 PTZ control....31

4.10 Video analytics 32

4.11 Security 34

4.12 Client code examples....34
4.12.1 Requesting Stream URI 34
4.12.2 Media Setup 35

5 Web Services frame work 36

5.1 Services overview 36
5.1.1 Device service....37
5.1.2 Media service....37
5.1.3 Event service 37
5.1.4 Imaging service....37
5.1.5 PTZ service....37
5.1.6 Analytics service 37
5.1.7 Storage service....37
5.1.8 Services requirements....38

5.2 WSDL overview....38

5.3 Namespaces 39

5.4 Types ... 40
5.5 Messages ... 40
5.6 Operations ... 41
5.6.1 One-way operation type ... 42
5.6.2 Request-response operation type ... 43
5.7 Port Types ... 43
5.8 Binding ... 44
5.9 Ports ... 44
5.10 Services ... 44
5.11 Error handling ... 44
5.11.1 Protocol errors ... 44
5.11.2 SOAP errors ... 45
5.12 Security ... 48
5.12.1 User-based access control ... 49
5.12.2 User token profile ... 49

6 IP configuration 50

7 Device discovery 51

7.1 General ... 51
7.2 Modes of operation ... 51
7.3 Discovery definitions ... 52
7.3.1 Endpoint reference ... 52
7.3.2 Service addresses ... 52
7.3.3 Hello ... 52
7.3.4 Probe and Probe Match ... 54
7.3.5 Resolve and Resolve Match ... 54
7.3.6 Bye ... 54
7.3.7 SOAP Fault Messages ... 54
7.4 Remote discovery extensions ... 55
7.4.1 Network scenarios ... 55
7.4.2 Discover proxy ... 57
7.4.3 Remote Hello and Probe behaviour ... 58
7.4.4 Client behaviour ... 59
7.4.5 Security ... 60

8 Device management 62

8.1 Capabilities ... 62
8.1.1 Get WSDL URL ... 62
8.1.2 Capability exchange ... 62
8.2 Network ... 66
8.2.1 Get hostname ... 66
8.2.2 Set hostname ... 66
8.2.3 Get DNS settings ... 66
8.2.4 Set DNS settings ... 67
8.2.5 Get NTP settings ... 68
8.2.6 Set NTP settings ... 68
8.2.7 Get dynamic DNS settings ... 69
8.2.8 Set dynamic DNS settings ... 69
8.2.9 Get network interface configuration ... 70
8.2.10 Set network interface configuration ... 71

8.2.11 Get network protocols 72
8.2.12 Set network protocols 72
8.2.13 Get default gateway 73
8.2.14 Set default gateway 73
8.2.15 Get zero configuration 73
8.2.16 Set zero configuration 74
8.2.17 Get IP address filter 74
8.2.18 Set IP address filter 75
8.2.19 Add an IP filter address 75
8.2.20 Remove an IP filter address 76

8.3 System 77
8.3.1 Device Information 77
8.3.2 Backup 77
8.3.3 Restore 78
8.3.4 Get system date and time 78
8.3.5 Set system date and time 79
8.3.6 Factory default 79
8.3.7 Firmware upgrade 80
8.3.8 Get system logs 81
8.3.9 Get support information 81
8.3.10 Reboot 82
8.3.11 Get scope parameters 82
8.3.12 Set scope parameters 83
8.3.13 Add scope parameters 83
8.3.14 Remove scope parameters 84
8.3.15 Get discovery mode 84
8.3.16 Set discovery mode 85
8.3.17 Get remote discovery mode 85
8.3.18 Set remote discovery mode 86
8.3.19 Get remote DP addresses 86
8.3.20 Set remote DP addresses 87

8.4 Security 87
8.4.1 Get access policy 87
8.4.2 Set access policy 88
8.4.3 Get users 88
8.4.4 Create users 89
8.4.5 Delete users 90
8.4.6 Set users settings 90
8.4.7 Create self-signed certificate 91
8.4.8 Get certificates 91
8.4.9 Get certificate status 92
8.4.10 Set certificate status 92
8.4.11 Get certificate request 92
8.4.12 Get client certificate status 93
8.4.13 Set client certificate status 93
8.4.14 Load NVT certificate 94

8.5 Input/Output (I/O) 94
8.5.1 Get relay outputs 94
8.5.2 Set relay output settings 95
8.5.3 Trigger relay output 96

8.6 Service specific fault codes 96

9 Imaging configuration 101

9.1 Imaging settings 101
9.1.1 Get imaging settings 103

9.1.2 Set imaging settings 103
9.1.3 Get options....................104
9.1.4 Move 104
9.1.5 Get move options....................105
9.1.6 Stop....................106
9.1.7 Get imaging status....................106

9.2 Service specific fault codes....................107

10 Media configuration 108

10.1 Audio and video codecs....................108

10.2 Media Profile 109
10.2.1 Create media profile 109
10.2.2 Get media profiles 109
10.2.3 Get media profile 110
10.2.4 Add video source configuration to a profile 110
10.2.5 Add video encoder configuration to a profile....................111
10.2.6 Add audio source configuration to a profile....................112
10.2.7 Add audio encoder configuration to a profile....................112
10.2.8 Add PTZ configuration to a profile....................113
10.2.9 Add video analytics configuration to a profile....................114
10.2.10 Add metadata configuration to a profile 114
10.2.11 Remove video source configuration from a profile....................115
10.2.12 Remove video encoder configuration from a profile....................116
10.2.13 Remove audio source configuration from a profile....................116
10.2.14 Remove audio encoder configuration from a profile 117
10.2.15 Remove PTZ configuration from a profile 118
10.2.16 Remove video analytics configuration from a profile....................118
10.2.17 Remove metadata configuration from a profile 119
10.2.18 Delete media profile....................119

10.3 Video source 120
10.3.1 GetVideoSources 120

10.4 Video source configuration....................120
10.4.1 Get video source configurations....................121
10.4.2 Get video source configuration....................121
10.4.3 Get compatible video source configurations 122
10.4.4 Get video source configuration options....................122
10.4.5 Modify a video source configuration....................123

10.5 Video encoder configuration 124
10.5.1 Get video encoder configurations....................124
10.5.2 Get video encoder configuration 124
10.5.3 Get compatible video encoder configurations 125
10.5.4 Get video encoder configuration options....................125
10.5.5 Modify a video encoder configuration....................126

10.6 Audio source 127
10.6.1 Get audio sources 127

10.7 Audio source configuration....................128
10.7.1 Get audio source configurations....................128
10.7.2 Get audio source configuration 128
10.7.3 Get compatible audio source configurations 129
10.7.4 Get audio source configuration options....................130
10.7.5 Modify an audio source configuration....................130

10.8 Audio encoder configuration 131
10.8.1 Get audio encoder configurations 131

10.8.2 Get audio encoder configuration 132
10.8.3 Get compatible audio encoder configurations 133
10.8.4 Get audio encoder configuration options 133
10.8.5 Modify audio encoder configurations 134

10.9 Video analytics configuration 135
10.9.1 Get video analytics configurations 135
10.9.2 Get video analytics configuration 136
10.9.3 Get compatible video analytics configurations 136
10.9.4 Modify a video analytics configuration 137

10.10 Metadata configuration 138
10.10.1 Get metadata configurations 138
10.10.2 Get metadata configuration 138
10.10.3 Get compatible metadata configurations 139
10.10.4 Get metadata configuration options 139
10.10.5 Modify a metadata configuration 140

10.11 Stream URI 141
10.11.1 Request stream URI 141

10.12 Snapshot 142
10.12.1 Request snapshot URI 142

10.13 Multicast 142
10.13.1 Start multicast streaming 142
10.13.2 Stop multicast streaming 143

10.14 Synchronization Points 143
10.14.1 Set synchronization point 143

10.15 Service specific fault codes 144

11 Real time streaming 146

11.1 Media stream protocol 146
11.1.1 Transport format 146
11.1.2 Media Transport 147
11.1.3 JPEG over RTP 151

11.2 Media control protocol 155
11.2.1 Stream control 155

11.3 Error Handling 158

12 Event handling 159

12.1 Basic Notification Interface 159
12.1.1 Introduction 159
12.1.2 Requirements 160

12.2 Real-time Pull-Point Notification Interface 161
12.2.1 Create pull point subscription 163
12.2.2 Pull messages 163

12.3 Notification Streaming Interface 164

12.4 Properties 164
12.4.1 Property Example 164

12.5 Notification Structure 165
12.5.1 Notification information 165
12.5.2 Message Format 166
12.5.3 Property example, continued 167
12.5.4 Message Description Language 169

12.5.5 Message Content Filter 170

12.6 Synchronization Point 171

12.7 Topic Structure 172
12.7.1 ONVIF Topic Namespace 172
12.7.2 Topic Type Information 172
12.7.3 Topic Filter 173

12.8 Get event properties 174

12.9 SOAP Fault Messages 175

12.10 Notification example 175
12.10.1 GetEventPropertiesRequest 175
12.10.2 GetEventPropertiesResponse 176
12.10.3 CreatePullPointSubscription 177
12.10.4 CreatePullPointSubscriptionResponse 177
12.10.5 PullMessagesRequest 178
12.10.6 PullMessagesResponse 178
12.10.7 UnsubscribeRequest 179
12.10.8 UnsubscribeResponse 180

12.11 Service specific fault codes 180

13 PTZ control 181

13.1 PTZ Model 182

13.2 PTZ Node 183
13.2.1 GetNodes 183
13.2.2 GetNode 184

13.3 PTZ Configuration 184
13.3.1 GetConfigurations 186
13.3.2 GetConfiguration 186
13.3.3 GetConfigurationOptions 186
13.3.4 SetConfiguration 187

13.4 Move Operations 188
13.4.1 AbsoluteMove 188
13.4.2 RelativeMove 189
13.4.3 ContinuousMove 190
13.4.4 Stop 191
13.4.5 GetStatus 192

13.5 Preset operations 192
13.5.1 SetPreset 193
13.5.2 GetPresets 194
13.5.3 GotoPreset 194
13.5.4 RemovePreset 195

13.6 Home Position operations 195
13.6.1 GotoHomePosition 196
13.6.2 SetHomePosition 196

13.7 Auxiliary operations 197
13.7.1 SendAuxiliaryCommand 197

13.8 Predefined PTZ spaces 197
13.8.1 Absolute Position Spaces 198
13.8.2 Relative Translation Spaces 198
13.8.3 Continuous Velocity Spaces 199
13.8.4 Speed Spaces 200

13.9 Service specific fault codes....200

14 Video analytics 203

14.1 Scene Description Interface....203
14.1.1 Overview....203
14.1.2 Frame Related Content....203
14.1.3 Scene Elements....206

14.2 Rule interface....211
14.2.1 Rule representation....211
14.2.2 Rule description language....212
14.2.3 Standard Rules....213
14.2.4 Operations on rules....214

14.3 Analytics Modules Interface....217
14.3.1 Analytics module configuration....217
14.3.2 Analytics Module Description Language....217
14.3.3 Operations on Analytics Modules....218

14.4 Service-specific fault codes....221

15 Security 223

15.1 Transport level security....223
15.1.1 Supported cipher suites....223
15.1.2 Server authentication....224
15.1.3 Client authentication....224

15.2 Message level security....224

A Notification topics (informative) 225

A.1 Media configuration topics....225
A.1.1 Profile....225
A.1.2 VideoSourceConfiguration....225
A.1.3 AudioSourceConfiguration....225
A.1.4 VideoEncoderConfiguration....226
A.1.5 AudioEncoderConfiguration....226
A.1.6 VideoAnalyticsConfiguration....226
A.1.7 PTZConfiguration....226
A.1.8 MetaDataConfiguration....226
A.1.9 Device management topics....227
A.1.10 Relay....227
A.1.11 PTZ Controller Topics....227

B Scene descriptions (informative) 228

B.1 Colour Descriptor....228
B.1.1 Class Descriptor....228

Contributors

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

Introduction

The goal of the Open Network Video Interface Forum Core (ONVIF) specification is to realize a fully interoperable network video implementation comprised of products from different network video vendors. The ONVIF specification describes the ONVIF network video model, interfaces, data types and data exchange patterns. The specification reuses existing relevant standards where available, and introduces new specifications only where necessary to support the specific requirements of ONVIF.

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 Device Service WSDL [ONVIF DM 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 Remote Discovery WSDL [ONVIF DP 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:

Scope: Defines the scope of the ONVIF 1.0 specification.

Normative References

Terms and Definitions: Defines the terminology used in the specification

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 network video transmitters are discovered in local and remote networks.

Device Management: Defines the network video transmitter management commands.

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.

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

1 Scope

The ONVIF specifications define standardized 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 video transmitters 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 and video analytics.

The management and control interfaces defined in this specification are described as Web Services. The ONVIF specifications 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 specification defines procedures for device discovery. The device discovery mechanisms in the specification are based on the WS-Discovery [WS-Discovery] specification with extensions. These extensions have been introduced in order to cover the specific network video discovery needs.

The specification 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 References

2.1 Normative references

[ISO 14496-2]	ISO/IEC 14496-2:2004 Information technology -- Coding of audio-visual objects -- Part 2: Visual URL:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39259
[ISO 14496-3]	ISO/IEC 14496-3:2005 Information technology -- Coding of audio-visual objects -- Part 3: Audio URL:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42739
[ISO 14496-10]	ISO/IEC 14496-10:2008 Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding URL:http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50726
[ITU-T G.711]	ITU-T G.711 Pulse code modulation (PCM) of voice frequencies URL: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]	ITU-T G.726, 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM) URL:http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.726-199012-I!!PDF-E&type=items
[MTOM]	SOAP Message Transmission Optimization Mechanism URL:http://www.w3.org/TR/soap12-mtom/
[ONVIF Analytics WSDL]	ONVIF Video Analytics Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/analytics/wsdl/analytics.wsdl
[ONVIF DM WSDL]	ONVIF Device Management Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/device/wsdl/devicemgmt.wsdl
[ONVIF Event WSDL]	ONVIF Event Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/event/wsdl/event.wsdl
[ONVIF Imaging WSDL]	ONVIF Imaging Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/imaging/wsdl/imaging.wsdl
[ONVIF Media WSDL]	ONVIF Media Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/media/wsdl/media.wsdl
[ONVIF PTZ WSDL]	ONVIF PTZ Service WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/ptz/wsdl/ptz.wsdl
[ONVIF DP WSDL]	ONVIF Remote Discovery Proxy Services WSDL, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/network/wsdl/remotediscovery.wsdl
[ONVIF Schema]	ONVIF Schema, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/schema/onvif.xsd
[ONVIF Topic Namespace]	ONVIF Topic Namespace XML, ver 1.0, 2008. URL:http://www.onvif.org/onvif/ver10/topics/topicns.xml
[PKCS#10]	PKCS #10 v1.7: Certification Request Syntax Standard, RSA Laboratories, May 2000. URL:ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf
[RFC 2119]	“Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner, March 1997.

URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC 2131] “Dynamic Host Configuration Protocol”, R. Droms, March 1997.
URL:http://www.ietf.org/rfc/rfc2131.txt

[RFC 2136] “Dynamic Updates in the Domain Name System (DNS UPDATE)”, P. Vixie et al., April 1997.
URL:http://www.ietf.org/rfc/rfc2136.txt

[RFC 2246] “The TLS Protocol Version 1.0”, T. Dierks and C. Allen, January 1999.
URL:http://www.ietf.org/rfc/rfc2246.txt

[RFC 2326] “Real Time Streaming Protocol (RTSP)”, H. Schulzrinne, A. Rao and R. Lanphier, April 1998.
URL:http://www.ietf.org/rfc/rfc2326.txt

[RFC 2435] “RFC2435 - RTP Payload Format for JPEG-compressed Video”, L. Berc et al., October 1998.
URL:http://www.ietf.org/rfc/rfc2435.txt

[RFC 2616] “Hypertext Transfer Protocol -- HTTP/1.1”, R. Fielding et al., June 1999.
URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC 2782] “A DNS RR for specifying the location of services (DNS SRV)”, A. Gulbrandsen, P. Vixie and L. Esibov, February 2000.
URL:http://www.ietf.org/rfc/rfc2782.txt

[RFC 2818] “HTTP over TLS”, E. Rescorla, May 2000.
URL:http://www.ietf.org/rfc/rfc2818.txt

[RFC 3268] “Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS)”, P. Chown, June 2002.
URL:http://www.ietf.org/rfc/rfc3268.txt

[RFC 3315] “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, R. Droms et al., July 2003.
URL:http://www.ietf.org/rfc/rfc3315.txt

[RFC 3550] “RTP: A Transport Protocol for Real-Time Applications”, H. Schulzrinne et al., July 2003.
URL:http://www.ietf.org/rfc/rfc3550.txt

[RFC 3551] “RTP Profile for Audio and Video Conferences with Minimal Control”, H. Schulzrinne and S. Casner, July 2003.
URL:http://www.ietf.org/rfc/rfc3551.txt

[RFC 3927] “Dynamic Configuration of IPv4 Link-Local Addresses”, S. Cheshire, B. Aboba and E. Guttman, May 2005.
URL:http://www.ietf.org/rfc/rfc3927.txt

[RFC 3984] “RTP Payload Format for H.264 Video”, S. Wenger et al., February 2005.
URL:http://www.ietf.org/rfc/rfc3984

[RFC 3986] “Uniform Resource Identifier (URI): Generic Syntax”, T. Berners-Lee et al., January 2005.
URL:http://www.ietf.org/rfc/rfc3986.txt

[RFC 4122] “A Universally Unique IDentifier (UUID) URN Namespace”, P. Leach, M. Mealling and R. Salz, July 2005.
URL:http://www.ietf.org/rfc/rfc4122.txt

[RFC 4346] “The Transport Layer Security (TLS) Protocol Version 1.1”, T. Dierks and E. E. Rescorla, April 2006.
URL:http://www.ietf.org/rfc/rfc4346.txt

[RFC 4566] “SDP: Session Description Protocol”, M. Handley, V. Jacobson and C. Perkins, July 2006.

URL:http://www.ietf.org/rfc/rfc4566.txt

[RFC 4571] "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", J. Lazzaro, July 2006.

URL:http://www.ietf.org/rfc/rfc4571.txt

[RFC 4585] "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", J. Ott et al., July 2006.

URL:http://www.ietf.org/rfc/rfc4585.txt

[RFC 4702] "The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option", M. Stapp, B. Volz and Y. Rekhter, October 2006.

URL:http://www.ietf.org/rfc/rfc4702.txt

[RFC 4861] "Neighbor Discovery for IP version 6 (IPv6)", T. Narten et al., September 2007.

URL:http://www.ietf.org/rfc/rfc4861.txt

[RFC 4862] "IPv6 Stateless Address Auto configuration", S. Thomson, D. Narten and T. Jinmei, September 2007.

URL:http://www.ietf.org/rfc/rfc4862.txt

[RFC 5104] "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", S. Wenger et al., February 2008.

URL:http://www.ietf.org/rfc/rfc5104.txt

[RFC 5246] "The Transport Layer Security (TLS) Protocol Version 1.2", T. Dierks and E. E. Rescorla, August 2008.

URL:http://www.ietf.org/rfc/rfc5246.txt

[SOAP 1.2, Part 1] "SOAP Version 1.2 Part 1: Messaging Framework", M. Gudgin (Ed) et al., April 2007.

URL:http://www.w3.org/TR/soap12-part1/

[SOAP 1.2, Part 2] "SOAP Version 1.2 Part 2: Adjuncts (Second Edition)", M. Gudgin (Ed) et al., April 2007.

URL:http://www.w3.org/TR/2007/REC-soap12-part2-20070427/

[WS-Addressing] "Web Services Addressing 1.0 – Core", M. Gudgin (Ed), M. Hadley (Ed) and T. Rogers (Ed), May 2006.

URL:http://www.w3.org/TR/ws-addr-core/

[WS-BaseNotification] "Web Services Base Notification 1.3", OASIS Standard, October 2006

URL:http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

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

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

[WS-Discovery] "Web Services Dynamic Discovery (WS-Discovery)", J. Beatty et al., April 2005.

URL:http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf

[WS-Security] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard, February 2006.

URL:http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WS-Topics] "Web Services Topics 1.3", OASIS Standard, 1 October 2006.

URL:http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

[WS-UsernameToken] "Web Services Security UsernameToken Profile 1.0", OASIS Standard, March 2004.

URL:http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

[WSDL1.1] "Web Services Description Language (WSDL) 1.1", E. Christensen et al, March 2001.

	URL:http://www.w3.org/TR/wsdl
[XML-Schema, Part 1]	"XML Schema Part 1: Structures Second Edition", H. S. Thompson (Ed) et al., October 2004. URL:http://www.w3.org/TR/xmlschema-1/
[XML-Schema, Part 2]	"XML Schema Part 2: Datatypes Second Edition", P. V. Biron (ed) et al., October 2004. URL:http://www.w3.org/TR/xmlschema-2/
[XOP]	XML-binary Optimized Packaging URL:http://www.w3.org/TR/2005/REC-xop10-20050125/

2.2 Informative references

[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).

3 Terms and Definitions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119].

3.2 Definitions

Capability	The capability commands allow an NVC to ask for the services provided by an NVT.
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.
Input/Output (I/O)	Currently only relay ports are handled.
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 to a video or an audio encoder, PTZ and analytics configurations.
Metadata	All streaming data except video and audio, including video analytics results, PTZ position data and other functions.
Network Video Client (NVC)	Network video receiver or controller device communicating with an NVT over an IP network.
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 an NVC.
Optical Zoom	Changes the focal length (angle of view) for the NVT by moving the zoom lens in the camera’s optics.
PKCS	Refers to a group of Public Key Cryptography Standards devised and published by RSA Security.
PTZ Node	Low-level PTZ entity that maps to the PTZ device and its capabilities.
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.
Video Analytics	Algorithms or programs used to analyze video data and to generate data describing object location and behaviour.

3.3 Abbreviations

AAC	Advanced Audio Coding
ASN	Abstract Syntax Notation
AVP	Audio/Video Profile
AVPF	Audio/Video Profile for rtcp Feedback
BLC	Back Light Compensation

CBC	Cipher-Block Chaining
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
NVC	Network Video Client
NVT	Network Video Transmitter
OASIS	Organization for the Advancement of Structured Information Standards
ONVIF	Open Network Video Interface Forum
POSIX	Portable Operating System Interface
PTZ	Pan/Tilt/Zoom
REL	Rights Expression Language
RSA	Rivest ,Sharmir and Adleman
RTCP	RTP Control Protocol
RTP	Real Time 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
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
UDDI	Universal Discovery, Description and Integration Registry
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URN	Uniform Resource Name
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
WDR	Wide Dynamic Range

WS	Web Services
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
XML	eXtensible Markup Language

4 Specification overview

The ONVIF core specifications are based on network video use cases covering both local and wide area network scenarios. The ONVIF specification framework covers procedures from the network video transmitter deployment and the configuration phase to the real time streaming phase for these different network scenarios. The framework starts from a core set of interface functions, and it shall be easy to extend and enhance the specifications as future versions are released.

The main focus of the specification is the interface between a Network Video Transmitter (NVT) and a Network Video Client (NVC). The specification covers device discovery, device configuration, events, PTZ control, video analytics and real time streaming functions.

The core specification defines the ONVIF framework, commands and requirements. All services share a common XML schema and all data types are defined in [ONVIF Schema]. The different services are defined in the respective service WSDL document.

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, Simple Object Access Protocol (SOAP) [SOAP 1.2, Part 1] and WSDL [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.

The ONVIF specification framework is built upon Web Services standards. All configuration services defined in the specification are expressed as Web Services operations and defined in WSDL with HTTP as the underlying transport mechanism.

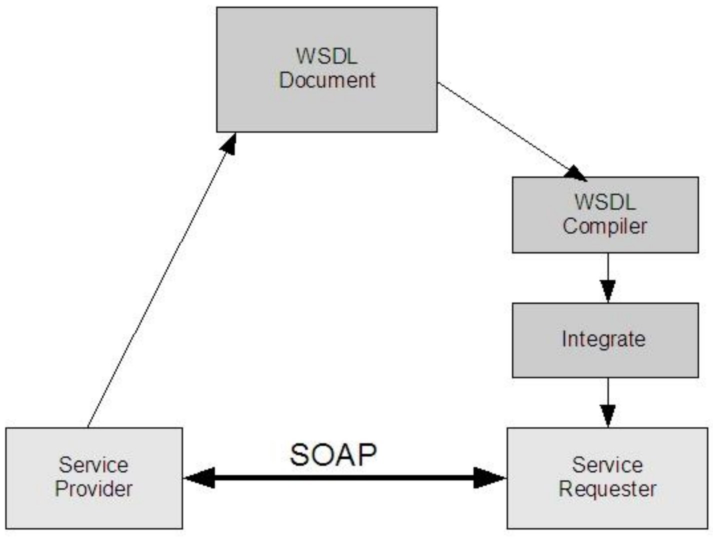


Figure 1: Web Services based development principles

Figure 1 gives an overview of the basic principles for development based on Web Services. The service provider (NVT) 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 (NVC) 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 specification 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 [WS-I BP 2.0] and use the document/literal wrapped pattern.

4.2 IP configuration

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

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

4.3 Device discovery

ONVIF-defined configuration interfaces are Web Services interfaces that are based on the WS-Discovery [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.

The specification defines ONVIF-specific discovery behaviour. 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.

All NVTs must include the service address of the device service. A successful discovery gives the NVT device service address. Once the NVC has the NVT device service address it can receive detailed NVT device information through the device service, see Section 4.4 below.

In addition, a new optional remote Discovery Proxy (DP) role is introduced in this specification. The remote DP allows NVTs to register the remote DP and NVCs to find registered NVTs through the remote DP even if the NVC and the NVT resides in different administrative network domains.

4.4 Device management

Device management functions are handled through the device service. The device service is the entry point to all other services provided by the NVT. WSDL for the device service is specified in [ONVIF DM WSDL]. The device management interfaces consist of these subcategories:

- Capabilities
- Network
- System
- Input/Output (I/O)
- Security

4.4.1 Capabilities

The capability commands allows an NVC to ask for the services provided by an NVT and to determine which ONVIF services, as well as brand specific services, are offered by the NVT. The capabilities are structured as the different NVT services and are further divided into subcategories (when applicable) as follows:

- Analytics
- Device
    - Capabilities
    - Network
    - System
    - I/O
    - Security
- Event
- Imaging
- Media
- PTZ
- *Storage*[1]

[1] Storage is NOT within the scope of the current specification but might be included in future versions.

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

4.4.2 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.

4.4.3 System

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

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

4.4.4 Security

The following security operations are used to manage the NVT 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.

4.4.5 Input/Output

The following Input/Output (I/O) commands are used to control the state or observe the status of the I/O ports. Currently only relay ports are handled.

- Get and set the mode of operation for the relay outputs.
- Trigger the relay to change the physical state.

4.5 Imaging configuration

The imaging service provides configuration and control data for imaging specific properties. WSDL is part of the framework and specified in [ONVIF Imaging WSDL].

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.

4.6 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 NVT provides media configuration through the media service. WSDL for the media service is specified in [ONVIF Media WSDL].

4.6.1 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).

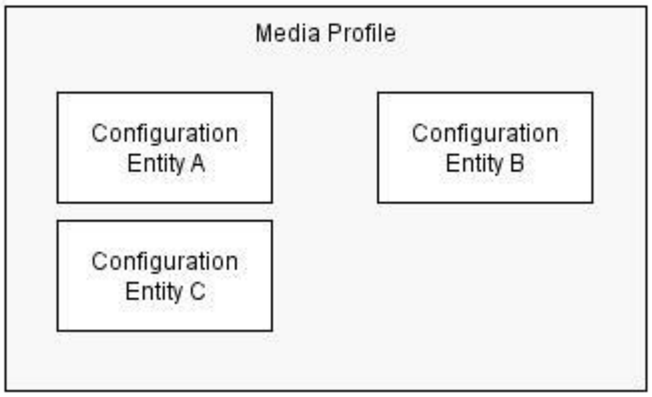


Figure 2: A media profile

An NVT MUST 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

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. All NVTs MUST support a fixed or dynamic profile with *at least* a video source and a video encoder configuration.

A complete profile configuration is illustrated in Figure 3.

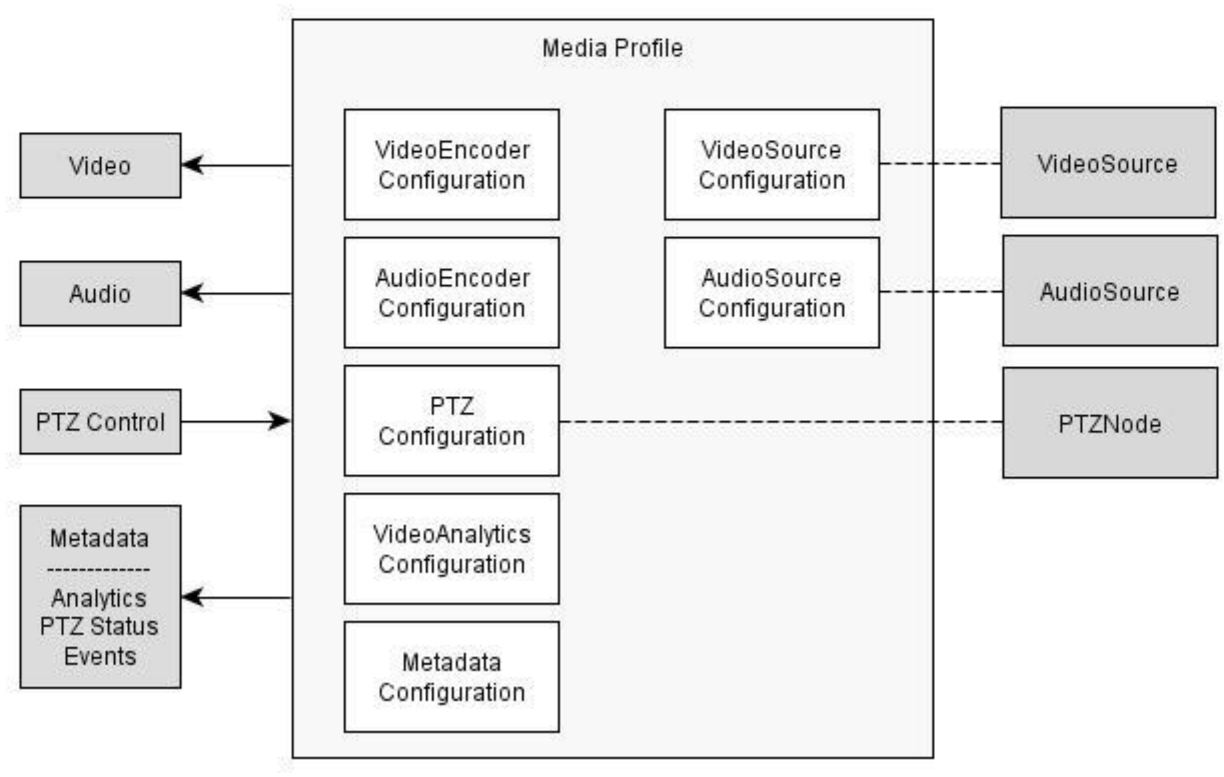


Figure 3: Complete profile configuration

A complete media profile defines how and what to present to the NVC 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.

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 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.

Refer to Section 4.12 for examples of how the profiles are used in a client implementation.

4.7 Real-time streaming

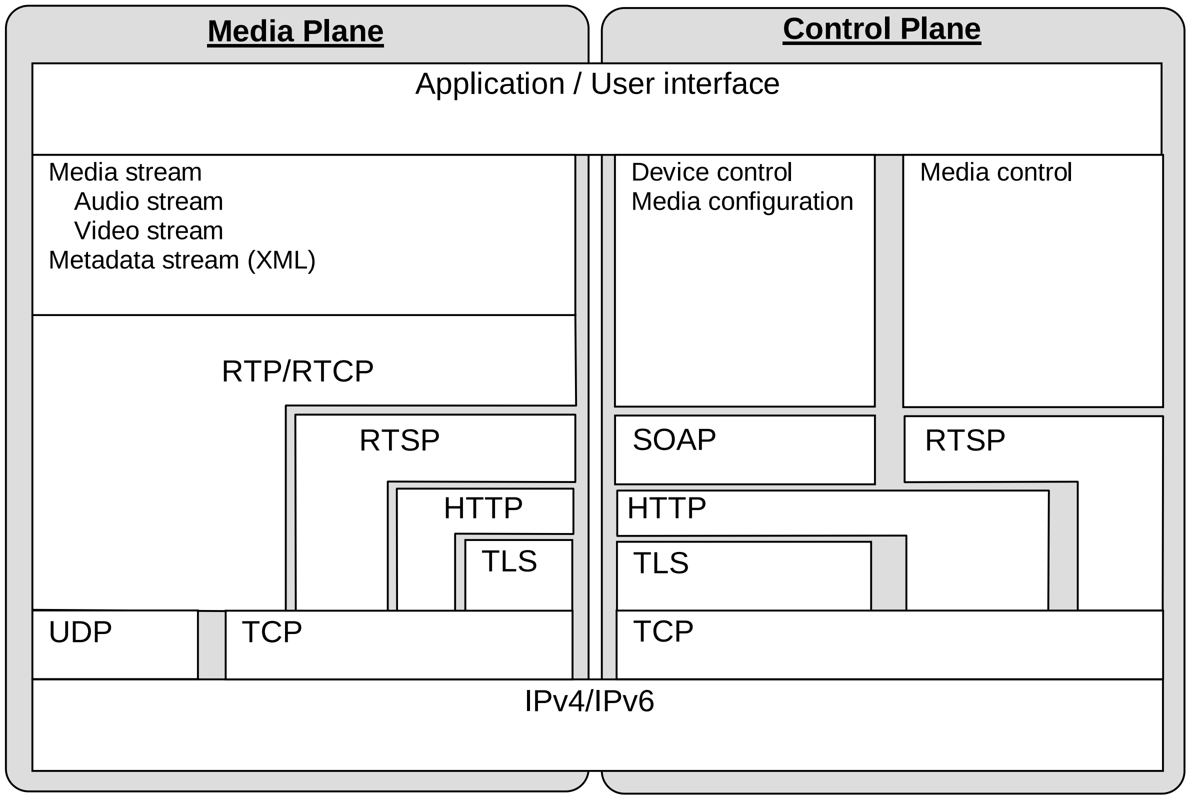


Figure 4: ONVIF layer structure

The ONVIF specification defines media streaming options and formats. A distinction is made between *media plane* and *control plane,* as illustrated in Figure 4. ONVIF defines a set of media streaming (audio, video and meta data) options all based on RTP [RFC 3550] 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 media configuration service as discussed in Section 4.5.

Media control is accomplished over RTSP [RFC 2326]. ONVIF defines RTP, RTCP and RTSP profiling, as well as JPEG over RTP extensions and multicast control mechanisms.

This specification provides streaming configurations for the following video codecs:

- JPEG (over RTP), see Section 11.1.3.
- 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]

4.8 Event handling

Event handling is based on the OASIS WS-BaseNotification [WS-BaseNotification] and WS-Topics [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. Consequently, this specification *does not* require particular notification topics, instead it defines a set of *basic* notification topics that an NVT is *recommended* to support, see Appendix A.

WSDL for the event service including extensions is specified in [ONVIF Event WSDL].

4.9 PTZ control

The PTZ service is used to control a Pan Tilt and Zoom (PTZ) video encoder device. WSDL for the PTZ service is specified in [ONVIF PTZ WSDL].

The PTZ control principle follows the *MediaProfile* model (see Section 4.6) 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.

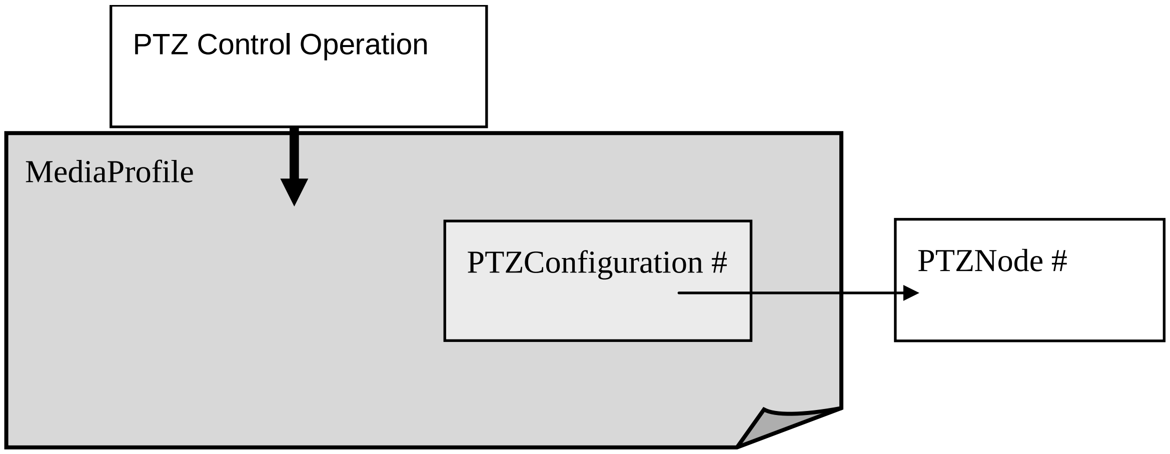


Figure 5: PTZ control model

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.

The ONVIF specification 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.

4.10 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). There are also object-oriented rules 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.

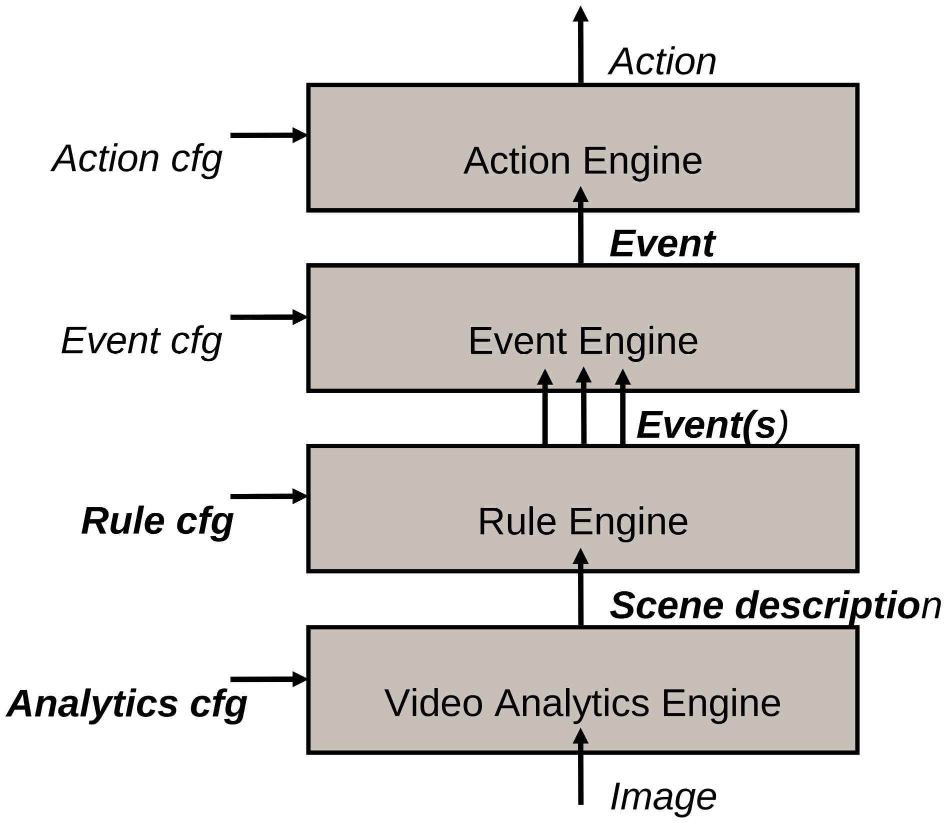


Figure 6: Video analytics architecture

The following interfaces are within the scope of the current ONVIF specification:

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

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

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 NVT.

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 Section 4.8). All other interfaces in the architecture are included for completeness only, but are *outside* the scope of the current specification.

WSDL for the video analytics service is part of the framework and specified in [ONVIF Analytics WSDL].

An NVT supporting analytics MUST implement the Scene Description and Event Interface, as well as the Configuration of Analytics by the Media Service. If the NVT additionally supports a rule engine, responsible for analytics engine as defined by ONVIF, then it MUST implement the Rules Analytics Modules Interface.

4.11 Security

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

- Transport-level security
- Message-level security

The general security requirements, definitions and transport security requirements are specified in Section 15. Message level security requirements are specified in Section 5.12. Security management is handled through the device management service as listed above in Section 4.4.4.

4.12 Client code examples

This section offers several examples of client code media configuration in C# pseudo code notation.

4.12.1 Requesting Stream URI

Client C1 wants to start streaming live video using media profile P1 and protocol RTP.

```
URI = GetStreamUri(RTP, P1)
---initiate an RTSP session using URI---
```

4.12.2 Media Setup

This example uses the following information:

- Client C1 connects to the NVT and would like to set up a JPEG video stream with frame rate f1 and resolution x1.
- Client C2 connects to the NVT and would like to set up an H.264 video stream with frame rate f2 and resolution x2 .
- Client C3 connects to the NVT and would like to set up an H.264 video stream but it does not care about the settings.
- Client C4 creates a new media profile using the source configuration of media profile P1 and the video encoder configuration of media profile P2.

Pseudo code example:

```
C1:
PA = GetProfiles()
foreach (Profile P1 in PA)
  if (P1.VideoEncoderConfiguration.Encoding == JPEG)
   E = P1.VideoEncoderConfiguration
   E.RateControl.FrameRate = f1
   E.Resolution = x1
   SetVideoEncoderConfiguration(E)
   ---start streaming using media profile P1---
   break

C2:
PA = GetProfiles()
foreach (Profile P2 in PA)
  if (P2.VideoEncoderConfiguration.Encoding == H264)
   E = P2.VideoEncoderConfiguration
   E.RateControl.FrameRate = f2
   E.Resolution = x2
   SetVideoEncoderConfiguration(E)
   ---start streaming using media profile P2---
   break

C3:
PA = GetProfiles()
foreach (Profile P3 in PA)
  If (P3. VideoEncoderConfiguration.Encoding == H264)
      ---start streaming using media profile P3---
      break

C4:
P1 = GetProfile(1)
P2 = GetProfile(2)
P4 = CreateProfile()
AddVideoSourceConfiguration(P4, P1.VideoSourceConfiguration)
AddVideoEncoderConfiguration(P4, P2.VideoEncoderConfiguration)
---start streaming using media profile P4---
```

5 Web Services frame work

All ONVIF management and configuration commands are based on Web Services.

For the purpose of this specification:

- The NVT is a service provider.
- The NVC is a service requester.

A typical network video system does not have a single client that handles all NVT configuration and device management operations for one NVT. Instead, a distinction between *control* and network video *receiver* functionality may exist. To simplify the descriptions and definitions, however, this specification introduces a single *client* role, the NVC. 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 NVT 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, NVTs (service providers or devices) 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 NVT and NVC MUST 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

The NVT Web Services defined in this specification are divided into different services categories. Each category has its *own* WSDL service definitions and name space (see Section 5.3 for the name space definitions). The following services are defined:

- device service
- media service
- event service
- imaging service
- PTZ service
- analytics service
- storage service

The device service is the *target service* of the NVT 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 Device service

The device service consists of a set of commands that control the NVT and get information for the services supported by the NVT. The device service category includes device management for network, system and security configurations. The device management service is defined in Section 8.

5.1.2 Media service

The media service is used to configure real-time streaming of video, audio and metadata. The stream configuration service is defined in Section 10.

5.1.3 Event service

The event service provides event notification services to an NVC. The event service is defined in Section 12.

5.1.4 Imaging service

The imaging service is used to handle NVT imaging configurations. The imaging service is defined in Section 9.

5.1.5 PTZ service

The PTZ service is used to control NVT pan tilt and zoom. The PTZ service is defined in Section 13.

5.1.6 Analytics service

The analytics related interfaces are used to control the video analytics functions of the NVT. The analytics service is defined in Section 14.

5.1.7 Storage service

The storage service provides network video storage control functions. Storage is *not within the scope* of this version of the specification, but may be included in a future version.

An NVT is free to implement a storage service as an extension. If an NVT provides such an extension, the NVT provider MAY define these extensions as a storage service. A proprietary storage service MUST NOT use the name space as defined in Section 5.3 for the storage service (as this name space is reserved for future ONVIF storage definitions) but should use an own name space.

5.1.8 Services requirements

An NVT MUST provide the device management, media and event services. An NVT MAY support any of the other services depending on the capabilities of the device. Given the functionality supported by the NVT, however, certain ONVIF services and functions are REQUIRED or OPTIONAL. The exact compliance requirements are defined as part of the different service definitions in this specification.

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

env:Receiver,

ter:ActionNotSupported,

see Section 5.11.2 for the definitions of the ONVIF error codes.

5.2 WSDL overview

"WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate" [WSDL1.1].

This specification follows the WSDL 1.1 specification [WSDL1.1] 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

The following prefix and namespaces are used in this specification:

Table 1: Defined namespaces in this specification

Prefix	Namespace URI	Description
tt	http://www.onvif.org/ver10/schema	XML schema descriptions in this specification.
tds	http://www.onvif.org/ver10/device/wsdl	The namespace for the WSDL device service.
trt	http://www.onvif.org/ver10/media/wsdl	The namespace for the WSDL media service.
timg	http://www.onvif.org/ver10/imaging/wsdl	The namespace for the WSDL imaging service.
tev	http://www.onvif.org/ver10/event/wsdl	The namespace for the WSDL event service.
tptz	http://www.onvif.org/ver10/ptz/wsdl	The namespace for the ptz control service.
tan	http://www.onvif.org/ver10/analytics/wsdl	The namespace for the analytics service.
tst	http://www.onvif.org/ver10/storage/wsdl	The namespace for the storage service.
ter	http://www.onvif.org/ver10/error	The namespace for ONVIF defined faults.
dn	http://www.onvif.org/ver10/network/wsdl	The namespace used for the *remote* device discovery service in this specification.
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace

The following namespaces are referenced by this specification:

Table 2: Referenced namespaces (with prefix)

Prefix	Namespace URI	Description
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace for WSDL framework.
wsoap12	http://schemas.xmlsoap.org/wsdl/soap12/	WSDL namespace for WSDL SOAP 1.2 binding.
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace for WSDL HTTP GET & POST binding.
soapenc	http://www.w3.org/2003/05/soap-encoding	Encoding namespace as defined by SOAP 1.2 [SOAP 1.2, Part 2]
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XML-Schema, Part 2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML schema instance namespace.
d	http://schemas.xmlsoap.org/ws/2005/04/discovery	Device discovery namespace as defined by [WS-Discovery].
wsadis	http://schemas.xmlsoap.org/ws/2004/08/addressing	Device addressing namespace referred in WS-Discovery [WS-Discovery].
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [WS-Addressing].
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 specification refers to the following namespaces (not referred by prefix)

Table 3: Referenced namespaces (without prefix)

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

This specification uses XML schema descriptions [XML-Schema, Part1], [XML-Schema, Part 2] when defining data types. 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

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 (i.e. "parameters") for the message part name.

We use the following WSDL notation for the messages in this specification

```
<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 any of the services prefixes defined in the specification (tds, trt, timg, tec, tan, tst, tptz, dn).

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 4.

Table 4: 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 case of that operation specific faults are defined, this field describe 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 Section 5.11.2.2. *In addition* any command can also through a *generic fault,* see Section 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 version of the specification does not define any one-way operations.

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**$_{\mathbf{r1}}$ [a_{r1}][b_{r1}] *Type*$_{r2}$ **Name**$_{\mathbf{r2}}$ [a_{r2}][b_{r2}] : *Type*$_{rn}$ **Name**$_{\mathbf{rn}}$ [a_{rn}][b_{rn}]
'Operation_Name'Response	*Description of the response message.* *Type*$_{s1}$ **Name**$_{\mathbf{s1}}$ [a_{s1}][b_{s2}] *Type*$_{s2}$ **Name**$_{\mathbf{s2}}$ [a_{s2}][b_{s2}] : *Type*$_{sn}$ **Name**$_{\mathbf{sn}}$ [a_{sn}][b_{sn}]
'‘FaultMessage_Name'	*In case of that operation specific faults are defined, this field describe 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.

An NVT service provider MAY define additional operations not defined in this specification.

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

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 an NVT according to this specification MUST follow the requirements in [WS-I BP 2.0]. This implies for example that the WSDL SOAP 1.2 bindings MUST be used.

The SOAP binding can have different styles. An NVT MUST 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 specification, the NVT and the NVC SHOULD 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 MUST 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.

ONVIF uses a separate name space for ONVIF specific faults:

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

SOAP fault messages for different Web Services are defined as part of the different Web Services definitions. The NVT and NVC MUST use SOAP 1.2 fault message handling as specified in this specification and MUST 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). The NVT and NVC 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

The table below lists the ONVIF generic fault codes and, if applicable, subcodes. All NVT and NVC implementations MUST handle all the faults listed below. Any ONVIF web service command may through one or several of the generic faults.

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

Table 5: Generic faults

Fault Code	Subcode	Fault Reason	Description
env:VersionMismatch		SOAP version mismatch	The device found an invalid element information item instead of the expected *Envelope* element information item.
env:MustUnderstand		SOAP header blocks not understood	One or more mandatory SOAP header blocks were not understood.
env:DataEncodingUnknown		Unsupported SOAP data encoding	SOAP header block or SOAP body child element information item is scoped with data encoding that is not supported by the device.
env:Sender	ter:WellFormed	Well-formed Error	XML Well-formed violation occurred.

env:Sender	ter:TagMismatch	Tag Mismatch	There was a tag name or namespace mismatch.
env:Sender	ter:Tag	No Tag	XML element tag was missing.
env:Sender	ter:Namespace	Namespace Error	SOAP Namespace error occurred.
env:Sender	ter:MissingAttr	Required Attribute not present	There was a missing required attribute.
env:Sender	ter:ProhibAttr	Prohibited Attribute	A prohibited attribute was present.
env:Sender	ter:InvalidArgs	Invalid 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 specification; see Section 8.6, Section 9.2, Section 10.15, Section 12.11, Section 13.9 and Section 14.4.

5.11.2.3 HTTP errors

If the NVT waits for the start of the inbound message and no SOAP message is received, the NVT MUST NOT generate a SOAP fault and instead sends an HTTP error response.

Table 6: HTTP errors

HTTP Error	HTTP Error Code	HTTP Reason
Malformed Request	400	Bad Request
Requires Authorization	401	Unauthorized
HTTP Method is neither POST or GET	405	Method Not Allowed
Unsupported message encapsulation method	415	Unsupported media

An NVT should avoid reporting internal errors as this can expose security weaknesses that can be misused.

5.12 Security

The services defined in this specification are protected using the WS-Security framework [WS-Security] .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]

An NVT and NVC MUST support the user name token profile as specified in [WS-Security] and Section 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 NVT for TLS-based access (see Section 15.1). 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.

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.

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

Section 8.4 defines commands to get and set an access security policy in arbitrary format.

5.12.2 User token profile

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

An ONVIF-compliant NVC MUST use both nonce and timestamps as defined in [WS-UsernameToken]. The NVT MUST reject any Username Token not using *both* nonce *and creation* timestamps.

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

6 IP configuration

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

The NVT MUST have at least one network interface that gives it IP network connectivity. Similarly, the NVC MUST have at least one network interface that gives IP connectivity and allows data communication between the NVT and NVC.

The NVT and NVC MUST support IPv4 based network communication. The NVT and NVC SHOULD support IPv6 based network communication.

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

The NVT SHOULD support dynamic IP configuration of link-local addresses according to [RFC3927]. An NVT that supports IPv6 must support stateless IP configuration according to [RFC4862] and neighbour discovery according to [RFC4861].

The NVT MUST support dynamic IP configuration according to [RFC 2131]. An NVT that supports IPv6 MUST support stateful IP configuration according to [RFC3315].

The NVT MAY support any additional IP configuration mechanism.

Network configuration of an NVT is accomplished through an IP network interface or though 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 MUST be possible to make NVT IP configurations through the parameter configuration interface specified in Section 8.2. An NVT user can enable or disable any of the IP address configuration options according to this specification through a network configuration interface. The default NVT configuration SHALL be to have both DHCP and dynamic link-local (stateless) address configuration enabled. Even if the NVT is configured through a static address configuration it SHOULD have the link-local address default enabled.

When an NVT 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 specification.

7 Device discovery

7.1 General

An NVC searches for available NVTs using the dynamic Web Services discovery protocol [WS-Discovery].

An NVT compliant with this specification MUST implement the Target Service role as specified in [WS-Discovery].

An NVC compliant with this specification MUST implement the Client role as specified in [WS-Discovery].

The Discovery Proxy role *as described in* [WS-Discovery] MUST NOT be supported by the NVT or NVC (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 NVT SHALL be able to operate in *two* modes:

- Discoverable
- Non-discoverable

An NVT 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 NVT *default* behaviour SHALL be the discoverable mode. In order to thwart denial-of-service attacks, it SHALL be possible to set an NVT into non-discoverable mode through the operation defined in Section 8.3.16.

7.3 Discovery definitions

7.3.1 Endpoint reference

An NVT or an endpoint that takes the client role SHOULD use a URN:UUID [RFC4122] as the address property of its endpoint reference.

The NVT or an endpoint that takes the client role MUST use a stable, globally unique identifier that is constant across network interfaces and uses IPv4/v6 addresses as part of the property of its endpoint reference. The combination of an wsadis:Address and wsadis:ReferenceProperties as defined in [WS-Addressing] 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 NVT device service address or addresses.

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

7.3.3 Hello

7.3.3.1 Types

An NVT MUST include the following device type in the `<d:Types>` declaration:

```
dn:NetworkVideoTransmitter,
```

The following declaration is in the SOAP Hello body:

```
<d:Types>dn:NetworkVideoTransmitter</d:Types>.
```

The Hello message MAY include additional types.

7.3.3.2 Scopes

An NVT MUST include the scope `<d:Scopes>` attribute with the scopes of the device in the Hello message.

The NVT scope is set by using [RFC 3986] URIs. The NVT SHALL use case-insensitive comparison when performing a probe match. 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>
```

Table 7 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.

Table 7: Scope parameters

Category	Defined values	Description
type	video_encoder	A video_encoder indicates that this device is a network video encoder device. An NVT with network video support, MUST include the video_encoder type in its scope list.
	ptz	A ptz scope indicates that the device is a ptz device. An NVT with PTZ support MUST 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 an NVT with audio encoder support MUST 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 Section 14. A video analytics NVT compliant with this specification MUST 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. An NVT MUST include at least one location entry into its scope list.
hardware	any character string or path value.	A string or path value describing the NVT hardware. An NVT MUST include at least one hardware entry into its scope list.
name	any character string or path value.	The searchable name of the device. An NVT MUST include at least one name entry into its scope list.

An NVT MUST include at least one entry of the type, location, hardware and name categories respectively in the scopes list. The NVT MAY include any other additional scope attributes in the scopes list.

An NVT 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/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
```

An NVC 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

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

The IP addressing configuration principles for an NVT are defined in Section 6.

7.3.4 Probe and Probe Match

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

The NVT MUST 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].

An NVT MUST 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 Section 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, an NVT SHOULD implement the resolve match response.

7.3.6 Bye

An NVT 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 NVT and NVC 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 an NVT receives a unicast Probe message and it does not support the matching rule, then the NVT 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. An NVT that supports remote service discovery MUST 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 NVC and the NVT *do not* reside in the same administrative domain, it is not possible for the NVC to find *and* connect to NVTs using multicast probe. For example, if the NVT or the NVC 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 NVT resides in one administrative domain (private) and the NVC resides in a public network, see Figure 7.

2. The NVT resides in a public network and the NVC resides in one administrative domain (private), see Figure 8.

3. The NVT resides in one administrative domain (private) and the NVC resides in *another* administrative domain (private), see Figure 9.

4. Both the NVT and the NVC reside in a public network, see Figure 10.

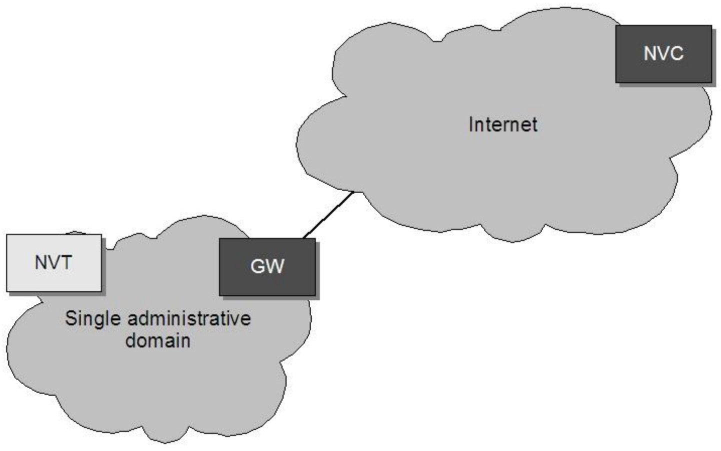


Figure 7: NVT in an administrative domain (private) and the NVC in a public network

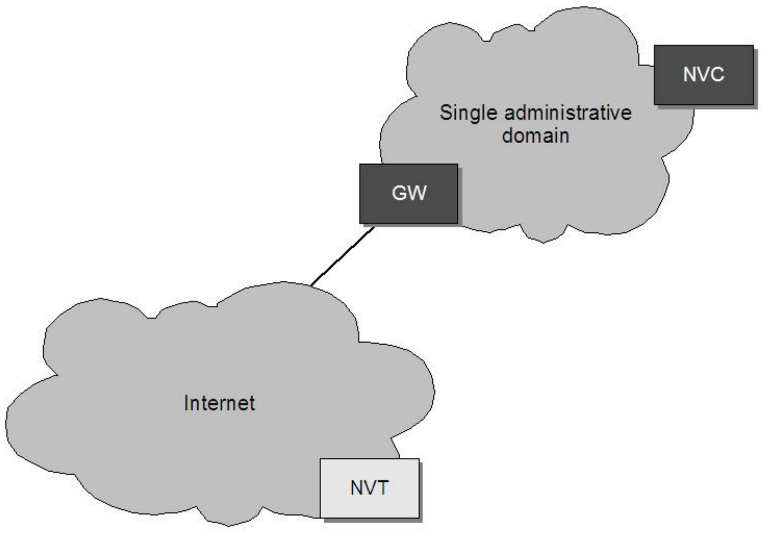


Figure 8: NVT in public network and the NVC in an adminstrative domain (private)

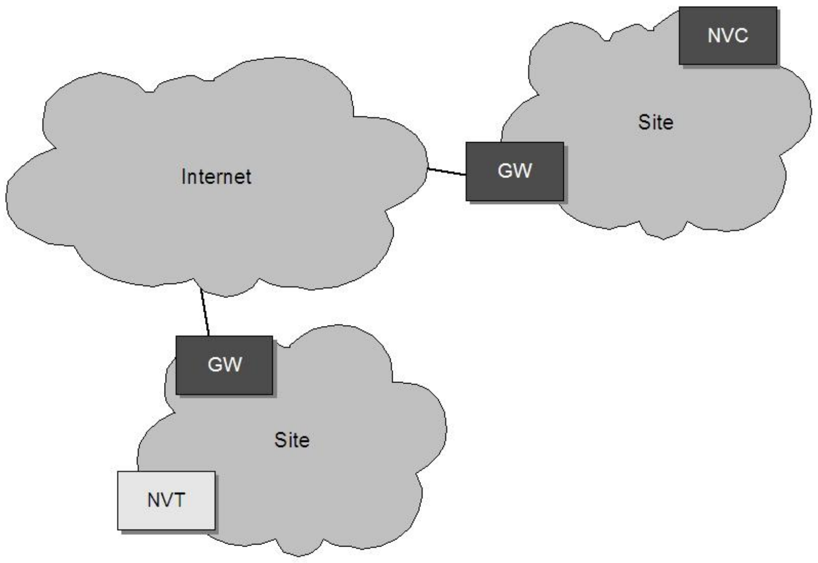


Figure 9: NVT in an administrative domain (private) and the NVC in another administrative domain (private)

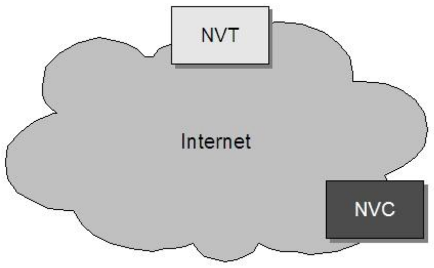


Figure 10: Both NVT and the NVC in a public network.

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 NVT hello messages and responds to these as defined in Section 7.4.3.

2. Responds to probe queries on behalf of registered NVTs from NVCs.

The DP may reside in the same administrative domain as the NVT. In order to support network scenarios where the NVC and NVT reside in different domains without multicast connectivity, place the DP in a publicly available network so that NVT and NVC endpoints can access it. It must be possible for the NVT 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 NVT 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 must be possible to enable/disable the NVT remote discovery registration. An NVT supporting remote discovery MUST implement the remote Hello disable/enable operation as defined in Section 8.3.18.

An NVT that is not configured with a home DP address or an NVT 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 8.3.19 and Section 8.3.20.

An NVT that supports remote discovery MAY also offer local configuration of the home DP address. Such configurations are done through an NVT 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 an NVT has remote discovery enabled but *lacks* remote DP address configuration, it MUST 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 NVT, a DP address must be configured using direct address configuration before enabling remote discovery.

In order for NVTs to make a successful DP lookup for NVTs, an administrator must 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 NVT resides behind a NAT/Firewall, like the scenarios shown in Figure 7 or Figure 9, a unicast Probe from the DP *will not* automatically reach the NVT if the NVT does not return a public network address. Furthermore, if the NVT resides behind a firewall, the NVT 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.

An NVT 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 NVT to the DP using the HTTP binding as defined in [ONVIF DP WSDL]. The remote Hello MUST include its scope list in the Hello message.

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

Similarly, when an NVT 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.

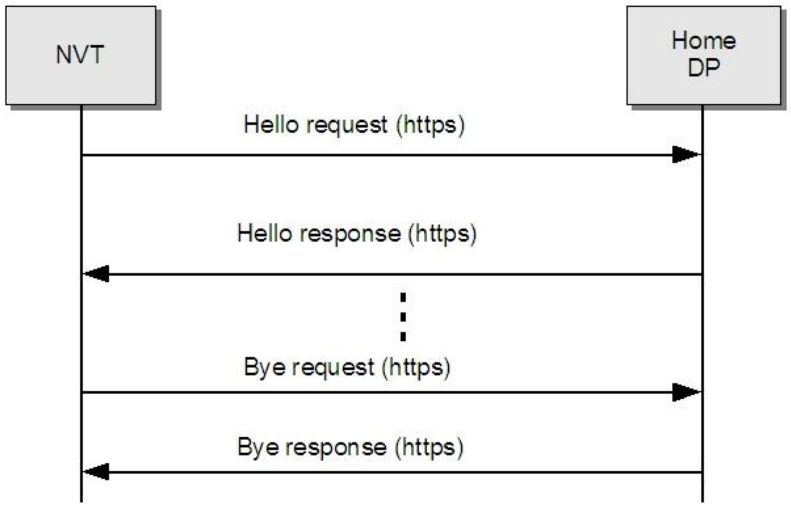


Figure 11: Remote discovery message exchange pattern

7.4.4 Client behaviour

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

7.4.4.1 NVC home DP configuration

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

An NVC 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 NVC to the DP using the http binding (see [ONVIF DP WSDL]).

Once the home DP receives a Probe message from any NVC, it responses with corresponding Probe Match message according to the normal WS-Discovery message exchange pattern, see the sequence chart in Figure 12.

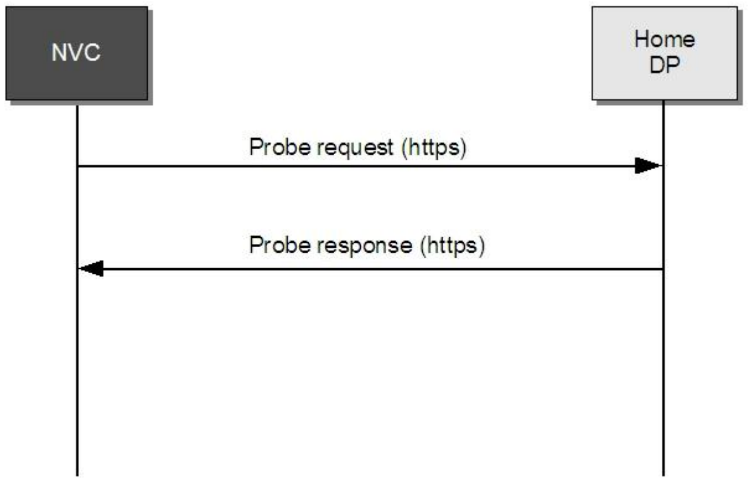


Figure 12: Message sequence for NVCs pre-configured with home DP address

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 NVTs or privacy issues if it is important to hide the presence of NVTs in the network. The risk of the latter two problems will very much depend on the NVT 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 NVC to switch off the device discovery function in the device. In non-discoverable mode, an NVT 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, must be taken to protect the DP from attacks. The remote Hello and Probe and Probe Match messages, as defined in Section 7.4.3, MUST 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 an NVT in the device data base the DP SHOULD authenticate the NVT to make sure that it is a “legal” NVT that announces its presence, for example by using NVT client certificates. NVT client certificate provisioning is outside the scope of the current specification.

The NVC to DP remote Probe and Probe Match messages MUST be sent over HTTPS. The DP MUST 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 management 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 NVT capabilities and configurations or to set NVT configurations. An NVT MUST 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 an NVT. The command gives in return a URL entry point where all the necessary product specific WSDL and schema definitions can be retrieved. The NVT MUST provide a URL for WSDL and schema download through the GetWsdlUrl 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 an NVT using the capability exchange request response operation. The NVT MUST 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 8 describes how to interpret the indicated capability. Apart from the addresses, the capabilities only reflect optional functions in this specification.

GetCapabilities	Request-Response
Message name	**Description**
GetCapabilitiesRequest	*This message contains a request for NVT capabilities. The NVC can either ask for all capabilities or just the capabilities for a particular service category.* tt:CapabilityCategory **Category** [0][unbounded]
GetCapabilitiesResponse	*The capability response message contains the requested NVT capabilities using a hierarchical XML capability structure.* tt:CapabilitiesResponseType **Capabilities** [1][1]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:NoSuchService	*The requested WSDL service category is not supported by the NVT.*

Table 8: The capabilities in the GetCapabilities command

Category	**Capability**	**Description**
Analytics	XAddr	The address to the analytics service. If this field is empty the NVT supports analytics but not the rules or module interfaces.
	RuleSupport	Indication if the NVT supports rules interface and rules syntax as specified in Section 14.2.
	AnalyticsModuleSupport	Indication if the NVT supports the scene analytics module interface as specified in Section 14.3.
Device	XAddr	The address to the device service.
Device – Network	IPFilter	Indication if the NVT 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 NVT supports zero configuration according to the commands in Section 8.2.15 and Section 8.2.16.
	IPVersion6	Indication if the NVT supports IP version 6.

	DynDNS	Indication if the NVT supports Dynamic DNS configuration according to Section 8.2.7and Section 8.2.8 .
Device – System	DiscoveryResolve	Indication if the NVT responses to resolve requests as described in Section 7.3.5.
	DiscoveryBye	Indication if the NVT sends bye messages as described in Section 7.3.6
	RemoteDiscovery	Indication if the NVT supports remote discovery support as specified in Section 7.4.
	SupportedVersions	List of the NVT supported ONVIF specification versions.
	SystemBackup	Indication if the NVT supports system backup and restore as specified in Section 8.3.4 and Section 8.3.5.
	FirmwareUpgrade	Indication if the NVT supports firmware upgrade as specified in Section 8.3.7.
	SystemLogging	Indication if the NVT supports system log retrieval as specified in Section 8.3.8.
Device – IO	InputConnectors	The number of input connectors.
	RelayOutputs	The number of relay outputs.
Device - Security	TLS1.1	Support of TLS 1.1.
	TLS1.2	Support of TLS 1.2.
	OnboardKeyGeneration	Indication if the NVT supports onboard key generation and creation of self-signed certificates as specified in Section 8.4.7.
	AccessPolicyConfig	Indication if the NVT supports retrieving and loading NVT access control polices according to Section 8.4.1 and Section 8.4.2.

	X.509Token	Indication if the NVT supports the WS-Security X.509 token [WS-X.509Token].
	SAMLToken	Indication if the NVT supports the WS-Security SAML token [WS-SAMLToken].
	KerberosToken	Indication if the NVT supports the WS-Security Kerberos token [WS-KerberosToken].
	RELToken	Indication if the NVT supports the WS-Security REL token [WS-RELToken].
Event	XAddr	The address to the event service
	WSSubscriptionPolicySupport	Indication if the NVT supports the WS Subscription policy according to Section12.1.2
	WSPullPointSupport	Indication if the NVT supports the WS Pull Point according to Section 12.1.2
	WSPausableSubscription-ManagerInterfaceSupport	Indication if the NVT supports the WS Pausable Subscription Manager Interface according to Section12.1.2
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 11.1.1.1.
	RTP_TCP	Indication if the NVT supports RTP over TCP, see Section 11.1.1.2.
	RTP_RTSP_TCP	Indication if the NVT supports RTP/RTSP/TCP transport, see Section 11.1.1.3.
PTZ	XAddr	The address to the PTZ service.

8.2 Network

8.2.1 Get hostname

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

Table 9: 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 beenobtained 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 an NVT. It MUST be possible to set the NVT hostname configurations through the SetHostname command.

Table 10: 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 NVT.*

8.2.3 Get DNS settings

This operation gets the DNS settings from an NVT. The NVT MUST return its DNS configurations through the GetDNS command.

Table 11: 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 an NVT. It MUST be possible to set the NVT DNS configurations through the SetDNS command.

Table 12: Set DNS command

SetDNS	Request-Response
Message name	**Description**
SetDNSRequest	*This message contains:* • *"SearchDomain": The domain(s) to search if the hostname is not fully qualified.* • *"DNSManual": A list of manually given DNS servers* 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 an NVT. If the NVT supports NTP, it MUST be possible to get the NTP server settings through the GetNTP command.

Table 13: 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][1] 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 an NVT. If the NVT supports NTP, it MUST be possible to set the NTP server settings through the SetNTP command.

Table 14: 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 an NVT. If the NVT supports dynamic DNS as specified in [RFC 2136] and [RFC 4702], it MUST be possible to get the type, name and TTL through the GetDynamicDNS command.

Table 15: 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 NVT desires no update, the NVT wants DHCP to update and the NVT does the update itself.* • *"Name": The DNS name in case of the NVT 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 an NVT. If the NVT supports dynamic DNS as specified in [RFC 2136] and [RFC 4702], it MUST be possible to set the type, name and TTL through the SetDynamicDNS command.

Table 16: SetDynamicDNS command

SetDynamicDNS	Request-Response
Message name	**Description**
SetDynamicDNSRequest	*This message contains:* • *"Type": The type of update. There are three possible types: the NVT desires no update, the NVT wants DHCP to update, and the NVT does the update itself.* • *"Name": The DNS name in case of the NVT 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 an NVT. The NVT MUST support return of network interface configuration settings as defined by the NetworkInterface type through the GetNetworkInterfaces command.

Table 17: GetNetworkInterfaces command

GetNetworkInterfaces	Request-Response
Message name	**Description**
GetNetworkInterfacesRequest	*This is an empty message.*
GetNetworkInterfacesRespon se	*This message contains an array of NVT 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 an NVT. The NVT MUST support network configuration of supported network interfaces through the SetNetworkInterfaces command.

Table 18: 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:* • *"Message": A message that can be used by NVTs that have to reboot in case of changes in the network settings*. xs:string **Message** [1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:InvalidNetworkInterface	*The supplied network interface token does not exists.*
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.*

8.2.11 Get network protocols

This operation gets defined network protocols from an NVT. The NVT MUST support the GetNetworkProtocols command returning configured network protocols.

Table 19: 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 NVT. 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 an NVT. The NVT MUST support configuration of defined network protocols through the SetNetworkProtocols command.

Table 20: SetNetworkProtocols command

SetNetworkProtocols	Request-Response
Message name	**Description**
SetNetworkProtocolsRequest	*This message configures one ore more defined network protocols supported by the NVT. 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 an NVT. The NVT MUST support the GetNetworkDefaultGateway command returning configured default gateway(s).

Table 21: 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 an NVT. The NVT MUST support configuration of default gateway through the SetNetworkDefaultGateway command.

Table 22: 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 an NVT. If the NVT supports dynamic IP configuration according to [RFC3927], it MUST support the return of IPv4 zero configuration address and status through the GetZeroConfiguration command

Table 23: 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 NVT. If the NVT supports dynamic IP configuration according to [RFC 3927], it MUST support the configuration of IPv4 zero configuration address and status through the SetZeroConfiguration command.

Table 24: 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 an NVT. If the NVT supports NVT access control based on IP filtering rules (denied or accepted ranges of IP addresses), the NVT MUST support the GetIPAddressFilter command.

Table 25: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 an NVT. If the NVT supports NVT access control based on IP filtering rules (denied or accepted ranges of IP addresses), the NVT MUST support configuration of IP filtering rules through the SetIPAddressFilter command.

Table 26: 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 an NVT. If the NVT supports NVT access control based on IP filtering rules (denied or accepted ranges of IP addresses), the NVT MUST support adding of IP filtering addresses through the AddIPAddressFilter command.

Table 27: 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 an NVT. If the NVT supports NVT access control based on IP filtering rules(denied or accepted ranges of IP addresses), the NVT MUST support deletion of IP filtering addresses through the RemoveIPAddressFilter command.

Table 28: RemoveIPAddressFilter command

RemoveIPAddressFilter	Request-Response
Message name	**Description**
RemovePAddressFilterReque st	*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.*

8.3 System

8.3.1 Device Information

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

Table 29: GetDeviceInformation command

GetDeviceInformation	Request-Response
Message name	**Description**
GetDeviceInformationRequest	*This is an empty message.*
GetDeviceInformationRespon se	*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 Backup

This operation is retrieves system backup configuration file(s) from an NVT. The NVT 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 specification.

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

Table 30: 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!

8.3.3 Restore

This operation restores the system backup configuration files(s) previously retrieved from an NVT. The NVT 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 specification. If the command is supported, it MUST accept backup files returned by the GetSystemBackup command.

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

Table 31: 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.*

8.3.4 Get system date and time

This operation gets the NVT system date and time. The NVT MUST 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.

If system time and date are set manually, the NVT MUST return UTCDateTime *or* LocalDateTime in the response.

Table 32: 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 NVT.* • *"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 NVT* tt:SetDateTimeType **DateTimeType** [1][1] xs:boolean **DayLightSavings** [1][1] tt:TimeZone **TimeZone** [0][1]

	tt:DateTime **UTCDateTime** [0][1] tt:DateTime **LocalTimeDate** [0][1]
Fault codes	**Description**
	No command specific faults!

8.3.5 Set system date and time

This operation sets the NVT system date and time. The NVT MUST 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 NVC MUST include UTCDateTime *or* LocalDateTime in the request.

Table 33: SetSystemDateAndTime command

SetSystemDateAndTime	Request-Response
Message name	**Description**
SetSystemDateAndTime-Request	*This message contains the date and time information of the NVT.* • *"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.*

8.3.6 Factory default

This operation reloads parameters of an NVT to their factory default values. The NVT MUST support hard and soft factory default through the SetSystemFactoryDefault command. The meaning of *soft factory default* is NVT product-specific and vendor-specific. The effect of a *soft factory default* operation is not fully defined. However, it must 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 34: 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 NVT 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!

8.3.7 Firmware upgrade

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

The firmware is transmitted through MTOM [MTOM].

Table 35: 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 NVC 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.*

8.3.8 Get system logs

This operation gets a system log from an NVT. The NVT SHOULD support system log information retrieval through the GetSystemLog command. The exact format of the system logs is *outside the scope* of this specification.

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

Table 36: 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 NVC access log* tt:SystemLogType **LogType** [1][1]
GetSystemLogResponse	*This message contains the requested system log information. The NVT can choose if it wants to return the system log information as binary data in an attachment or as a common string.* tt:AttachmentData **BinaryFormat** [0][1] xs:string **StringFormat** [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*

8.3.9 Get support information

This operation gets arbitrary device diagnostics information from an NVT. The NVT MAY support retrieval of diagnostics information through the GetSystemSupportInformation command. The exact format of the diagnostic information is *outside the scope* of this specification.

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

Table 37: 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 NVT 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.*

8.3.10 Reboot

This operation reboots an NVT. Before the NVT reboots the response message MUST be sent. The NVT MUST support reboot through the SystemReboot command.

Table 38: 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 NVC as for an example "Rebooting in x seconds."* xs:string **Message** [1][1]
Fault codes	**Description**
	No command specific faults!

8.3.11 Get scope parameters

This operation *requests* the scope parameters of an NVT. 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 8.3.11 and Section 8.3.12. The NVT MUST support retrieval of discovery scope parameters through the GetScopes command. As some scope parameters are mandatory, the NVC always expects a scope list in the response.

Table 39: 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.*

8.3.12 Set scope parameters

This operation *sets* the scope parameters of an NVT. 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 NVT MUST support configuration of discovery scope parameters through the SetScopes command.

Table 40: 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.*

8.3.13 Add scope parameters

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

Table 41: 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.*

8.3.14 Remove scope parameters

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

Table 42: RemoveScopes command

RemoveScopes	Request-Response
Message name	**Description**
RemoveScopesRequest	*The set 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.*

8.3.15 Get discovery mode

This operation gets the discovery mode of an NVT. See Section 7.2 for the definition of the different NVT discovery modes. The NVT MUST support retrieval of the discovery mode setting through the GetDiscoveryMode command.

Table 43: 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!

8.3.16 Set discovery mode

This operation sets the discovery mode operation of an NVT. See Section 7.2 for the definition of the different NVT discovery modes. The NVT MUST support configuration of the discovery mode setting through the SetDiscoveryMode command.

Table 44: 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!

8.3.17 Get remote discovery mode

This operation gets the remote discovery mode of an NVT. See Section 7.4 for the definition of remote discovery remote extensions. An NVT that supports remote discovery MUST support retrieval of the remote discovery mode setting through the GetRemoteDiscoveryMode command.

Table 45: 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!

8.3.18 Set remote discovery mode

This operation sets the remote discovery mode of operation of an NVT. See Section 7.4 for the definition of remote discovery remote extensions. An NVT that supports remote discovery MUST support configuration of the discovery mode setting through the SetRemoteDiscoveryMode command.

Table 46: 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!

8.3.19 Get remote DP addresses

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

Table 47: GetDPAddresses command

GetDPAddresses	Request-Response
Message name	**Description**
GetDPAddressesRequest	*This is an empty message.*
GetDPAddressesResponse	*This message contains the NVT 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!

8.3.20 Set remote DP addresses

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

Table 48: SetDPAddresses command

SetDPAddresses	Request-Response
Message name	**Description**
SetDPAddressesRequest	*This message contains the NVT 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 must be protected using appropriate authorization levels in order not to compromise the NVT security (anonymous or media users are *not* allowed to perform these operations).

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 specification does not mandate any particular policy description format or security policy but this is up to the NVT 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 NVT supports access policy settings based on WS-Security authentication, then the NVT MUST support this command.

Table 49: 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 NVT policy file does not exist or it is empty.*

8.4.2 Set access policy

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

Table 50: 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 an NVT. The NVT MUST support retrieval of registered NVT users and their credentials for the user token through the GetUsers command.

Table 51: 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* • *UserL 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 NVT users and corresponding credentials on an NVT for the user token profile, see Section 5.12 for user token definitions. The NVT MUST support creation of NVT users and their credentials for the user token through the CreateUsers command. Either all users are created successfully or a fault message MUST be returned.

Table 52: 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 ter:UsernameClash	*Username already exists.*
env:Sender ter:OperationProhibited ter:PasswordTooLong	*The password was too long*
env:Sender ter:OperationProhibited ter:UsernameTooLong	*The username was too long*
env:Sender ter:OperationProhibited ter:Password	*Too weak password.*

env:Receiver ter:Action ter: TooManyUsers	*Maximum number of supported users exceeded.*

8.4.5 Delete users

This operation deletes users on an NVT for the user token profile, see Section 5.12 for user token definitions. The NVT MUST support deletion of NVT users and their credentials for the user token through the DeleteUsers command. Either all users are deleted successfully or a fault message MUST be returned.

Table 53: 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.*

8.4.6 Set users settings

This operation updates the settings for one or several users on an NVT for the user token profile.. The NVT MUST support update of NVT users and their credentials for the user token through the SetUser command. Either all change requests are processed successfully or a fault message MUST be returned.

Table 54: SetUser command

SetUser	Request-Response
Message name	**Description**
SetUserRequest	*This message contains a list of a 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.*

8.4.7 Create self-signed certificate

This operation creates a self-signed NVT certificate. The certificate is created using a suitable *onboard* key generation mechanism. If the NVT supports onboard public key pair generation, the NVT MUST support the certificate creation command. Certificates are identified using certificate IDs. These IDs are either chosen by the certificate generation requester or by the NVT (in case of that no ID value is given).

Table 55: 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.*

8.4.8 Get certificates

This operation gets all NVT *server* certificates for the device (including self-signed). The server certificates are used for TLS authentication purposes. This command lists *only* the server certificates of the device (not trusted client certificates or trusted roots). The certificates are returned as binary data. An NVT MUST support this command and the certificates MUST be encoded using ASN.1 [X.681], [X.682], [X.683] DER [X.690] encoding rules. Each certificate has a unique device name.

Table 56: GetCertificates command

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

8.4.9 Get certificate status

This operation gets the status (enabled/disabled) of the NVT TLS server certificates. An NVT MUST support this command.

Table 57: GetCertificatesStatus command

GetCertificatesStatus	Request-Response
Message name	**Description**
GetCertificatesStatusRequest	*This is an empty message.*
GetCertificatesStatus-Response	*This message contains a list of the NVT 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!

8.4.10 Set certificate status

This operation sets the status (enable/disable) of the NVT TLS server certificates. An NVT MUST support this command. Typically *only* one NVT server certificate is allowed to be enabled at a time.

Table 58: SetCertificatesStatus command

SetCertificatesStatus	Request-Response
Message name	**Description**
SetCertificatesStatusRequest	*This message contains a list of NVT 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.*

8.4.11 Get certificate request

This operation requests a PKCS #10 certificate signature request from the device. The returned binary information field MUST be formatted exactly as specified in [PKCS#10]. In order for this command to work, the NVT must already have a private key to be certified. The key is referred by a certificate. This might for example be a self-signed certificate generated using the command defined in Section 8.4.7.

An NVT that supports onboard key generation MUST support this command.

Table 59: 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 be encoded as DER ASN.1 objects.* xs:token **CertificateID** [1][1] xs:string **Subjec**t [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	*Unknown certificate reference.*
env:Receiver ter:Action ter:Signature	*PKCS#10 signature creation failed.*

8.4.12 Get client certificate status

This operation gets the status (enabled/disabled) of the NVT TLS client authentication. An NVT MUST support this command.

Table 60: GetClientCertificateMode command

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

8.4.13 Set client certificate status

This operation sets the status (enabled/disabled) of the NVT TLS client authentication. An NVT MUST support this command.

Table 61: SetClientCertificateMode command

SetClientCertificateMode	Request-Response
Message name	**Description**
SetClientCertificateMode-Request	*This message contains the requested NVT 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.*

8.4.14 Load NVT certificate

NVT TLS server certificate(s) created using the PKCS#10 certificate request command, for example, can be loaded into the NVT using this command (see Section 8.4.7). The certificate ID in the request is optional set to the ID value the NVC wish to have. The NVT may sort the received certificate(s) based on the public key and subject information in the certificate(s).

Table 62: LoadCertificates command

LoadCertificates	Request-Response
Message name	**Description**
LoadCertificatesRequest	*This message contains a list of the NVT server 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 NVT.*
env:Sender ter:InvalidArgVal ter:CertificateID	*Certificate ID already exists.*

8.5 Input/Output (I/O)

8.5.1 Get relay outputs

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

Table 63: 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 64: 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[2].

Table 65: 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]
SetRelayOutputStateResponse	*This is an empty message.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:RelayToken	*Unknown relay token reference.*

8.6 Service specific fault codes

Table 66 lists the device service-specific fault codes. In addition, each command can also generate a generic fault, see Table 5.

The specific faults are defined as 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 66: Device service specific fault codes

Fault Code	Parent Subcode Subcode	Fault Reason	Description
env:Receiver	ter:Action ter:EmptyPolicy	The policy is empty	The NVT policy file does not exist or it is empty.
env:Receiver	ter:Action ter:EmptyScope	The scope list is empty	Scope list is empty.
env:Receiver	ter:Action ter: FirmwareUpgradeFailed	Upgrade failed	The firmware upgrade failed.
env:Receiver	ter:Action	Generating a key failed	The private/public key generation failed.

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

	ter:KeyGeneration		
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 NVT.
env:Sender	ter:InvalidArgVal 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 NVT.
env:Sender	ter:InvalidArgVal ter:CertificateID	Invalid ID	Unknown certificate reference or the certificate ID already exists.
env:Sender	ter:InvalidArgVal ter:InvalidBackupFile	Invalid file	The backup file(s) are invalid.
env:Sender	ter:InvalidArgVal ter:InvalidDateTime	Invalid data	An invalid date or time was specified.
env:Sender	ter:InvalidArgVal ter:InvalidDnsName	Invalid name	The suggested NTP server name is invalid.
env:Sender	ter:InvalidArgVal ter:InvalidFirmware	Invalid firmware	The firmware was invalid i.e. not supported by this device.
env:Sender	ter:InvalidArgVal	Invalid address	The supplied gateway address was invalid.

	ter:InvalidGatewayAddress		
env:Sender	ter:InvalidArgVal ter:InvalidHostname	Invalid name	The requested hostname cannot be accepted by the NVT.
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:InvalidArgVal 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:ScopeOverwrite	Trying overwriting permanent device scope setting	Scope parameter overwrites permanent device scope setting, command rejected.
env:Sender	ter:OperationProhibited ter: UsernameClash	Username already exists	Username already exists.
env:Sender	ter:OperationProhibited	Too long username	The username is too long.

	ter:UsernameTooLong		

9 Imaging configuration

The imaging service provides operations used to control and configure imaging properties on an NVT. The NVT 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.

9.1 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 colour 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 NVT:
    - 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 – Disable exposure algorithm on the NVT:
    - 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 9.1.4.

Ir cut filter: Toggle 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: White dynamic range (on/off):

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

The available NVT imaging settings can be retrieved through the GetVideoSources command part of the media service, as specified in Section 10.3.1. The imaging settings are part of the video source.

9.1.1 Get imaging settings

This operation requests the imaging setting for a video source on the NVT. 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 NVT through the GetImagingSettings command.

The imaging settings parameters are described in Section 9.1.

Table 67: 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:ImagingSettings **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.*

9.1.2 Set imaging settings

This operation sets the imaging settings for a video source on an NVT. If the NVT 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 NVT through the SetImagingSettings command.

The possible configurable imaging settings parameters are described in Section 9.1. Settings options are obtained through the command defined in Section 9.1.3

Table 68: 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:ImagingSettings **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.*

9.1.3 Get options

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

Table 69: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:ImagingOptions **ImagingOptions**[0][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.*

9.1.4 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. An NVT with support for remote focus control SHOULD support absolute, relative or continuous control through the *Move* operation.

Imaging capabilities specifies which specific focus operations that is 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 70: 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:Focus **FocusMove**[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.*

9.1.5 Get move options

The GetMoveOptions command retrieves the focus lens move options to be used in the move command as defined in Section 9.1.4. An NVT that supports the lens move operation MUST also support the GetMoveOptions command.

Table 71: 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:MoveOptions **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.*

9.1.6 Stop

The stop command stops all ongoing focus movements in a profile on the NVT. If the NVT supports focus, it SHOULD be possible to stop focus through the stop operation. The operation will not affect ongoing autofocus operation.

Table 72: 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.*

9.1.7 Get imaging status

The GetStatus command requested the current imaging status from the NVT. If the NVT 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 73: 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:ImagingStatus **ImagingStatus**[1][unbounded]
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.*

9.2 Service specific fault codes

Table 74 lists the imaging service specific fault codes. In addition each command can also generate a generic fault, see Table 5.

The specific faults are defined as subcode of a generic fault, see 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 74: Imaging specific fault codes

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

10 Media configuration

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

The media configuration service allows an NVC 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 4.6.

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
- Media streaming:
  - Request stream URI
  - Get snapshot
  - 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.

10.1 Audio and video codecs

The NVT streams audio and video data using suitable encoding algorithms. The NVT supports any audio and video codecs, bitrates and resolution according to the manufacturer’s choice. In order to ensure interoperability between the NVC and NVT, this specification mandates the following codec profiles:

- The NVT MUST support JPEG QVGA.

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

10.2 Media Profile

A media profile consists of a set of media configurations as described in Section 4.6. Media profiles are used by an NVC to configure properties of a media stream from an NVT.

10.2.1 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 MUST support the creation of media profiles as defined in this specification through the CreateProfile command.

Table 75: CreateProfile command

CreateProfile	Request-Response
Message name	**Description**
CreateProfileRequest	*Contains the friendly **Name** of the Profile to create as well as an optional **ProfileToken** parameter, specifying the unique identifier of the new media profile* tt:Name **Name** [1][1] tt:ReferenceToken **ProfileToken** [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.*

10.2.2 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 NVC does not need to know the media profile in order to use the command. The NVT MUST support the retrieval of media profiles through the GetProfiles command.

Table 76: 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!

10.2.3 Get media profile

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

Table 77: 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.*

10.2.4 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 MUST support addition of a video source to a profile through the AddVideoSourceConfiguration command.

Table 78: 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.*

10.2.5 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 MUST support addition of a video encoder 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 79: 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.*

10.2.6 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 NVC MUST support addition of audio source to a profile through the AddAudioSourceConfiguration command.

Table 80: 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.*

10.2.7 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 NVC MUST support addition of audio 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 81: 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.*

10.2.8 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 MUST 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 13.

Table 82: 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.*

10.2.9 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 MUST 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 14.1 and analytics configurations are managed through the commands defined in Section 10.9.

A profile containing only a video analytics configuration but no video source configuration is incomplete. Therefore, an NVC 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 ConflictingConfiguration Fault.

Table 83: 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.*

10.2.10 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 MUST 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 10.9.4.

Table 84: 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.*

10.2.11 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 MUST support removal of a video source from a profile through the RemoveVideoSourceConfiguration command.

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

Table 85: 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:ProfileToken **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.*

10.2.12 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 MUST support removal of a video encoder configuration from a profile through the RemoveVideoEncoderConfiguration command.

Table 86: 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:ProfileToken **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.*

10.2.13 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 NVC MUST support removal of an audio source from a profile through the RemoveAudioSourceConfiguration command.

Table 87: 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:ProfileToken **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.*

10.2.14 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 NVC MUST support removal of audio configurations from a profile through the RemoveAudioEncoderConfiguration command.

Table 88: 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:ProfileToken **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.*

10.2.15 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 MUST support removal of PTZ configurations from a profile through the RemovePTZConfiguration command.

Table 89: 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:ProfileToken **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.*

10.2.16 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 MUST support removal of an analytics configuration from a profile through the RemoveVideoAnalyticsConfiguration command.

Table 90: 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:ProfileToken **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.*

10.2.17 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 MUST support the removal of a metadata configuration from a profile through the RemoveMetadataConfiguration command.

Table 91: 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:ProfileToken **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.*

10.2.18 Delete media profile

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

Table 92: 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.*

10.3 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 9 for more information.

10.3.1 GetVideoSources

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

Table 93: 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!

10.4 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.

10.4.1 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 NVC need not know anything about the video source configurations in order to use the command. The NVT MUST support the listing of available video source configurations through the GetVideoSourceConfigurations command.

Table 94: 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!

10.4.2 Get video source configuration

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

Table 95: GetVideoSourceConfiguration command

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

10.4.3 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 MUST support the listing of compatible (with a specific profile) video source configurations through the GetCompatibleVideoSourceConfigurations command.

Table 96: 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.*

10.4.4 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 MUST support the listing of available video source parameter options (for a given profile and configuration) through the GetVideoSourceConfigurationOptions command.

Table 97: 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.*

10.4.5 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 NVC requests a new stream URI and restarts any affected stream. If the NVC intends to change a running stream, RTSP commands should be used. The NVT MUST support the modification of video source parameters through the SetVideoSourceConfiguration command.

Table 98: SetVideoSourceConfiguration command

SetVideoSourceConfiguration	Request-Response
Message name	**Description**
SetVideoSourceConfiguration-Request	*The **Configuration** element contains the modified video source configuration. The configuration must 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.*

10.5 Video encoder configuration

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

- Encoding – 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 means higher quality and a higher bitrate.
- RateControl – A structure defining the output framerate and bitrate limit.
- MPEG4/H264 specifics – Defines the encoding profile and GOV length.

TheVideoEncoderConfiguration structure also contains multicast parameters and a session timeout to define video streaming behaviour.

10.5.1 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 NVC need not know anything apriori about the video encoder configurations in order to use the command. The NVT MUST support the listing of available video encoder configurations through the GetVideoEncoderConfigurations command.

Table 99: 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!

10.5.2 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 MUST support the retrieval of a specific video encoder configuration through the GetVideoEncoderConfiguration command.

Table 100: 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.*

10.5.3 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 MUST support the listing of compatible (with a specific profile) video encoder configurations through the GetCompatibleVideoEncoderConfigurations command.

Table 101: 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.*

10.5.4 Get video encoder configuration options

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

Table 102: 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.*

10.5.5 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. The changes are not guaranteed to take effect unless the NVC requests a new stream URI and restarts any affected stream. If the NVC intends to change a running stream, RTSP commands should be used. The NVT MUST support the modification of video encoder parameters through the SetVideoEncoderConfiguration command.

Table 103: SetVideoEncoderConfiguration command

SetVideoEncoderConfiguration	Request-Response
Message name	**Description**
SetVideoEncoderConfiguration-Request	*The **Configuration** element contains the modified video encoder configuration. The configuration must 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.*

10.6 Audio source

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

10.6.1 Get audio sources

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

Table 104: 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.*

10.7 Audio source configuration

An AudioSourceConfiguration contains a reference to an AudioSource that is to be used for input in a media profile.

10.7.1 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 NVC 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 NVC MUST support listing of available audio source configurations through the GetAudioSourceConfigurations command.

Table 105: 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. A audio source configuration does always point at a real audio source with the SourceToken element.* tt:AudioEncoderConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.7.2 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 NVC MUST support the retrieval of a specific audio source configuration through the GetAudioSourceConfiguration command.

Table 106: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.7.3 Get compatible audio source configurations

This operation requests all NVT the audio source configurations 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 NVC MUST support listing of compatible (with a specific profile) audio source configurations through the GetCompatibleAudioSourceConfigurations command.

Table 107: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.7.4 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 NVC MUST support the listing of available audio parameter options (for a given profile and configuration) through the GetAudioSourceConfigurationOptions command.

Table 108: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.7.5 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. The changes are not guaranteed to take effect unless the NVC requests a new stream URI and restarts any affected stream. If the NVC intends to change a running stream, RTSP commands SHOULD be used. An NVT that supports audio streaming from NVT to NVC MUST support the configuration of audio source parameters through the SetAudioSourceConfiguration command.

Table 109: SetAudioSourceConfiguration command

SetAudioSourceConfiguration	Request-Response
Message name	**Description**
SetAudioSourceConfiguration-Request	*The **Configuration** element contains the modified audio source configuration. The configuration must 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.8 Audio encoder configuration

An AudioEncoderConfiguration contains the following parameters for encoding audio data:

- Encoding – The encoding used for audio data.
- Bitrate – The output bitrate.
- SampleRate – The output sample rate.

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

10.8.1 Get audio encoder configurations

This operation lists all *existing* NVT audio encoder configurations. This command lists *all* configured audio encoder configurations in a device. The NVC 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 NVC MUST support the listing of available audio encoder configurations through the GetAudioEncoderConfigurations command.

Table 110: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.8.2 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 NVC MUST support the listing of a specific audio encoder configuration through the GetAudioEncoderConfiguration command.

Table 111: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.8.3 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 NVC MUST support listing of compatible (with a specific profile) audio encoder configurations through the GetCompatibleAudioEncoderConfigurations command.

Table 112: 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.8.4 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 NVC MUST support the listing of available audio encoder parameter options (for a given profile and configuration) through the GetAudioEncoderConfigurationOptions command.

Table 113: GetAudioEncoderConfigurationOptions command

GetAudioEncoderConfigurationOptions	Request-Response
Message name	**Description**
GetAudioEncoderConfiguration-OptionsRequest	*This message contains optional tokens of a 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:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.8.5 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 NVC requests a new stream URI and restarts any affected streams. If the NVC intends to change a running stream, RTSP commands SHOULD be used. An NVT that supports audio streaming from NVT to NVC MUST support the configuration of audio encoder parameters through the SetAudioEncoderConfiguration command.

Table 114: SetAudioEncoderConfiguration command

SetAudioEncoderConfiguration	Request-Response
Message name	**Description**
SetAudioEncoderConfiguration-Request	*The **Configuration** element contains the modified audio encoder configuration. The configuration must 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	*The configuration parameters are not possible to set.*

ter:ConfigModify	
env:Receiver ter:Action ter:ConfigurationConflict	*The new settings conflicts with other uses of the configuration.*
env:Sender ter:ActionNotSupported ter:AudioNotSupported	*NVT does not support audio.*

10.9 Video analytics configuration

VideoAnalyticsConfiguration contains parameters for an *analytics engine* and a *rule engine* (see Section 4.10). 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 10.10).

An NVT MAY NOT allow referencing the very same VideoAnalyticsConfiguration from multiple media profiles with different VideoSourceConfigurations. If the NVT allows it, it MUST 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.

10.9.1 Get video analytics configurations

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

Table 115: 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 NVT.* tt:VideoAnalyticsConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**

env:Sender ter:ActionNotSupported ter:VideoAnalyticsNot-Supported	*NVT does not support video analytics.*

10.9.2 Get video analytics configuration

the GetVideoAnalyticsConfiguration command fetches the video analytics configuration if the video analytics token is known. An NVT that supports video analytics MUST support the listing of a specific video analytics configuration through the GetVideoAnalyticsConfiguration command.

Table 116: 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	*NVT does not support video analytics.*

10.9.3 Get compatible video analytics configurations

This operation requests all video analytic 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 AddVideoAnalyticsConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. An NVT that supports video analytics MUST support the listing of compatible (with a specific profile) video analytics configuration through the GetCompatibleVideoAnalyticsConfigurations command.

Table 117: 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 ter:VideoAnalyticsNot-Supported	*NVT does not support video analytics.*

10.9.4 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 NVT or not. Running streams using this configuration MUST 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. An NVT that supports video analytics MUST support the configuration of video analytics parameters through the SetVideoAnalyticsConfiguration command.

Table 118: SetVideoAnalyticsConfiguration command

SetVideoAnalyticsConfiguration	Request-Response
Message name	**Description**
SetVideoAnalyticsConfiguration-Request	*The **Configuration** element contains the modified video analytics configuration. The configuration must 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: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	*NVT does not support video analytics.*

ter:VideoAnalyticsNot-Supported	

10.10 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 12.5. The analytics parameters define which data to include from the analytics engine part of the profile, see Section 10.9.

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 11.2.1.1.1)

10.10.1 Get metadata configurations

This operation lists all *existing* NVT metadata configurations. This command lists *all* configured metadata in a device. The NVC need not know anything apriori about the metadata in order to use the command. The NVT MUST support the listing of existing metadata configurations through the GetMetadataConfigurations command.

Table 119: 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 NVT.* tt:MetadataConfiguration **Configurations** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

10.10.2 Get metadata configuration

The GetMetadataConfiguration command fetches the metadata configuration if the metadata token is known. The NVT MUST support the listing of a specific metadata configuration through the GetMetadataConfigurations command.

Table 120: 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.*

10.10.3 Get compatible metadata configurations

This operation requests all the metadata 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 AddMetadataConfiguration command on the media profile. The result varies depending on the capabilities, configurations and settings in the device. The NVT MUST support the listing of compatible (with a specific profile) metadata configuration through the GetCompatibleMetadataConfigurations command.

Table 121: 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.*

10.10.4 Get metadata configuration options

This operation returns the available options for changing the metadata configuration. The NVT MUST support the listing of available metadata parameter options (for a given profile and configuration) through the GetMetadataConfigurationOptions command.

Table 122: 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.*

10.10.5 Modify a metadata configuration

This operation modifies a metadata configuration. The ForcePersistence flag indicates if the changes shall remain after reboot of the NVT. 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 NVC requests a new stream URI and restarts any affected streams. If the NVC intends to change a running stream, RTSP commands SHOULD be used. The NVT MUST support the configuration of metadata parameters through the SetMetadataConfiguration command.

Table 123: 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	*The configuration parameters are not possible to set.*

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

10.11 Stream URI

10.11.1 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 URI is valid only as it is specified in the response or until the Profile is reconfigured. The NVT MUST support the retrieval of a media stream URI for a specific media profile through the GetStreamUri command.

Table 124: 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 **Uri** to be used for requesting the media stream as well as parameters defining the lifetime of the Uri If **ValidUntilConnect** is true, the URI is only valid for one connection. If **ValidUntilReboot** is true the URI is only valid until the device is rebooted. If **Timeout** is more than zero, the URI is only valid for this long.* xs:anyURI **Uri** [1][1] xs:boolean **ValidUntilConnect** [1][1] xs:boolean **ValidUntilReboot** [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.*

10.12 Snapshot

10.12.1 Request snapshot URI

An NVC uses the GetSnapshotUri command to obtain a JPGEG shapsot form the NVT. The returned URI is valid only as long as is specified in the response or until the Profile is reconfigured. 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 125: 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 the **Uri** to be used for acquiring a snapshot in JPEG format as well as parameters defining the lifetime of the Uri If **ValidUntilConnect** is true, the URI is only valid for one connection. If **ValidUntilReboot** is true the URI is only valid until the device is rebooted. If **Timeout** is more than zero, the URI is only valid for this long.* xs:anyURI **Uri** [1][1] xs:boolean **ValidUntilConnect** [1][1] xs:boolean **ValidUntilReboot** [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.*

10.13 Multicast

See Section 11.1 for a detailed discussion of NVT and NVC multicast streaming.

10.13.1 Start multicast streaming

Commands the device to start multicast streaming using a specified media profile. 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 or audio multicast streaming MUST support the starting of a multicast stream through the StartMulticastStreaming command.

Table 126: 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 configuration or a reference to a video source configuration.*

10.13.2 Stop multicast streaming

Commands the device to stop multicast streaming of the specified media profile. An NVT that supports video or audio multicast streaming MUST support the stopping of a multicast stream through the StopMulticastStreaming command.

Table 127: 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 configuration or a reference to a video source configuration.*

10.14 Synchronization Points

10.14.1 Set synchronization point

Synchronization points allow NVCs 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 NVC can request a Synchronization Point which enforces the NVT to add an I-Frame as soon as possible. NVCs can request Synchronization Points for profiles. The NVT MUST 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 MUST be added to this video stream. If an event stream is associated to the profile, the synchronization point request MUST be handled as described in Section 12. If a PTZ metadata stream is associated to the profile, the PTZ position MUST be repeated within the metadata stream.

An NVT that supports MPEG-4 or H.264 MUST support the request for an I-Frame though the SetSynchronizationPoint command.

Table 128: SetSynchronizationPoint command

SetSynchronizationPoint	Request-response
Message name	**Description**
SetSynchronizationPoint-Request	Contains a Profile reference for which a Synchronization Point is requested. tt:ReferenceToken **ProfileToken** [1][1]
SetSynchronizationPoint-Response	This message is empty.
SetSynchronizationPoint faults	
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoProfile	The profile does not exist.

10.15 Service specific fault codes

Table 129 lists the media service specific fault codes. Additionally, each command can also generate a generic fault, see Table 5.

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 129: Media service specific fault codes

Fault Code	**Parent Subcode**	**Fault Reason**	**Description**
	Subcode		
env:Receiver	ter:ActionNotSupported	No audio capability	NVT does not support audio.
	ter:AudioNotSupported		

env:Receiver	ter:Action ter:MaxNVTProfiles	Maximum number reached	The maximum number of supported profiles has been reached.
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.

11 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 Section 10.

11.1 Media stream protocol

11.1.1 Transport format

Real-time Transport Protocol (RTP) is a media transfer protocol (see Section 11.1.2). The following four sections describe RTP data transfer..

11.1.1.1 RTP data transfer via UDP

UDP has the smallest overhead and is able to transfer real-time data in an efficient manner. An NVT MUST support the RTP/UDP protocol and the NVT SHOULD support RTP/UDP multicasting.

11.1.1.2 RTP/TCP

If there is a packet loss during media transfer via UDP, then the specification allows for RTP data transfer via TCP as an alternative means of media transport. An NVT, however, MAY support the RTP/TCP based option. If the NVT 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).

11.1.1.3 RTP/RTSP/TCP

The NVT SHOULD support media streaming using RTP/RTSP to traverse a firewall using an RTSP tunnel. This protocol MUST conform to [RFC 2326] (RTSP Section 11.2.1.1: Embedded [Interleaved] Binary Data).

11.1.1.4 RTP/RTSP/HTTP/TCP

The data stream MUST be sent via HTTP or HTTPS to traverse a firewall. An NVT MUST support media streaming using RTP/RTSP/HTTP/TCP and RTP/RTSP/HTTPS/TCP.

This protocol MUST conform to [RFC 2326] (RTSP Section 11.2.1.1: Embedded [nterleaved] Binary Data).

This tunnelling method MUST also conform with QuickTime available from Apple Inc. The mandatory parts of the following document MUST be implemented by an NVT.

`http://developer.apple.com/documentation/QuickTime/QTSS/Concepts/chapter_2_section_14.html`

11.1.2 Media Transport

11.1.2.1 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 11.1.3).

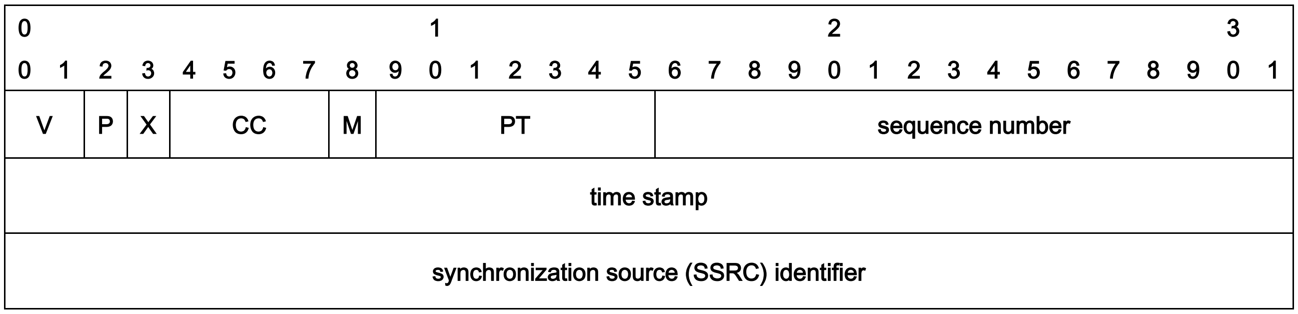


Figure 13: RTP header

An RTP header MUST be filled up with following values.

Table 130: 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. Extension MUST be used, if “JPEG over RTP” is used in this packet. . (see Section 11.1.3).
CSRC count (CC): 4 bits	0	
Marker (M): 1 bit	0/1	Usage depends on a codec. MUST conform to related RFCs (e.g. [RFC 3984] for H.264 Video).
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 11.1.2.2.1 for further details of Media Synchronization. Usage depends on a codec.
SSRC 32 bits		The synchronization source for the data stream. This specificaton makes no restrictions on the use of this field.

11.1.2.1.1 RTP for Metadata stream

Metadata streams are also transported by RTP. ONVIF defines the usage of payload type, marker and timestamp for RTP header for the metadata stream. 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 MUST 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 must 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 10.14.1) is requested for the stream, the previous XML document MUST be closed and a new one started. It is RECOMMENDED to start new XML documents after 5 seconds, 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. An NVT can select which of these parts SHOULD be multiplexed into the Metadata during the Media Configuration (see Section 10.10). Each part can appear multiple times in arbitrary order within the document.

The XML structure of the Metadata stream is as follows:

```
<xs:element name="MetaDataStream">
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="VideoAnalytics" type="tt:VideoAnalyticsStream"/>
        <xs:element name="PTZ" type="tt:PTZStream"/>
        <xs:element name="Event" type="tt:EventStream"/>
        ...
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The place-holders for the different metadata sources have the following structure:

```
<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>
```

It follows 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:Event>
    <wsnt:Message>
      <tt:Message UtcTime= "2008-10-10T12:24:57.628">
      ...
      </tt:Message>
    </wsnt:Message>
  </tt:Event>

  <tt:VideoAnalytics>
    <tt:Frame UtcTime="2008-10-10T12:24:57.921">
      ...
    </tt:Frame>
  </tt:VideoAnalytics>
```

```
</tt:MetaDataStream>
```

11.1.2.2 RTCP

The RTP Control Protocol provides feedback on quality of service being provided by RTP and synchronization of different media streams. The RTCP protocol MUST conform to [RFC 3550].

For a feedback request, [RFC 4585] and [RFC 5104] SHOULD be supported.

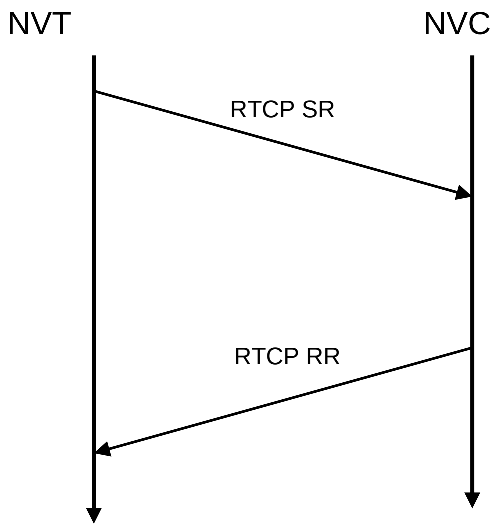


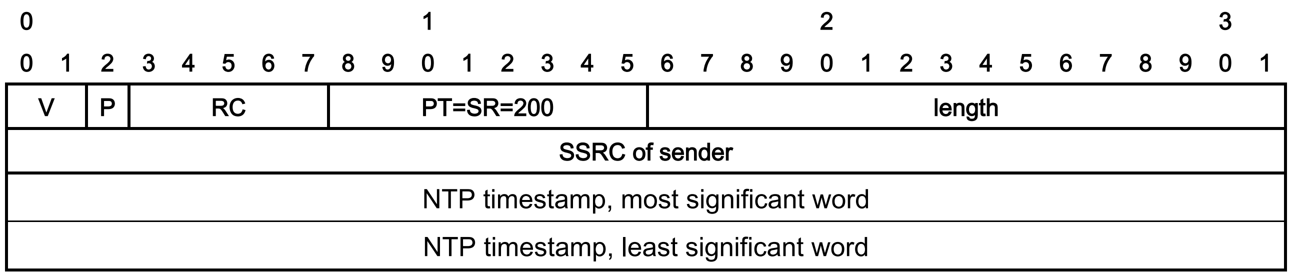
Figure 14: RTCP sequence

11.1.2.2.1 Media synchronization

An NVC MAY receive audio and video streams simultaneously from more than one NVT. In this case, each stream uses a different clock (from data acquisition to packet receiving). To synchronize different media streams, ONVIF defines usage of RTCP Sender Report (SR). RTCP SR MUST be 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 15.

An NVT MUST support RTCP Sender Report for media synchronization. The NVC SHOULD use RTCP for the media synchronization.



RTP timestamp
sender's packet count
:
:

Figure 15: RTCP Sender Report

The wall clock SHOULD be common in the NVT and each timestamp value SHOULD be determined properly. The NVC can synchronize different media streams at the appropriate timing based on the RTP clock and wall clock timestamps (see Figure 16).

In case of multiple NVTs, the NTP timestamp SHOULD be common to all NVTs, and the NTP server SHOULD be required in the system [3]. .

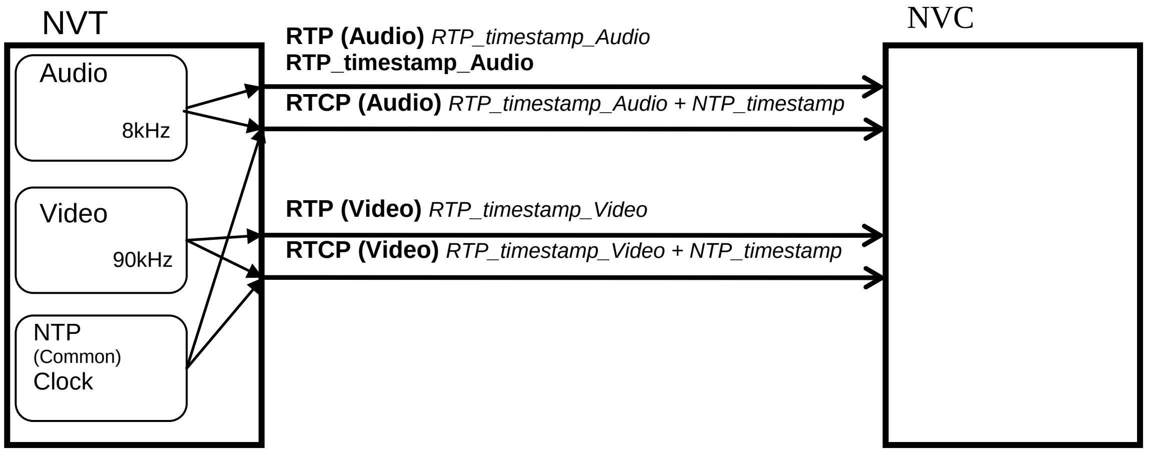

NVT
NVC
Audio
8kHz
Video
90kHz
NTP
(Common)
Clock
RTP (Audio) RTP_timestamp_Audio
RTP_timestamp_Audio
RTCP (Audio) RTP_timestamp_Audio + NTP_timestamp
RTP (Video) RTP_timestamp_Video
RTCP (Video) RTP_timestamp_Video + NTP_timestamp

Figure 16: Media Synchronization

11.1.3 JPEG over RTP

11.1.3.1 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 17.

[3] The NVC can get information about "NTP server availability" from the NVT by using the GET SOAP command. Refer to Section 8.2.5, Get NTP settings

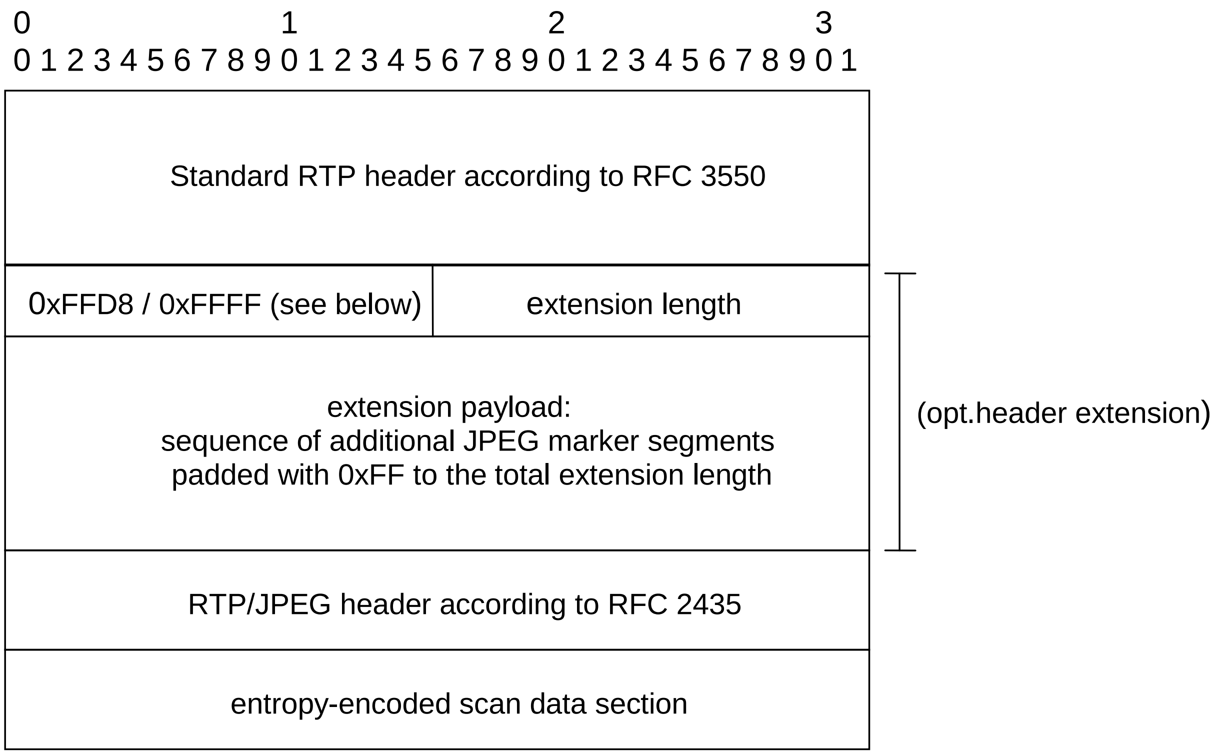

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Standard RTP header according to RFC 3550
0xFFD8 / 0xFFFF (see below)
extension length
extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length
(opt.header extension)
RTP/JPEG header according to RFC 2435
entropy-encoded scan data section

Figure 17: 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 MUST 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 specification, it’s 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 MUST NOT include this header.

11.1.3.2 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 MUST 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.

11.1.3.3 Supported colour spaces and sampling factors

NVTs SHOULD use only greyscale and YCbCr colour space. NVCs 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.

11.1.3.4 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.

11.1.3.5 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.

11.2 Media control protocol

11.2.1 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 10.11.1.

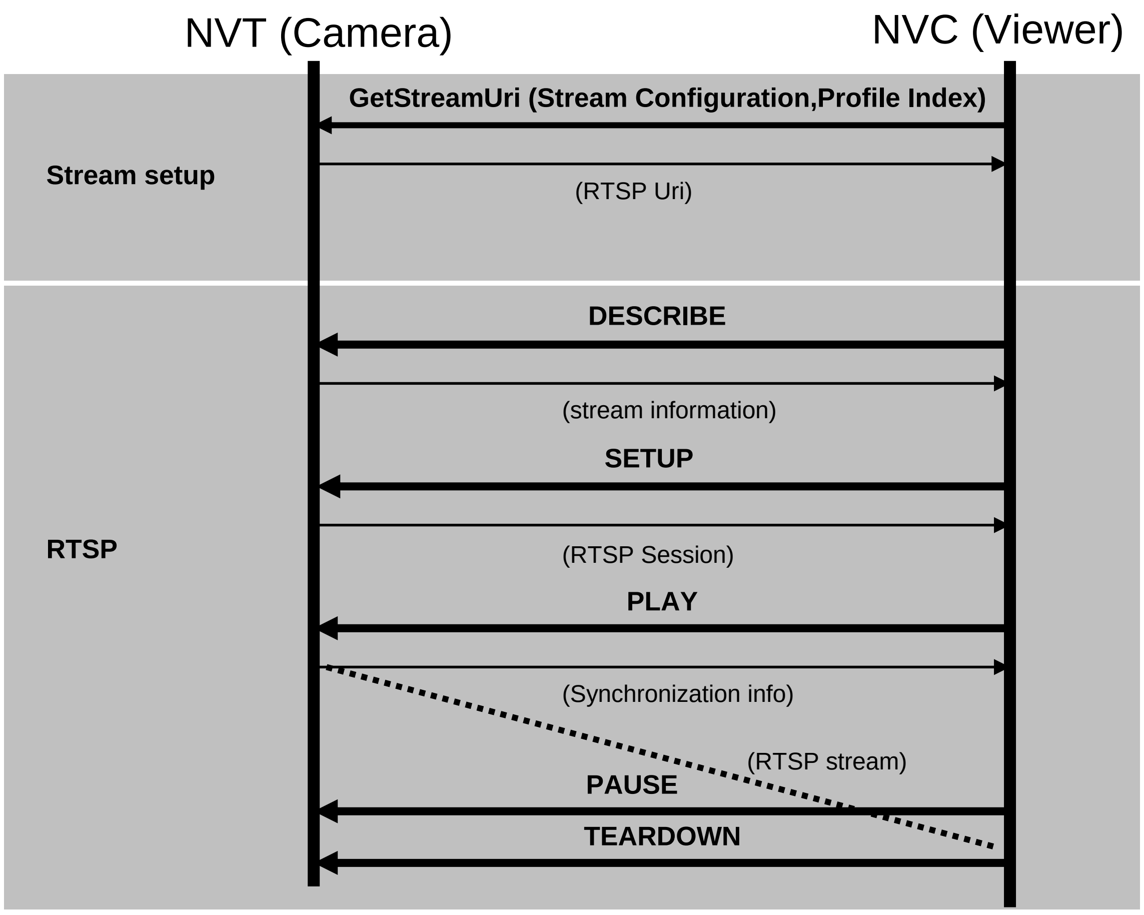


Figure 18: Stream Control

11.2.1.1 RTSP

All NVTs and NVCs MUST support RTSP ([RFC 2326]) for session initiation and playback control. RTSP MUST use TCP as its transport protocol, the typical 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 131: RTSP methods.

Method	Direction	ONVIF 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	M O	Required to keep an RTSP session alive (R->T direction only).
REDIRECT	T->R	X	
RECORD	R->T	X	

11.2.1.1.1 Keep-alive method for RTSP session

The NVC (RTSP client) keeps the RTSP Session alive and prevents it from session timeout (see [RFC 2326) Section 12.37). ONIVF recommends the following methods to keep RTSP alive for both Unicast and Multicast streaming.

1) The NVC can optionally set the Timeout parameter (in seconds) using the `Set<configurationEntity>EncoderConfiguration` command defined in Section 10.2, otherwise a default value of “60” is used.

4 X: Not supported, M: Mandatory, O: Optional

2) In all RTSP SETUP responses, the NVT SHOULD include the Timeout value according to [RFC 2326] Section 12.37 and the NVC SHOULD use the Timeout value for keep-alive.

3) To keep the RTSP Session alive, the NVC MUST call the NVT using any RTSP method. `SET_PARAMETER` is the RECOMMENDED RTSP method to use.

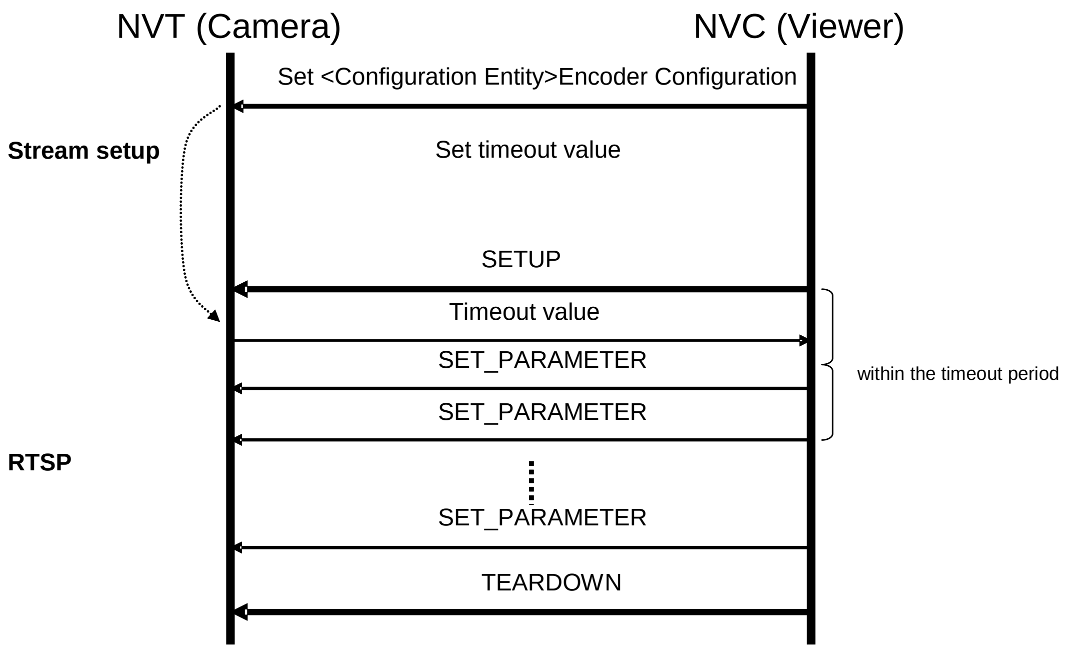


Figure 19: Keep Alive

11.2.1.1.2 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 use d for encoding a name.

Example of SDP;

```
NVC->NVT:        DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
                 CSeq: 1
NVT->NVC:        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=sendonly
                 a=rtpmap
                 a=rtpmap:107 vnd.onvif.metadata/90000
```

11.2.1.1.3 RTSP message example

The RTSP exchanges a message when the NVC requests one audio and one video stream from the NVT. The `GetStreamUriResponse` returns the RTSP Server URI (rtsp://example.com/nvif_camera). Refer to Section 10.11.1.

```
NVC->NVT:           DESCRIBE rtsp://example.com/onvif_camera RTSP/1.0
                    CSeq: 1
NVT->NVC:           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
NVC->NVT:           SETUP rtsp://example.com/onvif_camera/audio RTSP/1.0
                    CSeq: 2
                    Transport: RTP/AVP;unicast;client_port=8002-8003
NVT->NVC:           RTSP/1.0 200 OK
                    CSeq: 2
                    Transport: RTP/AVP;unicast;client_port=8002-8003;
                                 server_port=9004-9005
                    Session: 12345678; timeout=60
NVC->NVT:           SETUP rtsp://example.com/onvif_camera/video RTSP/1.0
                    CSeq: 3
                    Transport: RTP/AVP;unicast;client_port=8004-8005
       Session: 12345678
NVT->NVC:           RTSP/1.0 200 OK
                    CSeq: 3
                    Transport: RTP/AVP;unicast;client_port=8004-8005;
                    server_port=9006-9007
                    Session: 12345678; timeout=60
NVC->NVT:           PLAY rtsp://example.com/onvif_camera RTSP/1.0
                    CSeq: 4
                    Range: npt=now-
                    Session: 12345678
NVT->NVC:           RTSP/1.0 200 OK
                    CSeq: 4
                    Session: 12345678
                    RTP-Info: url=rtsp://example.com/onvif_camera;
                           seq=9810092;rtptime=3450012
NVC->NVT:           TEARDOWN rtsp://example.com/onvif_camera RTSP/1.0
                    CSeq: 5
                    Session: 12345678
NVT->NVC:           RTSP/1.0 200 OK
                    CSeq: 5
                    Session: 12345678
```

11.2.1.2 RTSP over HTTP

The RTSP over HTTP/HTTPS MUST be supported in order to traverse a firewall. See Section 11.1.1.4 RTP/RTSP/HTTP/TCP.

11.3 Error Handling

RTSP and HTTP protocol errors are classified into different categories (for example, status codes 1xx, 2xx, 3xx, 4xx and 5xx respectively). The NVT and the NVC SHOULD 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.

12 Event handling

An event is an action or occurrence detected by an NVT that an NVC can subscribe to. Events are handled through the event service. An NVT MUST provide an event services as defined in [ONVIF Event WSDL].

Event Handling in this specification is based on the [WS-BaseNotification] and [WS-Topics] specifications (subsequent sections use the terminology described in the specifications). This specification requires the implementation of the basic notification interface as described in section 12.1, which conforms completely to the [WS-BaseNotification] specification. In addition, the NVT MUST implement the Real-time Pull-Point Notification Interface and the Notification Streaming Interface as introduced in sections 12.2 and 12.3, respectively.

In order to allow tracking of object properties (such as video analytics object properties) within the notification framework, this specification introduces notification message extensions that allow an NVC to track properties through events. Properties are defined in Section 12.4.

The description of event payload and their filtering within subscriptions is discussed in Section 12.5. Section 12.6 describes how Synchronization Point can be requested by clients using one of the three Notification Interfaces. Section 12.7 describes the integration of Topics. Section 12.9 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.

12.1 Basic Notification Interface

Section 12.1.1 briefly introduces the Basic Notification Interface of the [WS-BaseNotification] specification. Section 12.1.2 summarizes the mandatory and the optional interfaces of the [WS-BaseNotification] specification.

12.1.1 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 an NVT.

Typical messages exchanged between the entities are shown in the sequence diagram in Figure 20. First, the Client establishes a connection to the Event Service. The Client can then subscribe for certain notifications by sending a SubscriptionRequest via this connection. If the Event Service accepts the Subscription, it dynamically instantiates a SubscriptionManager representing the Subscription. The Event Service MUST 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 NVT.

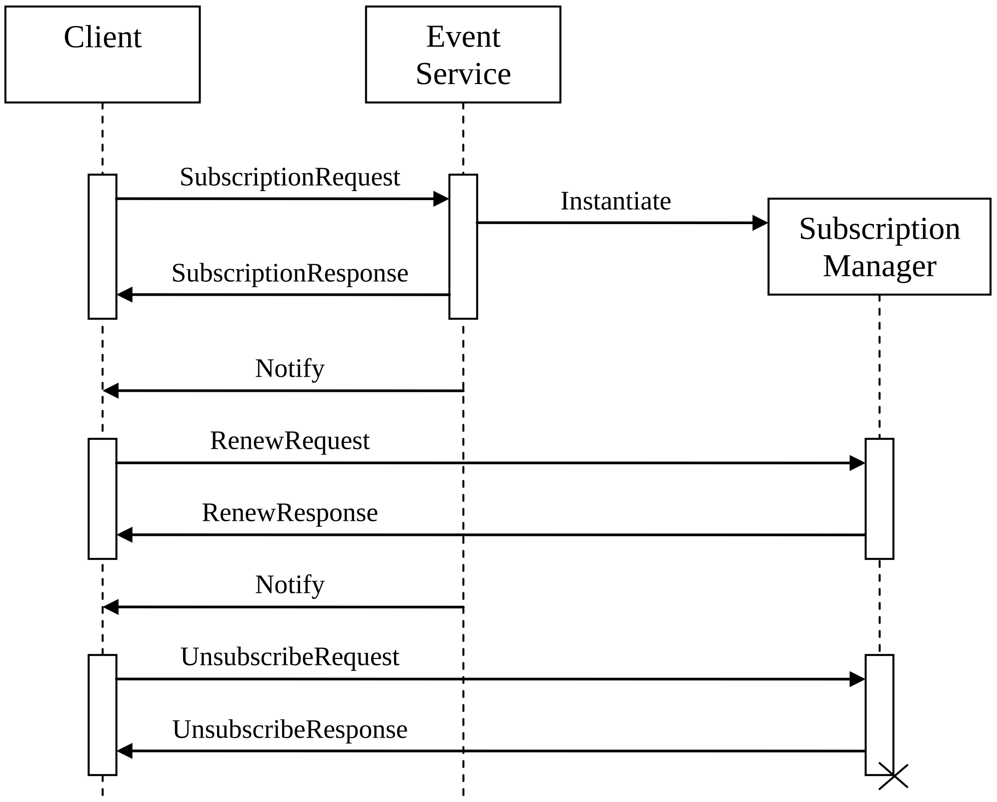


Figure 20: Sequence diagram for the Base Notification Interface

12.1.2 Requirements

This section details those interfaces of the [WS-BaseNotification] that an NVT must provide NVT.

An NVT MUST support the NotificationProducer Interface of the [WS-BaseNotification]. As a result, the NotificationProducer Resource Properties are OPTIONAL (see Section 12.5). The NVT MUST support TopicExpression and MessageContent filters with at least the dialects described in Sections 12.5.5 and 12.7.3. If the NVT does not accept the InitialTerminationTime of a subscription, it MUST provide a valid InitialTerminationTime within the Fault Message. The NVT MUST be able to provide notifications using the Notify wrapper of the [WS-BaseNotification] specification. The SubscriptionPolicy `wsnt:UseRaw` is OPTIONAL for the NVT. Although the

[WS-BaseNotification] has CurrentTime and TerminationTime as optional elements in a SubscribeResponse, an ONVIF compliant NVT MUST list them in SubscribeResponses. The NVT 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 an NVT is OPTIONAL.

The NVT MUST 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.

12.2 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 NVT for a PullPointSubscription with the CreatePullPointSubscriptionRequest message. The request contains a detailed description of the Subscription. The ConsumerReference MUST be omitted, in contrast to the subscription of the Basic Notification Interface (see Section 12.1),.

2) The NVT 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 MUST 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 21. The PullMessagesRequest contains Timeout and MessageLimit parameters.

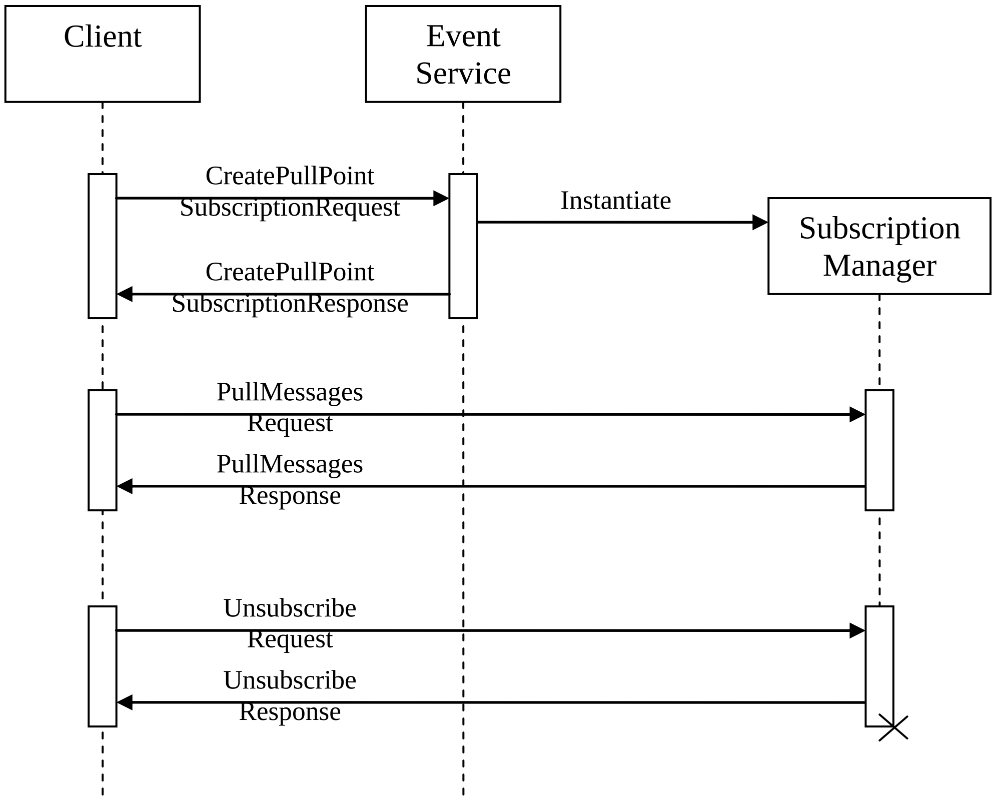


Figure 21: Sequence diagram for the Real-time Pull-Point Notification Interface.

4) The NVT MUST immediately respond with notifications that have been aggregated on behalf of the client. If there are no aggregated notifications, the NVT 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 MUST 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 an NVT internal default value. To inform the client about the updated termination time, the PullMessagesReponse MUST 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 NVT MUST support it for the PullPointSubscription.

12.2.1 Create pull point subscription

The NVT MUST provide the following CreatePullPointSubscription command:

Table 132: 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.

12.2.2 Pull messages

The NVT MUST provide the following PullMessages command for all SubscriptionManager endpoints returned by the CreatePullPointSubscription command.

Table 133: PullMessages command

PullMessages	Request-response
Message name	**Description**
PullMessagesRequest	*This message MUST 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 NVT. The Fault Message MUST contain the upper limits for both parameters.* xs:duration **MaxTimeout**[1][1] xs:int **MaxMessageLimit**[1][1]

Fault codes	Description
	No specific fault codes.

12.3 Notification Streaming Interface

Section 10.10 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, an NVC uses that profile to get an RTP stream that includes the configured notifications as metadata. The notification RTP streaming MUST be implemented by the NVTs.

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 12.5). 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` MUST NOT contain the SubscriptionReference element.

12.4 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 an NVC subscribes to a topic representing a certain property, the NVT MUST provide notifications informing the NVC of all objects with the requested property, which are alive at the time of the subscription. An NVC *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 12.6).

The property interface is defined in this specification 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 12.5 explains the structure of events and properties in detail.

12.4.1 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.

12.5 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 12.7).

Section 12.5.1 gives an overview of the information an NVC retrieves through notifications. Section 12.5.2 gives a detailed formatting of the Message payload, and Section 12.5.4 introduces a description language for the Message payload. Section 12.5.5 defines the grammar used in a subscription to filter notifications by their Message content.

12.5.1 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. The NVT MUST 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 NVT or a sub-service within the NVT 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 on properties (see Section 12.4), the client uses the NotificationProducer, the Topic, the Source Items and optional Key Items (see Section 12.5) in order to identify the property. These values MUST result in a unique identifier.

12.5.1.1 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”.

12.5.2 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"/>
      ...
```

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

```
    </xs:sequence>

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

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

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

The Items within the Message element are grouped into three categories: Source, Key, and Data. The Key group MUST 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 MUST be specified by the value attribute. The name of all Items MUST 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 12.7), where each topic can be augmented by a description of the message payload.

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

12.5.3 Property example, continued

The example in section 12.4.1 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.

12.5.4 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, the NVT 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:string" 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:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

The Name attribute of an Item MUST be unique within all Items independent from the group (Source, Key, Data) they are coming from. The IsProperty attribute MUST 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 MUST NOT be present. The Type attribute of a SimpleItemDescriptor MUST match the SimpleElement definition of an XML schema. Similarly, the Type attribute of an ElementItemDescriptor MUST match a global element declaration of an XML schema.

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

The location of all schema files used to describe Message payloads are listed in the GetEventPropertiesResponse message in Section 12.8.

12.5.4.1 Message Description Example

The following code is an example of a Message Description corresponding to the Property example of Section 12.5.3:

```
<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>
```

12.5.5 Message Content Filter

In the Subscription request, a client can filter notifications by TopicExpression (see Section 12.7.3) 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 NVT MUST 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"
```

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 ::= ‘//SimpleItem’ NodeTest | ‘//ElementItem’ NodeTest

[4] NodeTest ::= ‘[‘ AttrExpr ‘]’

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

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

[7] 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.

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

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

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

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

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

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

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

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

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

12.6 Synchronization Point

Properties, introduced in section 12.4, 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 12.5). The Synchronization Point is requested directly from the SubscriptionManager which was returned in either the SubscriptionResponse or in the CreatePullPointSubscriptionResponse. The property update it transmitted via the notification transportation of the notification interface. The following operation MUST be provided by all Subscription Manager Endpoints:

Table 134: 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 10.14.

12.7 Topic Structure

This specification extends the Topic framework defined in the [WS-Topics] specification. Section 12.7.1 describes an ONVIF Topic Namespace, which SHOULD be taken as a basis for vendor specific topics. The Appendix A shows typical examples for such extensions. Section 12.7.2 defines an interface to topic properties. This interface MUST be implemented by an NVT. Section 12.7.3 incorporates the Message Description Language defined in section 12.5.4 into the TopicSet structure. All topics grown from the ONVIF Topic Namespace describes the type of a topic according to section 12.7.3. Section 12.7.3 defines the Topic Expression Dialects which are supported by an NVT.

12.7.1 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 an NVT, 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 MUST use the Message Format as described in Section 12.5.2.

```
<wstop:TopicNamespace name="ONVIF"
 targetNamespace="http://www.onvif.org/ver10/topics" >
  <wstop:Topic name="Device" messageTypes="tt:Message"/>
  <wstop:Topic name="VideoSource" messageTypes="tt:Message"/>
  <wstop:Topic name="VideoEncoder" messageTypes="tt:Message"/>
  <wstop:Topic name="VideoAnalytics" messageTypes="tt:Message"/>
  <wstop:Topic name="RuleEngine" messageTypes="tt:Message"/>
  <wstop:Topic name="PTZController" messageTypes="tt:Message"/>
  <wstop:Topic name="AudioSource" messageTypes="tt:Message"/>
  <wstop:Topic name="AudioEncoder" messageTypes="tt:Message"/>
  <wstop:Topic name="UserAlarm" messageTypes="tt:Message"/>
  <wstop:Topic name="MediaControl" messageTypes="tt:Message"/>
</wstop:TopicNamespace>
```

12.7.2 Topic Type Information

The type information is added below a topic element by adding a MessageDescription element of type MessageDescriptionType defined in Section 12.5.4. 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>
```

12.7.3 Topic Filter

The NVT MUST 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 MUST 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 MUST be supported by an NVT, which extends the Concrete Topic Expressions by an “or” operation. Using this syntax makes it possible to select 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

[vc: If a namespace prefix is included in the RootTopic, it must correspond to a valid Topic Namespace definition and the local name must correspond to the name of a root Topic defined in that namespace.]

[6] ChildTopicExpression ::= '/' ChildTopicName

[7] ChildTopicName ::= QName | NCName

[vc: The NCName or local part of the QName must 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 MUST be used:

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

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>
```

12.8 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 MUST be implemented by the NVT in order to provide information about the FilterDialects, Schema files and topics supported by the NVT.

Table 135: 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!

The NVT MUST 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 the NVT (see Section 12.7.3):

```
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 NVT (see Section 12.5.5):

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

This specification does not require the support of any ProducerPropertiesDialect by the NVT.

The Message Content Description Language, introduced in Section 12.5.4, allows referencing of vendor-specific types. In order to ease the integration of such types into a client application, the GetEventPropertiesResponse MUST list all URI locations to schema files whose types are used in the description of notifications, with MessageContentSchemaLocation elements. This list MUST at least contain the URI of the ONVIF schema file.

12.9 SOAP Fault Messages

If an NVT encounters a failure while processing [WS-BaseNotification] messages from either a Client (NVC) or Subscriber Manager, then the NVT must generate a SOAP 1.2 fault message.

All SOAP 1.2 fault messages must be generated according to [WS-BaseNotification] and [WS-Topics] specifications.

12.10 Notification example

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

12.10.1 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>
      urn:#GetEventProperties
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:GetEventProperties>
    </tet:GetEventProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

12.10.2 GetEventPropertiesResponse

In this example, the NVT 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 NVT 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>
      urn:#GetEventProperties
    </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 wstop:topic="true">
          <tns1:LineDetector wstop:topic="true">
            <tns1:Crossed wstop:topic="true">
              <tt:MessageDescription>
                <tt:Source>
                  <tt:SimpleItem Name="VideoSourceConfigurationToken"
                                 Type="tt:ReferenceToken"/>
                  <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                                 Type="tt:ReferenceToken"/>
                </tt:Source>
                <tt:Data>
                  <tt:SimpleItem 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>
```

12.10.3 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 NVT. The client is reacts only to notifications that reference VideoAnalyticsConfiguration "2" and VideoSourceConfiguration "1". The Subscription SHOULD be terminated automatically after one minute if not explicitly renewed or messages are not pulled regularly.

```
<?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>
      urn:#CreatePullPointSubscription
    </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(//SimpleItem[@Name="VideoAnalyticsConfigurationToken"
                           and @Value="2"] ) and
          boolean(//SimpleItem[@Name="VideoSourceConfigurationToken"
                           and @Value="1"] )
        </wsnt:MessageContent>
      </tet:Filter>
      <tet:InitialTerminationTime>
        P1M
      </tet:InitialTerminationTime>
    </tet:CreatePullPointSubscription>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

12.10.4 CreatePullPointSubscriptionResponse

When the NVT 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 NVT 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>
      urn:#CreatePullPointSubscription
    </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>
```

12.10.5 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>
      urn:#PullMessages
    </wsa:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tet:PullMessages>
      <tet:Timeout>
      P5S
      </tet:Timeout>
      <tet:MessageLimit>
        2
      </tet:MessageLimit>
    </tet:PullMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

12.10.6 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>
```

```
        urn:#PullMessages
      </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>
```

12.10.7 UnsubscribeRequest

A client SHOULD explicitly terminate a subscription with an UnsubscribeRequest that the NVT 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>
                     urn:#Unsubscribe
              </wsa:Action>
       </SOAP-ENV:Header>
       <SOAP-ENV:Body>
              <wsnt:Unsubscribe/>
       </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

12.10.8 UnsubscribeResponse

The Subscription Endpoint is no longer available once the NVT 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>
                     urn:#Unsubscribe
              </wsa:Action>
       </SOAP-ENV:Header>
       <SOAP-ENV:Body>
              <wsnt:UnsubscribeResponse/>
       </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

12.11 Service specific fault codes

The event service does not define any service specific faults except those defined in [WS-BaseNotification].

13 PTZ control

The PTZ service provides operations used to perform NVT pan, tilt and zoom control. An NVT with pan, tilt and zoom capability MUST support the PTZ service. Similarly, an NVT with either just zoom or pan *and* tilt capability MUST support the PTZ service. The PTZ service is defined in [ONVIF PTZ WSDL]. The mandatory operations are indicated under the command descriptions.

The PTZ service 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 service 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 4.6.1) *and* all PTZ control operations are done by referring to a particular media profile.

The PTZ service does not provide operations to create or manipulate PTZ Nodes. For each available PTZ Node, the NVT MUST 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 NVT MUST 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.

13.1 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 13.8 defines a standard set of coordinate systems. A PTZ Node MUST 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*, that is scheduled for release after the release of this document. 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 MUST fail when an absolute Pan/Tilt position is specified. The corresponding command section describes those spaces that are required for a specific move command.

13.2 PTZ Node

A PTZ-capable NVT 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 MUST 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 MUST specify its allowed range.
- MaximumNumberOfPresets – All preset operations MUST 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 MUST be available for this PTZ Node.
- AuxiliaryCommands – A list of supported Auxiliary commands. If the list is not empty, the Auxiliary Operations MUST be available for this PTZ Node.

13.2.1 GetNodes

A PTZ capable NVT MUST implement this operation and return all PTZ Nodes available on the NVT.

Table 136: 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 NVT.* tt:PTZNode **PTZNodes**[0][unbounded]
Fault codes	**Description**
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported by the NVT.*

13.2.2 GetNode

A PTZ capable NVT MUST implement the GetNode operation and return the properties of the requested PTZ Node, if it exists. Otherwise, the NVT MUST respond with an appropriate Fault message.

Table 137: GetNode command

GetNode	Request-Response
Message name	**Description**
GetNodesRequest	*This message contains a reference by name or token to the requested PTZNode* tt:ReferenceToken **NodeToken**[0][1] tt:Name **NodeName**[0][1]
GetNodesResponse	*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 NVT*

13.3 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 MUST specify one Absolute Pan/Tilt Position Space as default.

- DefaultRelativePanTiltTranslationSpace – If the PTZ Node supports relative Pan/Tilt movements, it MUST specify one RelativePan/Tilt Translation Space as default.

- DefaultContinuousPanTiltVelocitySpace – If the PTZ Node supports continuous Pan/Tilt movements, it MUST specify one Continuous Pan/Tilt Velocity Space as default.

- DefaultPanTiltSpeedSpace – If the PTZ Node supports absolute or relative movements, it MUST specify one Pan/Tilt Speed Space as default.

- DefaultAbsoluteZoomPositionSpace – If the PTZ Node supports absolute zoom movements, it MUST specify one Absolute Zoom Position Space as default.

- DefaultRelativeZoomTranslationSpace – If the PTZ Node supports relative zoom movements, it MUST specify one Relative Zoom Translation Space as default.

- DefaultContinuousZoomVelocitySpace – If the PTZ Node supports continuous zoom movements, it MUST specify one Continuous Zoom Velocity Space as default.

- DefaultZoomSpeedSpace – If the PTZ Node supports either absolute or relative movements, it MUST specify one Zoom Speed Space as default.

- DefaultPTZSpeed – If the PTZ Node supports absolute or relative PTZ movements, it MUST specify corresponding default Pan/Tilt and Zoom speeds.

- DefaultPTZTimeout – If the PTZ Node supports continuous movements, it MUST 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 MUST 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 MUST 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 must 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 must 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.

13.3.1 GetConfigurations

A PTZ-capable NVT MUST return all available PTZConfigurations through the GetConfigurations operation.

Table 138: GetConfigurations command

GetConfigurations	Request-Response
Message name	**Description**
GetConfigurations	*This is an empty message.*
GetConfigurationsResponse	*The response contains all existing PTZConfigurations on the NVT.* tt:PTZConfiguration **PTZConfiguration** [0][unbounded]
Fault codes	**Description**
	No command specific faults!

13.3.2 GetConfiguration

A PTZ capable NVT MUST return the requested PTZ Configuration, if it exists, through the GetConfiguration operation.

Table 139: GetConfiguration command

GetConfiguration	Request-Response
Message name	**Description**
GetConfigurationRequest	*This message contains a reference to the requested PTZConfiguration.* tt:ReferenceToken **PTZConfigurationToken**[1][1]
GetConfigurationResponse	*The response contains the requested PTZConfiguration* tt:PTZConfiguration **PTZConfiguration** [1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:NoConfig	*The requested configuration does not exist.*
env:Receiver ter:ActionNotSupported ter:PTZNotSupported	*PTZ is not supported by the NVT.*

13.3.3 GetConfigurationOptions

A PTZ capable NVT MUST 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 13.1. If the PTZ Node supports continuous movements, it MUST return a Timeout Range within which Timeouts are accepted by the PTZ Node.

Table 140: 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.*

13.3.4 SetConfiguration

A PTZ capable NVT MUST implement the SetConfiguration operation. The ForcePersistence flag indicates if the changes remain after reboot of the NVT.

Table 141: SetConfiguration command

SetConfiguration	Request-Response
Message name	**Description**
SetConfigurationRequest	*The **Configuration** element contains the modified PTZ configuration. The configuration must 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: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.*

13.4 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.

13.4.1 AbsoluteMove

If a PTZ Node supports absolute Pan/Tilt or absolute Zoom movements, it MUST 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 MUST 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. An NVT 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 MUST be Speed Spaces supported by the PTZ Node.

The operation MUST fail if the requested absolute position is not reachable.

Table 142: 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 ter:InvalidSpeed	*The requested speed is out of bounds.*

13.4.2 RelativeMove

If a PTZ Node supports relative Pan/Tilt or relative Zoom movements, then it MUST 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 MUST 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. An NVT 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 MUST 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 MUST 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 MUST move to a reachable position along the border of valid positions.

Table 143: 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.*

13.4.3 ContinuousMove

A PTZ-capable NVT MUST support continuous movements. The Velocity argument of this command specifies a signed speed value for the Pan, Tilt and Zoom parameters to which the PTZ unit should move. 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 MUST 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. An NVT 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 MUST 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 13.4.4.

If the requested velocity leads to absolute positions which cannot be reached, the PTZ Node MUST 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 144: 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:PTZVector **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.*

13.4.4 Stop

A PTZ-capable NVT MUST support the Stop operation. If no Stop arguments are present, this command stops all ongoing pan, tilt and zoom movements. The Stop operation MAY be filtered to stop a specific movement by setting the corresponding stop argument.

Table 145: 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 **PanTiltStop**[0][1] xs:boolean **ZoomStop**[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.*

13.4.5 GetStatus

A PTZ-capable NVT MUST be able to report its PTZ status through the GetStatus command. The PTZ Status contains the following information:

- Position – Specifies the absolute position of the PTZ unit together with the Space references. The default absolute spaces of the corresponding PTZ configuration MUST be referenced within the Position element.
- MoveStatus – Indicates if the Pan/Tilt/Zoom device unit is currently moving, idle or in an unknown state.
- Error – States a current PTZ error.
- UTC Time – Specifies the UTC time when this status was generated.

Table 146: 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.*

13.5 Preset operations

This section describes operations that manage the presets of a PTZ Node. These operations MUST be implemented for PTZ Nodes supporting presets. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

13.5.1 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 NVT MAY internally save additional states such as imaging properties in the PTZ Preset which then should be recalled in the GotoPreset operation.

Table 147: 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.*

13.5.2 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 MUST be specified. If the PTZ Node supports absolute zoom position spaces, the zoom position MUST be specified.

Table 148: 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.*

13.5.3 GotoPreset

The GotoPreset operation recalls a previously set Preset. If the speed parameter is omitted, the default speed of the corresponding PTZ Configuration MUST 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 149: 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.*

13.5.4 RemovePreset

The RemovePreset operation removes a previously set Preset.

Table 150: 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.*

13.6 Home Position operations

This section describes operations used to manage the Home Position of a PTZ Node. These operations MUST 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.

13.6.1 GotoHomePosition

This operation moves the dome to its home position. The command is non-blocking and can be interrupted by other move commands.

Table 151: 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]
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.*

13.6.2 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 MUST return with a failure if the “home” position is fixed and cannot be overwritten. If the SetHomePosition is successful, it MUST be possible to recall the Home Position with the GotoHomePosition command.

Table 152: 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.*

13.7 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 MUST be implemented for PTZ nodes indicating auxiliary commands in the node properties. All operations require a *ProfileToken* referencing a Media Profile including a PTZConfiguration.

13.7.1 SendAuxiliaryCommand

This operation is used to call an auxiliary operation on the NVT. 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 MUST be implemented when the PTZ Node supports auxiliary commands.

Table 153: 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 **AuxiliaryCommand**[1][1]
SendAuxiliaryCommandResponse	*The response contains the auxiliary response.* tt:AuxiliaryData **AuxiliaryCommandResponse**[0][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.*

13.8 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*, which will be released by ONVIF at a later date.

13.8.1 Absolute Position Spaces

13.8.1.1 Generic Pan/Tilt Position Space

The Generic Pan/Tilt Position Space MUST 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>
```

13.8.1.2 Generic Zoom Position Space

The Generic Zoom Position Space MUST 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>
```

13.8.2 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.

13.8.2.1 Generic Pan/Tilt Translation Space

The Generic Pan/Tilt Translation Space MUST 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>
```

13.8.2.2 Generic Zoom Translation Space

The Generic Zoom Translation Space MUST 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>
```

13.8.3 Continuous Velocity Spaces

The Continuous Velocity Spaces are used to continuously move the PTZ unit in a certain direction.

13.8.3.1 Generic Pan/Tilt Velocity Space

The Generic Pan/Tilt Velocity Space MUST 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>
```

13.8.3.2 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>
```

13.8.4 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.

13.8.4.1 Generic Pan/Tilt Speed Space

The Generic Pan/Tilt Speed Space MUST 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>
```

13.8.4.2 Generic Zoom Speed Space

The Generic Zoom Speed Space MUST 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>
```

13.9 Service specific fault codes

Table 154 below lists the PTZ service specific fault codes. Each command can generate a generic fault, see Table 5.

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 154: PTZspecific fault codes

Fault Code	**Parent Subcode**	**Fault Reason**	**Description**
	Subcode		
env:Receiver	ter:Action	Preset cannot be set	Preset cannot be set while the PTZ unit is moving.
	ter:MovingPTZ		
env:Receiver	ter:Action	Number of presets limit reached	Maximum number of Presets reached.
	ter:TooManyPresets		
env:Receiver	ter:ActionNotSupported	PTZ not supported	PTZ is not supported by the NVT.
	ter:PTZNotSupported		
env:Receiver	ter:Action	Token already exist	The requested name or token already exist for another preset.
	ter:PresetExist		
env:Receiver	ter:Action	No PTZ status available	No PTZ status is available in the requested Media Profile.
	ter:NoStatus		
env:Receiver	ter:Action	Conflict when using new settings	The new settings result in an inconsistent configuration.
	ter:ConfigurationConflict		
env:Receiver	ter:Action	Home position cannot be overwritten	The home position is fixed and cannot be overwritten.
	ter:CannotOverwriteHome		
env:Sender	ter:InvalidArgVal	No such PTZ node	No such PTZ Node on the NVT
	ter:NoEntity		
env:Sender	ter:InvalidArgVal	No such configuration	No such configuration exist.
	ter:NoConfig		
env:Sender	ter:InvalidArgVal	The paramters could not be set	The configuration parameters are not possible to set.
	ter:ConfigModify		

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:Sender	ter:InvalidArgVal 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.

14 Video analytics

Section 4.10 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 Section 12. Section 14.1 introduces the XML-based scene description, which can be streamed as metadata to NVCs via RTP (see Section 11.1.2.1.1 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 10). The analytics service allows more fine-grained configuration of individual rules and individual analytics modules (see Sections 14.2 and 14.3).

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 Section 10.9). The NVT MUST ensure that a corresponding analytics engine starts operation when an NVC subscribes directly or indirectly for events produced by the analytics or rule engine or when an NVC requests the corresponding scene description stream.

14.1 Scene Description Interface

14.1.1 Overview

This specification defines the XML schema that MUST be used to encode Scene Descriptions by an NVT. 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 MUST be referenced in the Profile. The Video Analytics Engine then processes frames according to the referenced VideoSourceConfiguration.

14.1.2 Frame Related Content

The input of the Video Analytics Engine are 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 14.1.2.1 and 14.1.2.2. The appearance and behaviour of tracked objects is discussed in section 14.1.3.1. Interactions between objects like splits and merges are described in section 14.1.3.2.

A PTZ device can put information about the Pan, Tilt and Zoom at the beginning of a frame, allowing an NVC 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 14.1.2.1 describes how frames processed by the Video Analytics Algorithm are referenced within the Video Analytics stream.

14.1.2.1 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 MUST enable an NVC to map the Frame Node exactly to one video frame. For example,. the RTP timestamp of the corresponding encoded video frame MUST result in the same UTC timestamp after conversion. The synchronization between Video and Metadata streams is further described in the Real-time Viewing Section 11.1.2.2.1.

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.

14.1.2.2 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 must be specified. These positions must relate to a Coordinate System. The Default Coordinate System is shown in Figure 22. It maps onto the rectangle selected in the VideoSourceConfiguration of the corresponding Profile.

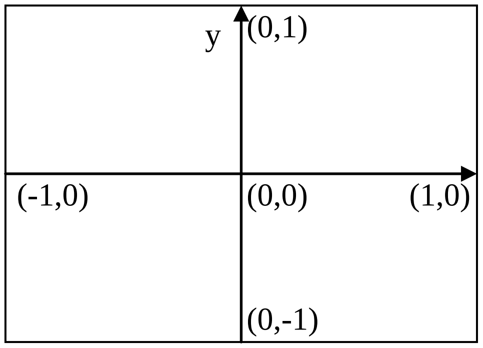


Figure 22: Default frame coordinate system

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

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

v given with respect to the coordinate system is transformed into the corresponding vector ***w*** of the default coordinate system by: $\begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} v_x \cdot s_x \\ v_y \cdot s_y \end{pmatrix}$.

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>
```

14.1.3 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 NVT regularly sends the Scene Description even if it is empty, in order to indicate that the analytics engine is operational. The NVT MUST 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.

14.1.3.1 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>

<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 14.1.3.3). The definition of colour and object class can be found in Appendix A.2.

This specification defines two standard Behaviours for Objects. When an Object stops moving, it can be marked as either Removed or Idle. These behaviours MUST 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">
```

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

```
    <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>
```

14.1.3.2 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 MUST 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>
```

14.1.3.3 Shape descriptor

Shape information MUST 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 MUST 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 MUST 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 must 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 MUST 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>
```

14.2 Rule interface

The Video Analytics Configuration consists of two parts (see Section 10.9). 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 14.2.1 to communicate the configuration of Rules. Section 14.2.2 specifies a language to describe the configuration of a specific Rule type. Section 14.2.3 defines two standard Rules that SHOULD be supported by an NVT implementing a Rule Engine. Section 14.2.4 introduces operations to manage rules. If the NVT supports a Rule Engine as defined by ONVIF, it MUST implement the complete Rule Interface.

14.2.1 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 12). The Name attribute of each Item MUST 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:VideoAnalyticsConfig>
  <tt:AnalyticsEngineConfig>
    ...
  </tt:AnalyticsEngineConfig>
  <tt:RuleEngineConfig>
    <tt:Rule Name="MyLineDetector" Type="tt:LineDetector">
      <tt:Parameters>
        <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:SimpleItem Name="Direction" Value="Any">
      </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:RuleEngineConfig>
</tt:VideoAnalyticsConfig>
```

14.2.2 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 are 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 MUST reference a SimpleType schema definition. In case of the ElementItemDescription, the Type attribute MUST 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 12. Additionally, the MessageDescription MUST contain a ParentTopic element naming the Topic a client has to subscribe to in order to receive this specific output. The Topic MUST be specified as a Concrete Topic Expression.

Section 14.2.3 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="MessageDescription" 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"/>
```

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

```
        <xs:attribute name="Type" type="xs:string" 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:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

14.2.3 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">
```

14.2.3.1 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.

The LineDetector resembles the following code using the Rule Description Language, detailed in the previous section:

```
<tt:RuleDescription Name="tt:LineDetector">
  <tt:Parameters>
    <tt:ElementItemDescription Name="Segments" Type="tt:Polyline"/>
    <tt:SimpleItemDescription Name="Direction" Type="tt:Direction"/>
  </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.

14.2.3.2 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, an NVC can derive the Entering and the Leaving parameters of the detector. An NVC 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.

14.2.4 Operations on rules

If the NVT supports a Rule Engine as defined by ONVIF, then it MUST 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 MUST fail.

14.2.4.1 Get Supported rules

The NVT MUST 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 14.2.2. 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 MUST be explicitly listed.

Table 155: 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 is either empty or contains a list of supported Rule Descriptions.*

	xs:anyURL **RuleContentSchemaLocation** [0][unbounded] tt:ConfigDescription **RuleDescription** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

14.2.4.2 Get Rules

The following operation retrieves the currently installed Rules:

Table 156: 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.*

14.2.4.3 Create rules

The following operation adds Rules to a VideoAnalyticsConfiguration. If all rules can not be created as requested, the NVT responds with a fault message.

Table 157: 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	*The suggested rules configuration is not valid on the NVT.*

ter:InvalidRule	
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 NVT to add the rules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The NVT cannot create the rules without creating a conflicting configuration.*

14.2.4.4 Modify Rules

The following operation modifies Multiple Rules. If all rules can not be modified as requested, the NVT responds with a fault message.

Table 158: 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 NVT.*
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 NVT to add the rules to the configuration.*
env:Receiver ter:Action ter:ConflictingConfig	*The NVT cannot delete the rules without creating a conflicting configuration.*

14.2.4.5 Delete Rules

The following operation deletes Multiple Rules.. If all rules can not be deleted as requested, the NVT responds with a fault message.

Table 159: 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 NVT cannot delete the rules without creating a conflicting configuration.*
env:Sender ter:InvalidArgs ter:RuleNotExistent	*The rule name or names do not exist.*

14.3 Analytics Modules Interface

The Video Analytics Configuration consists of two parts (see Section 10.9). The first part configures the Video Analytics Engine creating the SceneDescription. The second part configures the Rule Engine. Section 14.3.1 defines an XML structure for the first part that communicates the configuration of Analytics Modules. Section 14.3.2 defines the language that describes the configuration of a specific Analytics Module. Section 14.3.3 defines the operations required by the Analytics Modules Interface. If the NVT supports an Analytics Engine as defined by ONVIF, it MUST implement the complete Analytics Modules Interface.

14.3.1 Analytics module configuration

The Analytics Module Configuration is identical to the Rule Configuration, described in Section 14.2.1. 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" Type="0.01"/>
        <tt:SimpleItem Name="MinObjectHeight" Type="0.01"/>
        <tt:SimpleItem Name="MaxObjectWidth" Type="0.5"/>
        <tt:SimpleItem Name="MaxObjectHeight" Type="0.5"/>
      </tt:Parameters>
    </tt:AnalyticsModule>
  </tt:AnalyticsEngineConfig>
  <tt:RuleEngineConfig>
    ...
  </tt:RuleEngineConfig>
</tt:VideoAnalyticsConfig>
```

14.3.2 Analytics Module Description Language

The Analytics Module reuses the Rule Description Language, described in Section 14.2.2. The following AnalyticsModuleDescription element replaces the RuleDescription element:

```
<xs:element name="AnalyticsModuleDescription"
```

```
            type="tt:ConfigDescription"/>
```

Similar to rules, Analytics Modules produce Events and MUST 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:RuleDescription>
```

14.3.3 Operations on Analytics Modules

If the NVT supports an analytics engine as defined by ONVIF, it MUST support the subsequent operations to manage analytics modules. The Create/Delete/Modify operations MUST be atomic, all modifications can be processed or the complete operation MUST fail.

14.3.3.1 GetSupportedAnalyticsModules

The NVT 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 14.2.2. 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 ONVIF schema file MUST be explicitly listed.

Table 160: 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 is either empty or contains a list of supported Analytics Module Descriptions.* xs:anyURI **AnalyticsModuleContentSchemaLocation** [0][unbounded]

	tt:ConfigDescription **AnalyticsModuleDescription** [0][unbounded]
Fault codes	**Description**
env:Sender ter:InvalidArgs ter:NoConfig	*VideoAnalyticsConfiguration does not exist.*

14.3.3.2 GetAnalytics Modules

The following operation retrieves the currently installed Analytics Modules:

Table 161: 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.*

14.3.3.3 CreateAnalytics Modules

The following operation adds Analytics Modules to a VideoAnalyticsConfiguration. If all Analytics Modules can not be created as requested, the NVT responds with a fault message.

Table 162: 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 NVT to add the analytics modules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The NVT 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 NVT.*

14.3.3.4 ModifyAnalytics Modules

The following operation modifies multiple Analytics Modules. If all analytics modules can not be modified as requested, the NVT respond with a fault message.

Table 163: 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:ReferenceToke **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 NVT to add the analytics modules to the configuration.*
env:Receiver ter:Action ter:ConfigurationConflict	*The NVT 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 NVT.*

14.3.3.5 DeleteAnalytics Modules

The following operation deletes multiple Analytics Modules. If all analytics modules can not be deleted as requested, the NVT responds with a fault message.

Table 164: 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 NVT 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.*

14.4 Service-specific fault codes

Table 165 below lists the analytics service-specific fault codes. Each command can also generate a generic fault. Refer to Table 5.

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 165: 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 NVT to add the rules to the configuration.
env:Receiver	ter:Action ter:TooManyModules	No more space available.	There is not enough space in the NVT to add the analytics 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	The rule is invalid.	The suggested rule configuration is not valid on

	ter:InvalidRule		the NVT.
env:Sender	ter:InvalidArgVal	The module is invalid	The suggested analytics module configuration is not valid on the NVT.
	ter:InvalidModule		
env:Sender	ter:InvalidArgVal	The rule exists	The same rule name exists already in the configuration.
	ter:RuleAlreadyExistent		
env:Sender	ter:InvalidArgs	The rule does not exist	The rule name or names do not exist.
	ter:RuleNotExistent		
env:Sender	ter:InvalidArgs	The name exists	The same analytics module name exists already in the configuration.
	ter:NameAlreadyExistent		
env:Sender	ter:InvalidArgs	The name does not exist	The analytics module with the requested name does not exist.
	ter:NameNotExistent		

15 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 compliant 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

15.1 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.

An NVT compliant with this specification MUST support TLS 1.0 [RFC 2246] and related specifications. The NVT SHOULD support TLS 1.1 [RFC 4346]. The NVT MAY support TLS 1.2 [RFC 5246].

An NVT MUST support TLS 1.0 for protection of all of the ONVIF services it provides. An NVT also MUST support TLS 1.0 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 1.0 and other relevant specifications that can be used with TLS 1.0.

The NVC SHOULD support TLS 1.0 [RFC 2246] and TLS 1.1 [RFC 4346]. The NVC MAY support TLS 1.2 [RFC 5246].

15.1.1 Supported cipher suites

The NVT MUST support all of the following cipher suites [RFC 2246], [RFC 3268]:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

If the NVC supports TLS 1.0, then it MUST support the following cipher suites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

15.1.2 Server authentication

The NVT MUST support server authentication using TLS 1.0. The NVT MUST support processing of X.509 server certificates. The RSA key length MUST be at least 1024 bits.

The NVC SHOULD support server authentication using TLS 1.0.

This specification does not provide a full server certificate generation and Certificate Authority (CA) model. However, device management commands for NVT 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.

15.1.3 Client authentication

The NVT MUST support client authentication. Client authentication can be enabled/disabled with a device management command as described in Section 8.4

The server MUST include the RSA certificate type (rsa_sign, for example) in the certificate request [RFC 2246] for client certificates, and MUST support verification of the RSA client certificate and signature.

The NVC SHOULD support client authentication. If the client authentication is supported, the client MUST support RSA client certificate and signature and MUST 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.

15.2 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.

A Notification topics (informative)

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 12.

A.1.1 Profile

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="ProfileToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:Profile"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.2 VideoSourceConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="VideoSourceConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:VideoSourceConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.3 AudioSourceConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="AudioSourceConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:AudioSourceConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.4 VideoEncoderConfiguration

```
<tt:MessageDescription iIsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="VideoEncoderConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:VideoEncoderConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.5 AudioEncoderConfiguration

```
<tt:MessageDescription iIsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="AudioEncoderConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:AudioEncoderConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.6 VideoAnalyticsConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="VideoAnalyticsConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:VideoAnalyticsConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.7 PTZConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="PTZConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem Name="Config"
                 Type="tt:PTZConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

A.1.8 MetaDataConfiguration

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItem Name="MetaDataConfigurationToken"
                Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItem 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:SimpleItem Name="RelayToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItem 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 NVT 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:SimpleItem Name="PTZConfigurationToken"
                   Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItem Name="PresetToken" Type="tt:ReferenceToken"/>
    <tt:SimpleItem Name="PresetName"  Type="tt:Name"/>
  </tt:Data>
</tt:MessageDescription>
```

B Scene descriptions (informative)

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:ColorType"/>
          <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 MUST 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>
```