# ONVIF®

# Media Configuration Device Test Specification

Version 24.06

June 2024

# REVISION HISTORY

| Vers. | Date | Description |
|---|---|---|
| 1.02.4 | 29th/Jul, 2011 | First issue of Media Test Specification |
| 11.12 | 22nd/Dec, 2011 | New version numbering scheme has been applied.<br><br>Requirement level terms have been removed.<br><br>Audio outputs configuration test cases have been added.<br><br>Audio & Video Multicast streaming test cases have been added.<br><br>Start & Stop Multicast Streaming test case has been added. |
| 12.06 | 6th/Jun, 2012 | Media Streaming IPv6 Multicast test cases have been added.<br><br>Media Streaming with line breaks in base64 encoding test case has been added.<br><br>Media Service Capabilities test cases have been added.<br><br>Start & Stop Audio Multicast Streaming test cases have been added. |
| 12.12 | 20th/Dec, 2012 | Following test cases was added:<br><br>GET GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES AND GET VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY<br><br>VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODINGS<br><br>AUDIO ENCODER CONFIGURATION OPTIONS AND AUDIO ENCODER CONFIGURATIONS CONSISTENCY (BITRATE AND SAMPLERATE)<br><br>SET AUDIO ENCODER CONFIGURATION<br><br>AUDIO ENCODER CONFIGURATION - MULTICAST PORT (IPv4)<br><br>AUDIO ENCODER CONFIGURATION - MULTICAST ADDRESS (IPv4)<br><br>VIDEO ENCODER CONFIGURATION - MULTICAST PORT (IPv4)<br><br>VIDEO ENCODER CONFIGURATION - MULTICAST ADDRESS (IPv4)<br><br>VIDEO ENCODER CONFIGURATION - MULTICAST ADDRESS AND PORT IN RTSP Setup (IPv4)<br><br>VIDEO ENCODER CONFIGURATION - MULTICAST ADDRESS AND PORT IN RTSP Setup (IPv6)<br><br>VIDEO ENCODER CONFIGURATION – JPEG RESOLUTION<br><br>VIDEO ENCODER CONFIGURATION – MPEG4 RESOLUTION<br><br>VIDEO ENCODER CONFIGURATION – H.264 RESOLUTION |

| | | |
|---|---|---|
| | | MEDIA STREAMING – GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES (RTP-Unicast/UDP)<br><br>MEDIA STREAMING – GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES (RTP-Multicast/UDP)<br><br>MEDIA STREAMING – GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES (RTP-Unicast/RTSP/HTTP/TCP)<br><br>MEDIA STREAMING – GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES (RTP/RTSP/TCP)<br><br>MEDIA STREAMING – GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES (MIX OF TRANSPORT TYPES)<br><br>AUDIO STREAMING – G.711 (RTP-Unicast/UDP)<br><br>AUDIO STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP)<br><br>AUDIO STREAMING – G.711 (RTP/RTSP/TCP)<br><br>AUDIO STREAMING – G.726 (RTP-Unicast/UDP)<br><br>AUDIO STREAMING – G.726 (RTP-Unicast/RTSP/HTTP/TCP)<br><br>AUDIO STREAMING – G.726 (RTP/RTSP/TCP)<br><br>AUDIO STREAMING – AAC (RTP-Unicast/UDP)<br><br>AUDIO STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP)<br><br>AUDIO STREAMING – AAC (RTP/RTSP/TCP)<br><br>MEDIA STREAMING – G.711 (RTP-Multicast/UDP, IPv4)<br><br>MEDIA STREAMING – G.711 (RTP-Multicast/UDP, IPv6)<br><br>MEDIA STREAMING – G.726 (RTP-Multicast/UDP, IPv4)<br><br>MEDIA STREAMING – G.726 (RTP-Multicast/UDP, IPv6)<br><br>MEDIA STREAMING – AAC (RTP-Multicast/UDP, IPv4)<br><br>MEDIA STREAMING – AAC (RTP-Multicast/UDP, IPv6)<br><br>START AND STOP MULTICAST STREAMING – G.711 (IPv4, ONLY AUDIO PROFILE)<br><br>START AND STOP MULTICAST STREAMING – G726 (IPv4, ONLY AUDIO PROFILE)<br><br>START AND STOP MULTICAST STREAMING – AAC (IPv4, ONLY AUDIO PROFILE)<br><br>VIDEO AND AUDIO ENCODER CONFIGURATION – DIFFERENT PORTS<br><br>VIDEO AND AUDIO ENCODER CONFIGURATION – DIFFERENT ADDRESS<br><br>MEDIA STREAMING – JPEG (VALIDATING RTP HEADER EXTENSION) |

| | | |
|---|---|---|
| 13.06 | June, 2013 | Real Time Streaming Test Cases section and related parts was moved to separate document Real Time Streaming Test Specification.<br><br>Current document name was changed from Media Test Specification to Media Configuration Test Specification.<br><br>The following test cases were added or updated with ID change:<br><br>GET GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES AND GET VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY<br><br>VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODINGS<br><br>DYNAMIC MEDIA PROFILE CONFIGURATION<br><br>VIDEO SOURCE CONFIGURATION<br><br>JPEG VIDEO ENCODER CONFIGURATION<br><br>MPEG4 VIDEO ENCODER CONFIGURATION<br><br>H.264 VIDEO ENCODER CONFIGURATION<br><br>VIDEO SOURCE CONFIGURATION USE COUNT (ADD SAME VIDEO SOURCE CONFIGURATION TO PROFILE TWICE)<br><br>VIDEO SOURCE CONFIGURATION USE COUNT (ADD DIFFERENT VIDEO SOURCE CONFIGURATIONS IN PROFILE)<br><br>VIDEO SOURCE CONFIGURATION USE COUNT (REMOVE VIDEO SOURCE CONFIGURATION)<br><br>VIDEO SOURCE CONFIGURATION USE COUNT (DELETION PROFILE WITH VIDEO SOURCE CONFIGURATION)<br><br>VIDEO SOURCE CONFIGURATION USE COUNT (SET VIDEO SOURCE CONFIGURATION)<br><br>VIDEO ENCODER CONFIGURATION USE COUNT (ADD SAME VIDEO ENCODER CONFIGURATION TO PROFILE TWICE)<br><br>VIDEO ENCODER CONFIGURATION USE COUNT (ADD DIFFERENT VIDEO ENCODER CONFIGURATIONS IN PROFILE)<br><br>VIDEO ENCODER CONFIGURATION USE COUNT (REMOVE VIDEO ENCODER CONFIGURATION)<br><br>VIDEO ENCODER CONFIGURATION USE COUNT (PROFILE DELETION WITH VIDEO ENCODER CONFIGURATION)<br><br>VIDEO ENCODER CONFIGURATION USE COUNT (SET VIDEO ENCODER CONFIGURATION)<br><br>VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODINGS (ALL VIDEO ENCODER CONFIGURATIONS)<br><br>AUDIO SOURCE CONFIGURATION<br><br>AUDIO ENCODER CONFIGURATION<br><br>G.711 AUDIO ENCODER CONFIGURATION |

| | | |
|---|---|---|
| | | G.726 AUDIO ENCODER CONFIGURATION<br><br>AAC AUDIO ENCODER CONFIGURATION<br><br>GET AUDIO SOURCE CONFIGURATION – INVALID CONFIGURATIONTOKEN<br><br>GET AUDIO SOURCE CONFIGURATION OPTIONS – INVALID PROFILETOKEN<br><br>GET AUDIO SOURCE CONFIGURATION OPTIONS – INVALID CONFIGURATION TOKEN<br><br>SET AUDIO SOURCE CONFIGURATION – INVALID TOKEN<br><br>SET AUDIO ENCODER CONFIGURATION<br><br>AUDIO SOURCE CONFIGURATION USE COUNT (ADD SAME AUDIO SOURCE CONFIGURATION TO PROFILE TWICE)<br><br>AUDIO SOURCE CONFIGURATION USE COUNT (ADD DIFFERENT AUDIO SOURCE CONFIGURATIONS IN PROFILE)<br><br>AUDIO SOURCE CONFIGURATION USE COUNT (REMOVE AUDIO SOURCE CONFIGURATION)<br><br>AUDIO SOURCE CONFIGURATION USE COUNT (PROFILE DELETION WITH AUDIO SOURCE CONFIGURATION)<br><br>AUDIO SOURCE CONFIGURATION USE COUNT (SET AUDIO SOURCE CONFIGURATION)<br><br>AUDIO ENCODER CONFIGURATION USE COUNT (ADD SAME AUDIO ENCODER CONFIGURATION TO PROFILE TWICE)<br><br>AUDIO ENCODER CONFIGURATION USE COUNT (ADD DIFFERENT AUDIO ENCODER CONFIGURATIONS IN PROFILE)<br><br>AUDIO ENCODER CONFIGURATION USE COUNT (REMOVE AUDIO ENCODER CONFIGURATION)<br><br>AUDIO ENCODER CONFIGURATION USE COUNT (DELETION PROFILE WITH AUDIO ENCODER CONFIGURATION)<br><br>AUDIO ENCODER CONFIGURATION USE COUNT (SET AUDIO ENCODER CONFIGURATION)<br><br>SET AUDIO OUTPUT CONFIGURATION<br><br>SET AUDIO OUTPUT CONFIGURATION – INVALID CONFIGURATION<br><br>SET AUDIO OUTPUT CONFIGURATION – INVALID OUTPUTTOKEN<br><br>PTZ CONFIGURATION<br><br>METADATA CONFIGURATION<br><br>SOAP FAULT MESSAGE<br><br>START MULTICAST - INVALID PROFILE TOKEN<br><br>The following Annexes were added or updated: |

| | | |
|---|---|---|
| | | Annex A.3 Create Empty Profile<br><br>Annex A.4 Get Guaranteed Number of Video Encoder Instances and Get Video Encoder Configuration Options Consistency Verification Description |
| 13.12 | December, 2013 | Pre-requisites were updated for the following test cases:<br><br>SOAP FAULT MESSAGE<br><br>SOAP FAULT MESSAGE<br><br>START MULTICAST - INVALID PROFILE TOKEN |
| 14.12 | Decemeber, 2014 | The following test cases were updated with ID change:<br><br>SET AUDIO OUTPUT CONFIGURATION – INVALID CONFIGURATION<br><br>DYNAMIC MEDIA PROFILE CONFIGURATION<br><br>AUDIO SOURCE CONFIGURATION<br><br>AUDIO ENCODER CONFIGURATION<br><br>PTZ CONFIGURATION<br><br>METADATA CONFIGURATION |
| 15.06 | July, 2015 | The following test cases were added:<br><br>GET AUDIO OUTPUT CONFIGURATION OPTIONS<br><br>AUDIO OUTPUT CONFIGURATION<br><br>AUDIO DECODER CONFIGURATION<br><br>PROFILES AND AUDIO OUTPUT CONFIGURATIONS CONSISTENCY<br><br>AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUT CONFIGURATION CONSISTENCY<br><br>AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUT CONFIGURATION OPTIONS CONSISTENCY<br><br>PROFILES AND AUDIO OUTPUT CONFIGURATION OPTIONS CONSISTENCY<br><br>AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUTS CONSISTENCY<br><br>PROFILES AND AUDIO DECODER CONFIGURATIONS CONSISTENCY<br><br>AUDIO DECODER CONFIGURATIONS AND AUDIO DECODER CONFIGURATION CONSISTENCY |
| 18.06 | June, 2018 | Reformatting document using new template |
| 19.06 | Mar, 2018 | The following test case was updated according to #1623:<br><br>DYNAMIC MEDIA PROFILE CONFIGURATION |
| 19.06 | Apr, 2018 | The following test case was removed according to #1826: |

| | | |
|---|---|---|
| | | MEDIA-4-1-4 PTZ CONFIGURATION |
| 19.12 | Aug 12, 2019 | The following test case was updated according to #1888:<br><br>MEDIA-3-1-7 GET AUDIO SOURCE CONFIGURATION OPTIONS (GetAudioSourceConfigurations replaced with GetCompatibleAudioSourceConfigurations) |
| 19.12 | Aug 03, 2019 | The following test case was updated according to #1663:<br><br>MEDIA-1-1-5 DYNAMIC MEDIA PROFILE CONFIGURATION (step 11.24 and steps 11.24.* were updated to try to find PTZ configuration compatible with video source configuration if GetCompatibleConfigurations is not supported) |
| 20.12 | Aug 03, 2020 | The following test case was updated according to #2071:<br><br>MEDIA-3-4-8 GET AUDIO OUTPUT CONFIGURATION OPTIONS (steps 3-5 were replaced with Annex A.6, for steps 6-7 GetAudioOutputConfigurations was replaced with GetCompatibleAudioOutputConfigurations, step 9 was updated to use Annex A.6 result as profileToken, restoring procedure after Annex A.6 was added) |
| 20.12 | Aug 03, 2020 | Test Sequences were removed as deprecated. |
| 24.06 | May 28, 2024 | The following test case removed according to #193<br><br>MEDIA-7-1-2 SOAP FAULT MESSAGE |

Table of Contents

# 1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementation from different vendors. The set of ONVIF test specification describes the test cases need to verify the [ONVIF Network Interface Specs] and [ONVIF Conformance] requirements. And also the test cases are to be basic inputs for some Profile specification requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Media Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases need to be executed and passed. And also this specification acts as an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards ONVIF standard. This test tool is referred as ONVIF Client hereafter.

## 1.1 Scope

This ONVIF Media Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be functional black-box testing. The objective of this specification to provide test cases to test individual requirements of ONVIF devices according to ONVIF Media Service which is defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for [ONVIF Network Interface Specs].

This specification **does not** address the following:

- Product use cases and non-functional (performance and regression) testing.
- SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
- Network protocol implementation Conformance test for HTTP, HTTPS, RTP and RTSP protocol.
- Poor streaming performance test (audio/video distortions, missing audio/video frames, incorrect lib synchronization etc.).

  Wi-Fi Conformance test

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Network Interface Specs]; instead it would cover subset of it. The scope of this specification

is to derive all the normative requirements of [ONVIF Network Interface Specs] which are related to ONVIF Media Service and some of the optional requirements.

This ONVIF Media Test Specification covers ONVIF Media service which is a functional block of [ONVIF Network Interface Specs]. The following sections describe the brief overview of and scope of each functional block.

## 1.2 Media Configuration

Media Configuration section covers the test cases needed for the verification of media service features as mentioned in [ONVIF Network Interface Specs]. Media service is used to configure the media and other real time streaming configurations.

Briefly it covers the following things

1. Manage media profiles.
2. Manage configuration entities.
3. Initiate and manipulate Audio/Video streams for different media formats.

The scope of this specification is to cover following configuration entities and Audio/Video media formats.

1. Configuration Entities:
    a. Video source configuration
    b. Audio source configuration
    c. Video encoder configuration
    d. Audio encoder configuration
    e. PTZ configuration
    f. Metadata configuration
2. Video Codec:
    a. JPEG QVGA
    b. MPEG-4, Simple Profile
    c. H.264
3. Audio Codec:

a. G.711

b. G.726

c. AAC

# 2 Normative references

- [ONVIF Conformance] ONVIF Conformance Process Specification:

  https://www.onvif.org/profiles/conformance/

- [ONVIF Profile Policy] ONVIF Profile Policy:

  https://www.onvif.org/profiles/

- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Core Specs] ONVIF Core Specification:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Media Spec] ONVIF Media Service Specification:

  https://www.onvif.org/profiles/specifications/

- [ONVIF Base Test] ONVIF Base Device Test Specification:

  https://www.onvif.org/profiles/conformance/device-test/

- [ISO/IEC Directives, Part 2] ISO/IEC Directives, Part 2, Annex H:

  http://www.iso.org/directives

- [ISO 16484-5] ISO 16484-5:2014-09 Annex P:

  https://www.iso.org/obp/ui/#!iso:std:63753:en

- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:

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

- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:

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

- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:

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

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

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

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

  http://www.ietf.org/rfc/rfc2326.txt

# 3 Terms and Definitions

## 3.1 Conventions

The key words "shall", "shall not", "should", "should not", "may", "need not", "can", "cannot" in this specification are to be interpreted as described in [ISO/IEC Directives Part 2].

## 3.2 Definitions

This section defines terms that are specific to the ONVIF Media Service and tests. For a list of applicable general terms and definitions, please see [ONVIF Base Test].

**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.

**Media Profile** A media profile maps a video and/or audio source to a video and/or an audio encoder, PTZ and analytics configurations.

## 3.3 Abbreviations

This section describes abbreviations used in this document.

**AAC** Advanced Audio Coding.

**JPEG** Joint Photographic Experts Group.

**MPEG-4** Moving Pictures Experts Group-4.

**QVGA** Quarter Video Graphics Array.

**TTL** Time To Live.

# 4 Test Overview

This section describes about the test setup and prerequisites needed, and the test policies that should be followed for test case execution.

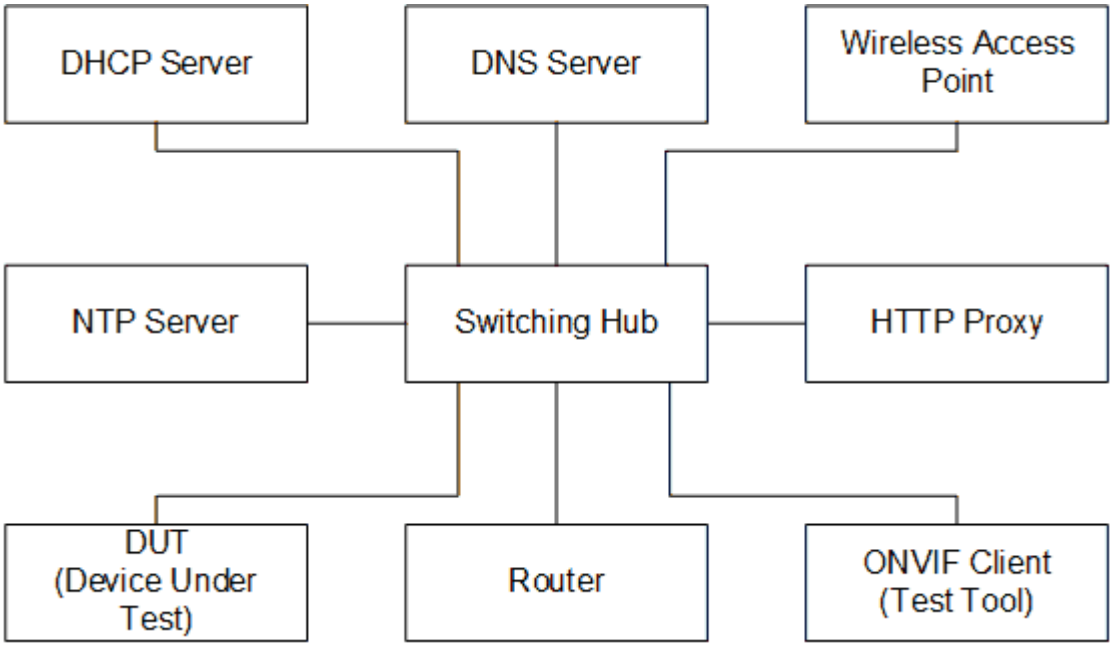
## 4.1 Test Setup

### 4.1.1 Network Configuration for DUT

The generic test configuration for the execution of test cases defined in this document is as shown below (Figure 4.1).

Based on the individual test case requirements, some of the entities in the below setup may not be needed for the execution of those corresponding test cases.

**Figure 4.1. Test Configuration for DUT**




**DUT:** ONVIF device to be tested. Hereafter, this is referred to as DUT (Device Under Test).

**ONVIF Client (Test Tool):** Tests are executed by this system and it controls the behavior of the DUT. It handles both expected and unexpected behavior.

**HTTP Proxy:** provides facilitation in case of RTP and RTSP tunneling over HTTP.

**Wireless Access Point:** provides wireless connectivity to the devices that support wireless connection.

**DNS Server:** provides DNS related information to the connected devices.

**DHCP Server:** provides IPv4 Address to the connected devices.

**NTP Server:** provides time synchronization between ONVIF Client and DUT.

**Switching Hub:** provides network connectivity among all the test equipments in the test environment. All devices should be connected to the Switching Hub. When running multiple test instances in parallel on the same network, the Switching Hub should be configured to use filtering in order to avoid multicast traffic being flooded to all ports, because this may affect test stability.

**Router:** provides router advertisements for IPv6 configuration.

## 4.2 Prerequisites

The pre-requisites for executing the test cases described in this Test Specification are:

1. The DUT shall be configured with an IPv4 address.
2. The DUT shall be IP reachable [in the test configuration].
3. The DUT shall be able to be discovered by the Test Tool.
4. The DUT shall be configured with the time i.e. manual configuration of UTC time and if NTP is supported by DUT, then NTP time shall be synchronized with NTP Server.
5. The DUT time and Test tool time shall be synchronized with each other either manually or by common NTP server

## 4.3 Test Policy

This section describes the test policies specific to the test case execution of each functional block.

The DUT shall adhere to the test policies defined in this section.

### 4.3.1 Media Configuration

Prior to the execution of Media Configuration test cases, DUT shall be discovered by ONVIF Client and it shall demonstrate media capabilities to ONVIF Client using device management service.

DUT shall support at least one media profile with Video Configuration. Video Configuration shall include video source and video encoder media entities.

DUT shall support JPEG QVGA encoding.

ONVIF Client shall explicitly specify the optional media formats supported by DUT.

ONVIF Client shall explicitly specify if the DUT supports Audio and PTZ.

DUT shall allow for creation of at least one media profile by ONVIF Client. In certain test cases, ONVIF Client may create new media configuration (i.e. media profile and media entities). In such cases, the test procedure will delete those modified configuration at the end of the test procedure.

DUT should respond with proper response message for all SOAP actions. Sending fault messages such as "ter:ConfigurationConflict" will be treated as FAILURE of the test cases.

Please refer to Section 5 for Media Configuration Test Cases.

# 5 Media Configuration Test Cases

## 5.1 Media Profile

### 5.1.1 MEDIA PROFILE CONFIGURATION

**Test Case ID:** MEDIA-1-1-1

**Specification Coverage:** Get media profiles

**Feature Under Test:** GetProfiles

**WSDL Reference:** media.wsdl

**Test Purpose:** To retrieve existing media profiles of DUT and the corresponding media entities (video source and video encoder).

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetProfilesRequest** message to retrieve existing media profiles of the DUT.
4. Verify that the DUT returns at least one media profile with video configuration (video source and video encoder) in the **GetProfilesResponse** message.
5. Verify that all media profile elements have ‘fixed’ attribute.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT has no default media profile configuration.
- DUT did not provide at least one media profile with video source and video encoder configurations.

- One or more media profiles do not have ‘fixed’ attribute.

## 5.1.2 PROFILES CONSISTENCY

**Test Case ID:** MEDIA-1-1-3

**Specification Coverage:** Get media profiles, Get media profile

**Feature Under Test:** GetProfiles, GetProfile

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfile command and GetProfiles command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their Configurations
4. The DUT sends **GetProfilesResponse** message.
5. Check that each Profile from **GetProfilesResponse** message has unique token.
6. ONVIF Client invokes **GetProfileRequest** (*ProfileToken*) message to retrieve profile from device.
7. The DUT sends **GetProfileResponse** message.
8. Verify that all parameters and their values for the same profile from **GetProfilesResponse** message and **GetProfileResponse** message are same.
9. Repeat steps 6-8 for other profiles from the **GetProfilesResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.

- The DUT did not send valid **GetProfilesResponse** message.
- The DUT return two or more Profiles in **GetProfilesResponse** message with the same *ProfileToken*.
- The DUT did not send **GetProfileResponse** message.
- The DUT did not send valid **GetProfileResponse** message.
- The DUT did not send equal parameters in the **GetProfileResponse** message and in the **GetProfilesResponse** message for the same profile.

### 5.1.3 DYNAMIC MEDIA PROFILE CONFIGURATION

**Test Case ID:** MEDIA-1-1-5

**Specification Coverage:** Create media profile, Get media profiles, Get media profile, Add video source configuration to a profile, Add video encoder configuration to a profile, Remove video source configuration from a profile, Remove video encoder configuration from a profile, Add PTZ configuration to a profile, Remove PTZ configuration from a profile, Add metadata configuration to a profile, Remove metadata configuration from a profile, Add audio source configuration to a profile, Remove audio source configuration from a profile, Add audio encoder configuration to a profile, Remove audio encoder configuration from a profile, Add audio output configuration to a profile, Remove audio output configuration from a profile, Add audio decoder configuration to a profile, Remove audio decoder configuration from a profile, Delete media profile.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of the DUT for dynamic media profile configuration.

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** request
4. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*

5. If *profileList* does not contain at least one media profile with video configuration (video source and video encoder), FAIL the test and skip other steps.
6. ONVIF Client invokes **CreateProfile** request message to create a new empty media profile.
7. The DUT returns an empty profile with no profile entities in the **CreateProfileResponse** message or SOAP 1.2 fault message (Action/MaxNVTProfiles) from the DUT. If fault was received, execute Annex A.3.
8. If **CreateProfileResponse** was returned and created profile has @fixed=true, FAIL the test and skip other steps.
9. Set *profileToken* := token of returned profile in steps 6 or 7.
10. ONVIF Client retrieves Video Source Configurations list by following the procedure mentioned in Annex A.9 with the following input and output parameters
    - out *videoSourceConfList* - Video Source Configurations list
11. For each Video Source Configuration videoSourceConfiguration in *videoSourceConfList* repeat the following steps:
    11.1. ONVIF Client invokes **AddVideoSourceConfiguration** request with parameters
        - ProfileToken := *profileToken*
        - ConfigurationToken := *videoSourceConfiguration*.@token

    11.2. The DUT sends **AddVideoSourceConfigurationResponse** message.

    11.3. ONVIF Client invokes **GetProfile** request with parameters
        - ProfileToken := *profileToken*

    11.4. The DUT responds with **GetProfileResponse** message with parameters
        - Profile := *profile1*

    11.5. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

    11.6. *profile1*.VideoSourceConfiguration.@token != *videoSourceConfiguration*.@token, FAIL the test and skip other steps.

    11.7. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurations** request with parameters
        - ProfileToken := *profileToken*

11.8. The DUT responds with compatible video encoder configurations in **GetCompatibleVideoEncoderConfigurationsResponse** with parameters

- Configurations list =: *videoEncoderConfigurationList*

11.9. If *videoEncoderConfigurationList* is empty, FAIL the test and skip other steps.

11.10. Set *videoEncoderConfiguration* := *videoEncoderConfigurationList*.Configurations[0].

11.11. ONVIF Client invokes **AddVideoEncoderConfiguration** request with parameters

- Profile Token := *profileToken*
- ConfigurationToken := *videoEncoderConfiguration*.@token

11.12. The DUT responds with **AddVideoEncoderConfigurationResponse**.

11.13. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

11.14. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

11.15. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

11.16. If *profile1*.VideoSourceConfiguration.@token != *videoSourceConfiguration*.@token, FAIL the test and skip other steps.

11.17. If *profile1*.VideoEncoderConfiguration.@token != *videoEncoderConfiguration*.@token, FAIL the test and skip other steps.

11.18. ONVIF Client invokes **RemoveVideoEncoderConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

11.19. The DUT responds with **RemoveVideoEncoderConfigurationResponse** message.

11.20. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

11.21. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

11.22. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

11.23. If *profile1*.VideoSourceConfiguration.@token != *videoSourceConfiguration*.@token, FAIL the test and skip other steps.

11.24. If *profile1* contains VideoEncoderConfiguration, FAIL the test and skip other steps.

11.25. If the DUT supports PTZ:

11.25.1. If DUT supports Get Compatible Configurations:

11.25.1.1. ONVIF Client invokes **GetCompatibleConfigurations** request with parameters

- ProfileToken := *profileToken*

11.25.1.2. The DUT responds with **GetCompatibleConfigurationsResponse** request with parameters

- PTZConfiguration list =: *ptzConfigurationList*

11.25.1.3. If *ptzConfigurationList* is empty go to step 26.

11.25.1.4. Set *ptzConfigurationToken* := *ptzConfigurationList*[0].@token.

11.25.1.5. ONVIF Client invokes **AddPTZConfiguration** request with parameters

- Profile Token := *profileToken*
- ConfigurationToken := *ptzConfigurationToken*

11.25.1.6. The DUT responds with **AddPTZConfigurationResponse**.

11.25.2. If DUT does not support Get Compatible Configurations:

11.25.2.1. ONVIF Client invokes **GetConfigurations** request

11.25.2.2. The DUT responds with **GetConfigurationsResponse** request with parameters

- PTZConfiguration list =: *ptzConfigurationList*

11.25.2.3. If *ptzConfigurationList* is empty go to step 26.

11.25.2.4. For each PTZ Configuration *ptzConfiguration* in *ptzConfigurationList* repeat the following steps:

11.25.2.4.1. Set *ptzConfigurationToken* := *ptzConfiguration*.@token.

11.25.2.4.2. ONVIF Client invokes **AddPTZConfiguration** request with parameters

- Profile Token := *profileToken*
- ConfigurationToken := *ptzConfigurationToken*

11.25.2.4.3. The DUT responds with **AddPTZConfigurationResponse** or with SOAP 1.2 fault.

11.25.2.4.4. If DUT responds with **AddPTZConfigurationResponse** go to step 11.25.3.

11.25.2.5. Go to step 26.

11.25.3. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

11.25.4. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

11.25.5. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

11.25.6. If *profile1*.VideoSourceConfiguration.@token != *videoSourceConfiguration*.@token, FAIL the test and skip other steps.

11.25.7. *profile1*.PTZConfiguration.@token != *ptzConfigurationToken*, FAIL the test and skip other steps.

11.25.8. ONVIF Client invokes **RemovePTZConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

11.25.9. The DUT responds with **RemovePTZResponse** message.

11.25.10. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

11.25.11. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

11.25.12. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

11.25.13. If *profile1*.VideoSourceConfiguration.@token != *videoSourceConfiguration*.@token, FAIL the test and skip other steps.

11.25.14. If *profile1* contains PTZConfiguration, FAIL the test and skip other steps.

11.26. ONVIF Client invokes **RemoveVideoSourceConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

11.27. The DUT responds with **RemoveVideoSourceConfigurationResponse** message.

11.28. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

11.29. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

11.30. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

11.31. If *profile1* contains VideoSourceConfiguration, FAIL the test and skip other steps.

12. ONVIF Client retrieves Metadata Configurations list by following the procedure mentioned in Annex A.10 with the following input and output parameters

- out *metadataConfList* - Metadata Configurations list

13. For each Metadata Configuration *metadataConfiguration* in *metadataConfList* repeat the following steps:

13.1. ONVIF Client invokes **AddMetadataConfiguration** request with parameters

- Profile Token := *profileToken*

- ConfigurationToken := *metadataConfiguration*.@token

13.2. The DUT responds with **AddMetadataConfigurationResponse**.

13.3. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

13.4. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

13.5. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

13.6. If *profile1*.MetadataConfiguration.@token != *metadataConfiguration*.@token, FAIL the test and skip other steps.

13.7. ONVIF Client invokes **RemoveMetadataConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

13.8. The DUT responds with **RemoveMetadataConfigurationResponse** message.

13.9. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

13.10. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

13.11. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

13.12. If *profile1* contains MetadataConfiguration, FAIL the test and skip other steps.

14. If the DUT supports Audio:

14.1. ONVIF Client retrieves Audio Source Configurations list by following the procedure mentioned in Annex A.11 with the following input and output parameters

- out *audioSourceConfList* - Audio Source Configurations list

14.2. For each Audio Source Configuration *audioSourceConfiguration* in *audioSourceConfList* repeat the following steps:

14.2.1. ONVIF Client invokes **AddAudioSourceConfiguration** request with parameters

- ProfileToken := *profileToken*
- ConfigurationToken := *audioSourceConfiguration*.@token

14.2.2. The DUT sends **AddAudioSourceConfigurationResponse** message.

14.2.3. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

14.2.4. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

14.2.5. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

14.2.6. *profile1*.AudioSourceConfiguration.@token != *audioSourceConfiguration*.@token, FAIL the test and skip other steps.

14.2.7. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurations** request with parameters

- ProfileToken := *profileToken*

14.2.8. The DUT responds with compatible audio encoder configurations in **GetCompatibleAudioEncoderConfigurationsResponse** with parameters

- Configurations list =: *audioEncoderConfigurationList*

14.2.9. If *audioEncoderConfigurationList* is empty, FAIL the test and skip other steps.

14.2.10. Set *audioEncoderConfiguration* := *audioEncoderConfigurationList*.Configurations[0].

14.2.11. ONVIF Client invokes **AddAudioEncoderConfiguration** request with parameters

- Profile Token := *profileToken*
- ConfigurationToken := *audioEncoderConfiguration*.@token

14.2.12. The DUT responds with **AddAudioEncoderConfigurationResponse**.

14.2.13. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

14.2.14. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

14.2.15. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

14.2.16. If *profile1*.AudioSourceConfiguration.@token != *audioSourceConfiguration*.@token, FAIL the test and skip other steps.

14.2.17. If *profile1*.AudioEncoderConfiguration.@token != *audioEncoderConfiguration*.@token, FAIL the test and skip other steps.

14.2.18. ONVIF Client invokes **RemoveAudioEncoderConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

14.2.19. The DUT responds with **RemoveAudioEncoderConfigurationResponse** message.

14.2.20. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

14.2.21. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

14.2.22. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

14.2.23. If *profile1*.AudioSourceConfiguration.@token != *audioSourceConfiguration*.@token, FAIL the test and skip other steps.

14.2.24. If *profile1* contains AudioEncoderConfiguration, FAIL the test and skip other steps.

14.2.25. ONVIF Client invokes **RemoveAudioSourceConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

14.2.26. The DUT responds with **RemoveAudioSourceConfigurationResponse** message.

14.2.27. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

14.2.28. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

14.2.29. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

14.2.30. If *profile1* contains AudioSourceConfiguration, FAIL the test and skip other steps.

15. If the DUT supports Audio Output:

15.1. ONVIF Client retrieves Audio Output Configurations list by following the procedure mentioned in Annex A.12 with the following input and output parameters

- out *audioOutputConfList* - Audio Output Configurations list

15.2. For each Audio Output Configuration *audioOutputConfiguration* in *audioOutputConfList* repeat the following steps:

15.2.1. ONVIF Client invokes **AddAudioOutputConfiguration** request with parameters

- ProfileToken := *profileToken*
- ConfigurationToken := *audioOutputConfiguration*.@token

15.2.2. The DUT sends **AddAudioOutputConfigurationResponse** message.

15.2.3. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

15.2.4. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

15.2.5. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

15.2.6. *profile1*.AudioOutputConfiguration.@token != *audioOutputConfiguration*.@token, FAIL the test and skip other steps.

15.2.7. ONVIF Client invokes **GetCompatibleAudioDecoderConfigurations** request with parameters

- ProfileToken := *profileToken*

15.2.8. The DUT responds with compatible audio decoder configurations in **GetCompatibleAudioDecoderConfigurationsResponse** with parameters

- Configurations list =: *audioDecoderConfigurationList*

15.2.9. If *audioDecoderConfigurationList* is empty, FAIL the test and skip other steps.

15.2.10. Set *audioDecoderConfiguration* := *audioDecoderConfigurationList*.Configurations[0].

15.2.11. ONVIF Client invokes **AddAudioDecoderConfiguration** request with parameters

- Profile Token := *profileToken*
- ConfigurationToken := *audioDecoderConfiguration*.@token

15.2.12. The DUT responds with **AddAudioDecoderConfigurationResponse**.

15.2.13. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

15.2.14. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

15.2.15. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

15.2.16. If *profile1*.AudioOutputConfiguration.@token != *audioOutputConfiguration*.@token, FAIL the test and skip other steps.

15.2.17. If *profile1*.AudioDecoderConfiguration.@token != *audioDecoderConfiguration*.@token, FAIL the test and skip other steps.

15.2.18. ONVIF Client invokes **RemoveAudioDecoderConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

15.2.19. The DUT responds with **RemoveAudioDecoderConfigurationResponse** message.

15.2.20. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

15.2.21. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

15.2.22. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

15.2.23. If *profile1*.AudioOutputConfiguration.@token != *audioOutputConfiguration*.@token, FAIL the test and skip other steps.

15.2.24. If *profile1* contains AudioDecoderConfiguration, FAIL the test and skip other steps.

15.2.25. ONVIF Client invokes **RemoveAudioOutputConfiguration** request message with parameters

- ProfileToken := *profile1*.@token

15.2.26. The DUT responds with **RemoveAudioOutputConfigurationResponse** message.

15.2.27. ONVIF Client invokes **GetProfile** request with parameters

- ProfileToken := *profileToken*

15.2.28. The DUT responds with **GetProfileResponse** message with parameters

- Profile := *profile1*

15.2.29. If *profile1*.@token != *profileToken*, FAIL the test and skip other steps.

15.2.30. If *profile1* contains AudioOutputConfiguration, FAIL the test and skip other steps.

16. If profile with ProfileToken = *profileToken* was created during test execution

16.1. ONVIF Client invokes **DeleteProfile** request with parameters

- ProfileToken := *profileToken*

16.2. The DUT responds with **DeleteProfileResponse** message.

16.3. ONVIF Client invokes **GetProfiles** request with parameters

- ProfileToken := *profileToken*

16.4. The DUT returns **env:Sender/ter:InvalidArgVal/ter:NoProfile** SOAP 1.2 fault.

17. ONVIF Client restores DUT configuration if requered.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send **CreateProfileResponse** message.
- The DUT did not send **GetCompatibleAudioDecoderConfigurations** message.
- The DUT did not send **GetCompatibleAudioEncoderConfigurations** message.
- The DUT did not send **GetAudioOutputConfigurations** message.
- The DUT did not send **GetAudioSourceConfigurations** message.
- The DUT did not send **GetMetadataConfigurations** message.
- The DUT did not send **GetCompatibleVideoEncoderConfigurations** message.
- The DUT did not send **GetVideoSourceConfigurations** message.
- The DUT did not send **GetCompatibleConfigurations** message.
- The DUT did not send **GetConfigurations** message.
- The DUT did not send **AddVideoSourceConfigurationResponse** message.
- The DUT did not send **AddVideoEncoderConfigurationResponse** message.
- The DUT did not send **AddPTZConfigurationResponse** message.
- The DUT did not send **AddMetadataConfigurationResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **AddAudioEncoderConfigurationResponse** message.

- The DUT did not send **AddAudioOutputConfigurationResponse** message.
- The DUT did not send **AddAudioDecoderConfigurationResponse** message.
- The DUT did not send **GetProfileResponse** message.
- The DUT did not send **RemoveVideoEncoderConfigurationResponse** message.
- The DUT did not send **RemoveVideoSourceConfigurationResponse** message.
- The DUT did not send **RemovePTZConfigurationResponse** message.
- The DUT did not send **RemoveMetadataConfigurationResponse** message.
- The DUT did not send **RemoveAudioEncoderConfigurationResponse** message.
- The DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- The DUT did not send **RemoveAudioOutputConfigurationResponse** message.
- The DUT did not send **RemoveAudioDecoderConfigurationResponse** message.
- The DUT did not send **DeleteProfileResponse** message.

**NOTE:** See Annex in [ONVIF Base Test] for Invalid SOAP 1.2 fault message definition.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2 Video Configuration

### 5.2.1 VIDEO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-2-1-2

**Specification Coverage:** Get media profiles, Get video encoder configurations, Get compatible video encoder configurations

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Video Encoder Configuration Operations

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** Request.
4. DUT sends the list of existing media profiles in **GetProfilesResponse** message.
5. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurations** request with one of the received media profile tokens as input argument.
6. DUT sends the list of video encoder configurations, compatible with the received media profile token.
7. ONVIF Client verifies the list of video encoder configurations sent by DUT.
8. ONVIF Client will invoke **GetVideoEncoderConfigurations** request to retrieve the list of video encoder configurations supported by the DUT.
9. DUT sends the list of all video encoder configurations supported by it.
10. ONVIF Client verifies the list of video encoder configurations sent by the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT did not send valid **GetCompatibleVideoEncoderConfigurationsResponse** message.
- DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.

## 5.2.2 GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES

**Test Case ID:** MEDIA-2-1-6

**Specification Coverage:** Get video source configurations, Get guaranteed number of video encoder instances

**Feature Under Test:** GetGuaranteedNumberOfVideoEncoderInstances

**WSDL Reference:** media.wsdl

**Test Purpose:** To retrieve minimum number of video encoder instances supported by DUT per video source configuration.

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurations** request.
4. DUT sends the list of video source configurations supported by it.
5. ONVIF Client invokes **GetGuaranteedNumberOfVideoEncoderInstancesRequest** message for a selected video source configuration.
6. DUT sends the minimum guaranteed number of video encoder instances.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **GetGuaranteedNumberOfVideoEncoderInstancesResponse** message.
- DUT did not send 'TotalNumber' value.

## 5.2.3 GET GUARANTEED NUMBER OF VIDEO ENCODER INSTANCES AND GET VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-2-1-7

**Specification Coverage:** GetGuaranteedNumberOfVideoEncoderInstances (ONVIF Media Service Specification)

**Feature Under Test:** GetGuaranteedNumberOfVideoEncoderInstances

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify Get Guaranteed Number of Video Encoder Instances and Get Video Encoder Configuration Options Consistency. To verify Get Guaranteed Number of Video Encoder Instances and Get Video Encoder Configurations Consistency. To verify Get Guaranteed Number of Video Encoder Instances and MaximumNumberOfProfiles capability Consistency.

**Pre-Requisite:** Media is supported by DUT. Media Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. If GetCapabilities supported by the DUT, ONVIF Client will invoke **GetCapabilitiesRequest** message (Category = 'Media') to get MaximumNumberOfProfiles capability. Otherwise skip steps 3-4 and go to the step 5.
4. Verify the **GetCapabilitiesResponse** message from the DUT.
5. If GetServices supported by the DUT, ONVIF Client will invoke **GetServiceCapabilitiesRequest** message to get MaximumNumberOfProfiles capability. Otherwise skip steps 5-6 and go to the step 7.
6. Verify the **GetServiceCapabilitiesResponse** message from the DUT.
7. ONVIF Client will invoke **GetVideoEncoderConfigurationsRequest** message to retrieve all DUT video encoder configurations.
8. Verify the **GetVideoSourceConfigurationsResponse** message from the DUT.
9. ONVIF Client will invoke **GetVideoSourceConfigurationsRequest** message to retrieve all DUT video source configurations.
10. Verify the **GetVideoSourceConfigurationsResponse** message from the DUT.
11. ONVIF Client will invoke **GetGuaranteedNumberOfVideoEncoderInstancesRequest** message (ConfigurationToken = "Token1", where "Token1" is a first video source

configuration token from **GetVideoSourceConfigurationsResponse** message) to retrieve guaranteed number of video encoder instances per first video source configuration.

12. Verify the **GetGuaranteedNumberOfVideoEncoderInstancesResponse** message from the DUT.

13. Verify that **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber is less or equal to total number of VideoEncoderConfigurations.

14. If GetCapabilities is supported by the DUT, verify that **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber is less or equal to total number of GetCapabilitiesResponse.Media.Extension.ProfileCapabilities.MaximumNumberOfProfiles if specified.

15. If GetServices is supported by the DUT, Verify that **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber is less or equal to total number of **GetServiceCapabilitiesResponse**.Capabilities.ProfileCapabilities.MaximumNumberOfProfiles if specified.

16. Repeat steps 11-15 to retrieve guaranteed number of video encoder instances for all video source configuration

17. ONVIF Client will invoke **GetVideoEncoderConfigurationOptionsRequest** message (no ConfigurationToken, no ProfileToken) to retrieve general video encoder options for the DUT.

18. Verify the **GetVideoEncoderConfigurationOptionsResponse** message from the DUT.

19. Verify that **GetVideoEncoderConfigurationOptionsResponse** contains Options.JPEG, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with JPEG element having value greater than 0.

20. Verify that **GetVideoEncoderConfigurationOptionsResponse** contains Options.MPEG4, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with MPEG4 element having value greater than 0.

21. Verify that **GetVideoEncoderConfigurationOptionsResponse** contains Options.H264, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with H264 element having value greater than 0.

22. Verify that **GetVideoEncoderConfigurationOptionsResponse** does not contain Options.JPEG, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponses** with skipped JPEG

element or with JPEG element having value greater than 0 (i.e. all **GetGuaranteedNumberOfVideoEncoderInstancesResponses** have JPEG element that is equal to 0).

23. Verify that **GetVideoEncoderConfigurationOptionsResponse** does not contain Options.MPEG4, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponses** with skipped MPEG4 element or with MPEG4 element having value greater than 0 (i.e. all **GetGuaranteedNumberOfVideoEncoderInstancesResponses** have MPEG4 element that is equal to 0).

24. Verify that **GetVideoEncoderConfigurationOptionsResponse** does not contain Options.H264, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponses** with skipped H264 element or with H264 element having value greater than 0 (i.e. all **GetGuaranteedNumberOfVideoEncoderInstancesResponses** have H264 element that is equal to 0).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** with Options.JPEG element, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with skipped JPEG element or with JPEG element having value greater than 0.
- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** with Options.MPEG4 element, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with skipped MPEG4 element or with MPEG4 element having value greater than 0.
- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** with Options.H264 element, if there are no **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with skipped H264 element or with H264 element having value greater than 0.

- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** without Options.JPEG element, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with JPEG element having value greater than 0.
- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** without Options.MPEG4 element, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with MPEG4 element having value greater than 0.
- The DUT returned **GetVideoEncoderConfigurationOptionsResponse** without Options.H264 element, if there is at least one **GetGuaranteedNumberOfVideoEncoderInstancesResponse** with H264 element having value greater than 0.
- The DUT returned **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber greater than total number of VideoEncoderConfigurations.
- The DUT returned **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber greater than **GetServiceCapabilitiesResponse**.Capabilities.ProfileCapabilities.MaximumNumberOfProfiles.
- The DUT returned **GetGuaranteedNumberOfVideoEncoderInstancesResponse**.TotalNumber greater than **GetCapabilitiesResponse**.Media.Extension.ProfileCapabilities.MaximumNumberOfProfiles.

**NOTE:** See Annex A.4 for more details about test checks.

## 5.2.4 VIDEO SOURCE CONFIGURATION

**Test Case ID:** MEDIA-2-1-8

**Specification Coverage:** Get media profiles, GetVideoSources, Get video source configurations, Get video source configuration, Get compatible video source configurations, Get video source configuration options, Modify a video source configuration

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Video Source Configuration Operations

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** Request.
4. DUT sends the list of existing media profiles in **GetProfilesResponse** message.
5. ONVIF Client invokes **GetVideoSources** request to retrieve the existing video sources of DUT.
6. ONVIF Client verifies the list of video sources sent by DUT.
7. ONVIF Client invokes **GetCompatibleVideoSourceConfigurations** request with one of the received media profile tokens as input argument.
8. DUT sends the list of video source configurations compatible with the received media profile token.
9. ONVIF Client verifies the list of video source configurations sent by DUT.
10. ONVIF Client invokes **GetVideoSourceConfigurations** request to retrieve the list of video source configurations supported by the DUT.
11. ONVIF Client verifies the list of video source configurations sent by the DUT.
12. ONVIF Client invokes **GetVideoSourceConfigurationOptions** request with one of the received video source configuration tokens as input argument.
13. ONVIF Client sends the range of configurable values supported by it for the received video source configuration token.
14. ONVIF Client invokes **SetVideoSourceConfiguration** request with video source configuration values outside the range specified in the **GetVideoSourceConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
15. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
16. ONVIF Client verifies the SOAP fault message sent by DUT.
17. ONVIF Client invokes **SetVideoSourceConfiguration** request with video source configuration values within the range specified in

**GetVideoSourceConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.

18. DUT modifies the video source configuration and sends the **SetVideoSourceConfigurationResponse** indicating success.
19. ONVIF Client invokes **GetVideoSourceConfiguration** request to verify the modified video source configuration.
20. DUT sends the modified video source configuration in **GetVideoSourceConfigurationResponse**.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT did not send valid **GetVideoSourcesResponse** message.
- DUT did not send valid **GetCompatibleVideoSourceConfigurationsResponse** message.
- DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **GetVideoSourceConfigurationOptionsResponse** message.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetVideoSourceConfiguration** request.
- DUT did not send **SetVideoSourceConfigurationResponse** message.
- DUT did not send **GetVideoSourceConfigurationResponse** message.
- DUT did not modify video source configuration correctly.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.5 JPEG VIDEO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-2-1-9

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration, Get video encoder configuration options, Modify a video encoder configuration.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT JPEG Video Encoder Configurations Setting

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** to retrieve the list of video encoder configurations supported by DUT.
4. DUT sends video encoder configurations in the **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** Request (Video Encoder Configuration token) to retrieve video encoder configuration options for the specified video encoder configuration.
6. DUT sends the range of configurable values for the received video encoder configuration in the GetVideoEncoderConfigurationOptionsResponse message.
7. Test steps 5 & 6 have to be repeated for all video encoder configurations until ONVIF Client finds a video encoder configuration with JPEG encoding support.
8. ONVIF Client invokes **SetVideoEncoderConfiguration** request with JPEG Resolution value configuration values outside the range defined in the **GetVideoEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
9. DUT sends the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**)
10. ONVIF Client verifies the SOAP fault message sent by DUT.
11. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = "JPEG", Resolution = Highest resolution based on number of pixels, Quality = QualityRange.Max, FramerateLimit = FrameRateRange.Max, BitrateLimit = "64000", EncodingInterval = EncodingIntervalRange.Min, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

12. DUT modifies JPEG video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.

13. ONVIF Client verifies the JPEG Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

14. DUT sends modified JPEG Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “JPEG”, Resolution = Highest resolution based on number of pixels).

15. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = “JPEG”, Resolution = Lowest resolution based on number of pixels, Quality = QualityRange.Min, FramerateLimit = FrameRateRange.Min, BitrateLimit = “64000”, EncodingInterval = EncodingIntervalRange.Max, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

16. DUT modifies JPEG video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.

17. ONVIF Client verifies the JPEG Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

18. DUT sends modified JPEG Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “JPEG”, Resolution = Lowest resolution based on number of pixels).

19. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = “JPEG”, Resolution = Median resolution based on number of pixels, Quality = Median value of QualityRange, FramerateLimit = Median value of FrameRateRange, BitrateLimit = “64000”, EncodingInterval = Median value of EncodingIntervalRange, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

20. DUT modifies JPEG video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.

21. ONVIF Client verifies the JPEG Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

22. DUT sends modified JPEG Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “JPEG”, Resolution = Median resolution based on number of pixels).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT doesn't support JPEG encoding.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetVideoEncoderConfiguration** request.
- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetVideoEncoderConfigurationResponse** message.
- The DUT did not modify JPEG Video Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.6 MPEG4 VIDEO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-2-1-10

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration, Get video encoder configuration options, Modify a video encoder configuration.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT MPEG4 Video Encoder Configurations Setting

**Pre-Requisite:** MPEG4 is implemented by DUT

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** to retrieve the list of video encoder configurations supported by DUT.

4. DUT sends video encoder configurations in the **GetVideoEncoderConfigurationsResponse** message.

5. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** Request (Video Encoder Configuration token) to retrieve video encoder configuration options for the specified video encoder configuration.

6. DUT sends the range of configurable values for the received video encoder configuration in the **GetVideoEncoderConfigurationOptionsResponse** message.

7. Test steps 5 & 6 have to be repeated for all video encoder configurations until ONVIF Client finds a video encoder configuration with MPEG4 encoding support.

8. ONVIF Client invokes **SetVideoEncoderConfiguration** request with MPEG4 configuration values outside the range defined in the **GetVideoEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.

9. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**)

10. ONVIF Client verifies the SOAP fault message sent by DUT.

11. Client invokes **SetVideoEncoderConfiguration** request (Encoding = "MPEG4", Mpeg4Profile = "SP", if "SP" is not supported "ASP", Resolution = Highest resolution based on number of pixels, Quality = QualityRange.Max, FramerateLimit = FrameRateRange.Max, BitrateLimit = "64000", EncodingInterval = EncodingIntervalRange.Min, GovLength = GovLengthRange.Min, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

12. DUT modifies MPEG4 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.

13. ONVIF Client verifies the MPEG4 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

14. DUT sends modified MPEG4 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = "MPEG4", Mpeg4Profile = "SP", if "SP" is not supported "ASP", Resolution = Highest resolution based on number of pixels).

15. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = "MPEG4", Mpeg4Profile = "ASP", if "ASP" is not supported "SP", Resolution = Lowest resolution based on number of pixels, Quality = QualityRange.Min, FramerateLimit = FrameRateRange.Min, BitrateLimit = "64000", EncodingInterval = EncodingIntervalRange.Max, GovLength =

GovLengthRange.Max, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

16. DUT modifies MPEG4 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.
17. ONVIF Client verifies the MPEG4 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.
18. DUT sends modified MPEG4 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “MPEG4”, Mpeg4Profile = “ASP”, if “ASP” is not supported “SP”, Resolution = Lowest resolution based on number of pixels).
19. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = “MPEG4”, Mpeg4Profile = “SP”, if “SP” is not supported “ASP”, Resolution = Median resolution based on number of pixels, Quality = Median value of QualityRange, FramerateLimit = Median value of FrameRateRange, BitrateLimit = “64000”, EncodingInterval = Median value of EncodingIntervalRange, GovLength = Median value of GovLengthRange, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.
20. DUT modifies MPEG4 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.
21. ONVIF Client verifies the MPEG4 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.
22. DUT sends modified MPEG4 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “MPEG4”, Mpeg4Profile = “SP”, if “SP” is not supported “ASP”, Resolution = Median resolution based on number of pixels).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT doesn’t support MPEG4 encoding.

- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetVideoEncoderConfiguration** request.
- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetVideoEncoderConfigurationResponse** message.
- The DUT did not modify MPEG4 Video Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.7 H.264 VIDEO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-2-1-11

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration, Get video encoder configuration options, Modify a video encoder configuration.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT H.264 Video Encoder Configurations Setting

**Pre-Requisite:** H.264 is implemented by DUT

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** to retrieve the list of video encoder configurations supported by DUT.
4. DUT sends video encoder configurations in the **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** Request (Video Encoder Configuration token) to retrieve video encoder configuration options for the specified video encoder configuration.
6. DUT sends the range of configurable values for the received video encoder configuration in the **GetVideoEncoderConfigurationOptionsResponse** message.

7. Test steps 5 & 6 have to be repeated for all video encoder configurations until ONVIF Client finds a video encoder configuration with H.264 encoding support
8. ONVIF Client invokes **SetVideoEncoderConfiguration** request with H.264 configuration values outside the range defined in the **GetVideoEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
9. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**)
10. ONVIF Client verifies the SOAP fault message sent by DUT.
11. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = "H264", H264Profile = "Baseline", if "Baseline" is not supported "Main", if "Main" is not supported "Extended" and if "Extended" is not supported "High", Resolution = Highest resolution based on number of pixels, Quality = QualityRange.Max, FramerateLimit = FrameRateRange.Max, BitrateLimit = "64000", EncodingInterval = EncodingIntervalRange.Min, GovLength = GovLengthRange.Min, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.
12. DUT modifies H.264 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.
13. ONVIF Client verifies the H.264 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.
14. DUT sends modified H.264 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = "H264", H264Profile "Baseline", if "Baseline" is not supported "Main", if "Main" is not supported "Extended" and if "Extended" is not supported "High", Resolution = Highest resolution based on number of pixels).
15. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = "H264", H264Profile = "Main", if "Main" is not supported "Extended", if "Extended" is not supported "High" and if "High" is not supported "Baseline", Resolution = Lowest resolution based on number of pixels, Quality = QualityRange.Min, FramerateLimit = FrameRateRange.Min, BitrateLimit = "64000", EncodingInterval = EncodingIntervalRange.Max, GovLength = GovLengthRange.Max, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.
16. DUT modifies H.264 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.
17. ONVIF Client verifies the H.264 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

18. DUT sends modified H.264 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “H264”, H264Profile = “Main”, if “Main” is not supported “Extended”, if “Extended” is not supported “High” and if “High” is not supported “Baseline”, Resolution = Lowest resolution based on number of pixels).

19. ONVIF Client invokes **SetVideoEncoderConfiguration** request (Encoding = “H264”, H264Profile = “Extended”, if “Extended” is not supported “High”, if “High” is not supported “Baseline” and if “Baseline” is not supported “Main”, Resolution = Median resolution based on number of pixels, Quality = Median value of QualityRange, FramerateLimit = Median value of FrameRateRange, BitrateLimit = “64000”, EncodingInterval = Median value of EncodingIntervalRange, GovLength = Median value of GovLengthRange, and force persistence = false). These values will be taken from **GetVideoEncoderConfigurationOptionsResponse** message.

20. DUT modifies H.264 video encoder configuration and responds with **SetVideoEncoderConfigurationResponse** message indicating success.

21. ONVIF Client verifies the H.264 Video Encoder Configuration settings on DUT by invoking **GetVideoEncoderConfiguration** request.

22. DUT sends modified H.264 Video Encoder Configuration in the **GetVideoEncoderConfigurationResponse** message (Encoding = “H264”, H264Profile “Extended”, if “Extended” is not supported “High”, if “High” is not supported “Baseline” and if “Baseline” is not supported “Main”, Resolution = Median resolution based on number of pixels).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetVideoEncoderConfigurationsResponse**.
- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT doesn’t support H.264 encoding.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetVideoEncoderConfiguration** request.
- DUT did not send **GetVideoEncoderConfigurationResponse** message.

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- The DUT did not modify H.264 Video Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.8 Video Source Configuration

### 5.2.8.1 VIDEO SOURCE CONFIGURATIONS AND PROFILES CONSISTENCY

**Test Case ID:** MEDIA-2-2-1

**Specification Coverage:** Get media profiles, Get video source configurations

**Feature Under Test:** GetProfiles, GetVideoSourceConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoSourceConfigurations command and GetProfiles command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their configurations.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of Video Source Configurations from device.
6. The DUT sends **GetVideoSourceConfigurationsResponse** message.
7. Check that each VideoSourceConfiguration from **GetVideoSourceConfigurationsResponse** message has unique token.
8. Check that each VideoSourceConfigurations from the **GetProfilesResponse** message are included in the **GetVideoSourceConfigurationsResponse** message.

9. Check that VideoSourceConfiguration parameters are same in the **GetProfilesResponse** message and in the **GetVideoSourceConfigurationsResponse** message for each VideoSourceConfiguration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT return two or more VideoSourceConfigurations in **GetVideoSourceConfigurationsResponse** message with the same ConfigurationToken.
- The DUT returned the **GetProfilesResponse** message with VideoSourceConfigurations that were not included in the **GetVideoSourceConfigurationsResponse** message.
- The DUT returned different parameters list and parameters values for the same VideoSourceConfiguration in the **GetVideoSourceConfigurationsResponse** message and in the **GetProfilesResponse** message.

### 5.2.8.2 VIDEO SOURCE CONFIGURATIONS AND VIDEO SOURCE CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-2-2-2

**Specification Coverage:** Get video source configurations, Get video source configuration

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoSourceConfigurations and GetVideoSourceConfiguration are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of Video Source Configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (ConfigurationToken) message to retrieve parameters of Video Source Configuration from device.
6. The DUT sends **GetVideoSourceConfigurationResponse** message.
7. Verify that all parameters and their values for video source ConfigurationToken from **GetVideoSourceConfigurationsResponse** message and **GetVideoSourceConfigurationResponse** message are the same.
8. Repeat steps 5-7 for other VideoSourceConfigurations from the **GetVideoSourceConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- The DUT did not send equal parameters for VideoSourceConfiguration in the **GetVideoSourceConfigurationResponse** message and in the **GetVideoSourceConfigurationsResponse** message.

### 5.2.8.3 VIDEO SOURCE CONFIGURATIONS AND VIDEO SOURCE CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-2-2-3

**Specification Coverage:** Get video source configuration options, Get video source configurations

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoSourceConfigurations Command and GetVideoSourceConfigurationOptions Command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of Video Source Configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationOptionsRequest** (ConfigurationToken) message to retrieve Video Source Configuration Options for video source configuration from device.
6. The DUT sends **GetVideoSourceConfigurationOptionsResponse** message.
7. Check that parameters for video source configuration are available according to options for video source configuration.
8. Repeat steps 5-7 for each VideoSourceConfiguration from the **GetVideoSourceConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.

- The DUT did not send **GetVideoSourceConfigurationOptionsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationOptionsResponse** message.
- The DUT did send inconsistent **GetVideoSourceConfigurationOptionsResponse** message:
- Source token of VideoSourceConfiguration does not exist in VideoSourceTokensAvailable list for this VideoSourceConfiguration in **GetVideoSourceConfigurationOptionsResponse**.
- VideoSourceConfiguration.Bounds.x is not between BoundsRange.XRange.Min and BoundsRange.XRange.Max.
- VideoSourceConfiguration.Bounds.y is not between BoundsRange.YRange.Min and BoundsRange.YRange.Max.
- VideoSourceConfiguration.Bounds.width is not between BoundsRange.WidthRange.Min and BoundsRange.WidthRange.Max.
- VideoSourceConfiguration.Bounds.height is not between BoundsRange.HeightRange.Min and BoundsRange.HeightRange.Max.

#### 5.2.8.4 PROFILES AND VIDEO SOURCE CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-2-2-4

**Specification Coverage:** Get video source configuration options, Get media profiles

**Feature Under Test:** GetProfiles, GetVideoSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfiles command and GetVideoSourceConfigurationOptions command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles from device.

4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationOptionsRequest** (ProfileToken, ConfigurationToken) message to retrieve VideoSourceConfigurationOptions for the video source configuration from device.
6. The DUT sends **GetVideoSourceConfigurationOptionsResponse** message.
7. Check that all parameters for the video source configuration from the profile are correct according to options for video source configuration.
8. Repeat steps 5-7 for other profiles with VideoSourceConfiguration from the **GetProfilesResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoSourceConfigurationOptionsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationOptionsResponse** message.
- The DUT did send inconsistent **GetVideoSourceConfigurationOptionsResponse** message:
- Source token of VideoSourceConfiguration does not exist in VideoSourceTokensAvailable list for this VideoSourceConfiguration in **GetVideoSourceConfigurationOptionsResponse**.
- VideoSourceConfiguration.Bounds.x is not between BoundsRange.XRange.Min and BoundsRange.XRange.Max.
- VideoSourceConfiguration.Bounds.y is not between BoundsRange.YRange.Min and BoundsRange.YRange.Max.
- VideoSourceConfiguration.Bounds.width is not between BoundsRange.WidthRange.Min and BoundsRange.WidthRange.Max.
- VideoSourceConfiguration.Bounds.height is not between BoundsRange.HeightRange.Min and BoundsRange.HeightRange.Max.

### 5.2.8.5 VIDEO SOURCE CONFIGURATIONS AND VIDEO SOURCES CONSISTENCY

**Test Case ID:** MEDIA-2-2-5

**Specification Coverage:** GetVideoSources, Get video source configurations

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSources

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoSourceConfigurations Command and GetVideoSources Command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of Video Source Configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourcesRequest** message to retrieve list of Video Sources from device.
6. The DUT sends **GetVideoSourcesResponse** message.
7. Check that each VideoSourceConfiguration from **GetVideoSourcesResponse** message has unique token.
8. Check that every VideoSourceConfiguration.SourceToken from the GetVideoSourceConfigurationsResponse message exists in **GetVideoSourcesResponse** message (VideoSource.token).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **GetVideoSourcesResponse** message.
- The DUT did not send valid **GetVideoSourcesResponse** message.
- The DUT return two or more VideoSources in **GetVideoSourcesResponse** message with the same token.
- The DUT returned the GetVideoSourceConfigurationsResponse message with VideoSources that were not included in the **GetVideoSourceConfigurationsResponse** message.

### 5.2.8.6 VIDEO SOURCE CONFIGURATION USE COUNT (CURRENT STATE)

**Test Case ID:** MEDIA-2-2-6

**Specification Coverage:** Get media profiles, Get video source configurations, Get video source configuration

**Feature Under Test:** GetProfiles, GetVideoSourceConfigurations, GetVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.

5. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their video source configurations from device.
6. The DUT sends **GetProfilesResponse** message.
7. Check the UseCount = usecount1 value for the first VideoSourceConfiguration (VSC1) in the list on step 3.
8. Check that there are not more than usecount1 profiles with this VideoSourceConfiguration in the list from step 6.
9. Check that UseCount value in **GetProfilesResponse** for every occurrence of this VideoSourceConfiguration is usecount1.
10. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (ConfigurationToken = the first VideoSourceConfiguration from list on step 4) message to retrieve video source configuration parameters.
11. The DUT sends **GetVideoSourceConfigurationResponse** message.
12. Check the UseCount value in **GetVideoSourceConfigurationResponse** (UseCount = usecount1).
13. Repeat steps 7-10 for all other VideoSourceConfigurations from the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- The DUT sent UseCount value which is less than amount of profiles with VideoSourceConfiguration.

- The DUT sent different UseCount values in GetProfilesResponse, GetVideoSourceConfigurationsResponse and **GetVideoSourceConfigurationResponse** messages.

### 5.2.8.7 VIDEO SOURCE CONFIGURATION USE COUNT (ADD SAME VIDEO SOURCE CONFIGURATION TO PROFILE TWICE)

**Test Case ID:** MEDIA-2-2-12

**Specification Coverage:** Get media profiles, Get media profile, Add video source configuration to a profile, Get video source configurations, Get video source configuration.

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfiguration, AddVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count after adding same video source configuration to profile twice.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **CreateProfileRequest** (Name = NewName, Token = New Profile) message to create profile.
6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (profile1)
8. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** and **RemoveVideoSourceConfigurationRequest** (ProfileToken = profile1) message to remove VideoEncoderConfiguration and VideoSourceConfiguration from profile1

9. The DUT sends **RemoveVideoEncoderConfigurationResponse** and **RemoveVideoSourceConfigurationResponse** message. The UseCount of value of the VideoSourceConfiguration is reduced by 1.
10. Execute steps 8-17
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (ConfigurationToken = first video source from the list on step 4 (VSC1), ProfileToken = New Profile) message to add VideoSourceConfiguration to the new profile.
14. The DUT sends **AddVideoSourceConfigurationResponse** message.
15. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (ConfigurationToken = VSC1) message to retrieve video source configuration parameters.
16. The DUT sends **GetVideoSourceConfigurationResponse** message.
17. Check the UseCount value in GetVideoSourceConfigurationResponse message (UseCount = usecount1+1, usecount1 is value of UseCount for VSC1 from the list on step 4). If step 6 is executed, the usecount1 could be reduced by 1.
18. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (ProfileToken = New Profile, ConfigurationToken = VSC1) message to replace video source configuration in profile.
19. The DUT sends **AddVideoSourceConfigurationResponse** message.
20. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (ConfigurationToken = VSC1) message to retrieve video source configuration parameters.
21. The DUT sends **GetVideoSourceConfigurationResponse** message.
22. Check that UseCount = usecount1+1, in **GetVideoSourceConfigurationResponse** message.
23. ONVIF Client invokes **DeleteProfileRequest** (ProfileToken = New Profile) message to remove profile with video source configuration.
24. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **AddVideoSourceConfigurationResponse** message.
- The DUT did not send valid **AddVideoSourceConfigurationResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the VideoSourceConfiguration to one more profile.
- UseCount value is changed 1 after repeated adding of the VideoSourceConfiguration in profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.8.8 VIDEO SOURCE CONFIGURATION USE COUNT (ADD DIFFERENT VIDEO SOURCE CONFIGURATIONS IN PROFILE)

**Test Case ID:** MEDIA-2-2-13

**Specification Coverage:** Get media profiles, Get media profile, Add video source configuration to a profile, Get video source configurations, Get video source configuration.

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfiguration, AddVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count after adding different video source configurations to profile.

**Pre-Requisite:** Media is supported by DUT. There are at least two VideoSourceConfigurations. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. If there is only one VideoSourceConfiguration in the **GetVideoSourceConfigurationsResponse** message go to the next test.
6. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
7. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
8. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
9. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** and **RemoveVideoSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration and VideoSourceConfiguration from profile1.
10. The DUT sends **RemoveVideoEncoderConfigurationResponse** and **RemoveVideoSourceConfigurationResponse** message. The UseCount value of the two configurations will be reduced by 1.
11. Execute steps 8-17
12. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
13. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
14. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source from the list on step 4 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
15. The DUT sends **AddVideoSourceConfigurationResponse** message.
16. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC1**) message to retrieve video source configuration parameters.
17. The DUT sends **GetVideoSourceConfigurationResponse** message.

18. Check the UseCount value in **GetVideoSourceConfigurationResponse** message. (**UseCount = usecount1+1**, usecount1 is the value of UseCount for **VSC1** from the list on step 4).If test step 6 is executed, the usecount1 could be reduced by 1.
19. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = VSC2**, where **VSC2** is other VSC from list on step 4) message to replace video source configuration in profile.
20. The DUT sends **AddVideoSourceConfigurationResponse** message.
21. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC1**) message to retrieve video source configuration parameters.
22. The DUT sends **GetVideoSourceConfigurationResponse** message.
23. Check that **UseCount = usecount1**, in **GetVideoSourceConfigurationResponse**.
24. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC2**) message to retrieve video source configuration parameters.
25. The DUT sends **GetVideoSourceConfigurationResponse** message.
26. Check that **UseCount = usecount2+1**, in **GetVideoSourceConfigurationResponse** where **usecount2** is UseCount value for **VSC2** in the list on step 4.
27. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with video source configuration.
28. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **AddVideoSourceConfigurationResponse** message.
- The DUT did not send valid **AddVideoSourceConfigurationResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.

- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the VideoSourceConfiguration to one more profile.
- UseCount value is not decreased by 1 after replacing of the VideoSourceConfiguration in profile for removed VideoSourceConfiguration and UseCount value is not increased by 1 after replacing of the VideoSourceConfiguration in profile for new VideoSourceConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.8.9 VIDEO SOURCE CONFIGURATION USE COUNT (REMOVE VIDEO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-2-2-14

**Specification Coverage:** Remove video source configuration from a profile, Get video source configurations, Get video source configuration

**Feature Under Test:** GetProfiles, GetVideoSourceConfigurations, GetVideoSourceConfiguration, RemoveVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count after removing video source configuration to profile.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profile is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.

6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
8. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** and **RemoveVideoSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration and VideoSourceConfiguration from **profile1**
9. The DUT sends **RemoveVideoEncoderConfigurationResponse** and **RemoveVideoSourceConfigurationResponse** message. The UseCount value of the two configurations will be reduced by 1.
10. Execute steps 8-13
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source from the list on step 4 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
14. The DUT sends **AddVideoSourceConfigurationResponse** message.
15. ONVIF Client invokes **RemoveVideoSourceConfigurationRequest** (**ProfileToken = New Profile**) message to remove video source configuration from NewProfileToken.
16. The DUT sends **RemoveVideoSourceConfigurationResponse** message.
17. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC1**) message to retrieve video source configuration parameters. The DUT sends **GetVideoSourceConfigurationResponse** message.
18. Check the UseCount value in **GetVideoSourceConfigurationResponse**. (**UseCount = usecount1**, usecount1 is the UseCount value for **VSC1** in the list on step 4). If test step 6 is executed, the usecount1 value could be reduced by 1.
19. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile.
20. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **AddVideoSourceConfigurationResponse** message.
- The DUT did not send valid **AddVideoSourceConfigurationResponse** message.
- The DUT did not send **RemoveVideoSourceConfigurationResponse** message.
- The DUT did not send valid **RemoveVideoSourceConfigurationResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- UseCount value is not decreased by 1 after removing of the VideoSourceConfiguration from profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.8.10 VIDEO SOURCE CONFIGURATION USE COUNT (DELETION PROFILE WITH VIDEO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-2-2-15

**Specification Coverage:** Get video source configurations, Get video source configuration, Delete media profile

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfiguration, DeleteProfile.

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count after deletion of profile with source configuration in it.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
8. ONVIF Client invokes **RemoveVideoEncoderConfiguration** and **RemoveVideoSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration and VideoSourceConfiguration from **profile1**
9. The DUT sends **RemoveVideoEncoderConfigurationResponse** and **RemoveVideoSourceConfigurationResponse** message. The UseCount values for the two configurations will be reduced by 1.
10. Execute steps 8-14
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source from the list on step 4 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
14. The DUT sends **AddVideoSourceConfigurationResponse** message.
15. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with video source configuration.
16. The DUT sends **DeleteProfileResponse** message.
17. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC1**) message to retrieve video source configuration parameters.

18. The DUT sends **GetVideoSourceConfigurationResponse** message.
19. Check the UseCount value in **GetVideoSourceConfigurationResponse** message (**UseCount = usecount1**, usecount1 is the UseCount value for **VSC1** in the list on step 4). If test step 6 is executed, the UseCount value could be reduced by 1.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send **AddVideoSourceConfigurationResponse** message.
- The DUT did not send valid **AddVideoSourceConfigurationResponse** message.
- The DUT did not send **RemoveVideoSourceConfigurationResponse** message.
- The DUT did not send valid **RemoveVideoSourceConfigurationResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- UseCount value is not decreased by 1 after deletion of the profile with the VideoSourceConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.8.11 VIDEO SOURCE CONFIGURATION USE COUNT (SET VIDEO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-2-2-16

**Specification Coverage:** Get video source configurations, Get video source configuration, Modify a video source configuration.

**Feature Under Test:** GetVideoSourceConfigurations, GetVideoSourceConfiguration, SetVideoSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Source Configuration use count after setting video source configuration parameters.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
4. The DUT sends **GetVideoSourceConfigurationsResponse** message.
5. ONVIF Client invokes **SetVideoSourceConfigurationRequest** (**token = the first Video Source Configuration from the list on step 4(VSC1), UseCount = NewUseCount**) message to set parameters for video source interval profile with video source configuration.
6. The DUT sends **SetVideoSourceConfigurationResponse** message.
7. ONVIF Client invokes **GetVideoSourceConfigurationRequest** (**ConfigurationToken = VSC1**) message to retrieve video source configuration parameters.
8. The DUT sends **GetVideoSourceConfigurationResponse** message.
9. Check that **UseCount = usecount1**, in **GetVideoSourceConfigurationResponse** where usecount1 is UseCount value for VSC1 in the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationsResponse** message.

- The DUT did not send **SetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **SetVideoSourceConfigurationResponse** message.
- The DUT did not send **GetVideoSourceConfigurationResponse** message.
- The DUT did not send valid **GetVideoSourceConfigurationResponse** message.
- UseCount value changed after trying to set UseCount value.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.9 Video Encoder Configuration

### 5.2.9.1 VIDEO ENCODER CONFIGURATIONS AND PROFILES CONSISTENCY

**Test Case ID:** MEDIA-2-3-1

**Specification Coverage:** Get media profiles, Get video encoder configurations

**Feature Under Test:** GetProfiles, GetVideoEncoderConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoEncoderConfigurations command and GetProfiles command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their video encoder configurations.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of Video Encoder Configurations from device.

6. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
7. Check that each VideoEncoderConfiguration from **GetVideoEncoderConfigurationsResponse** message has unique token.
8. Check that all VideoEncoderConfigurations from the **GetProfilesResponse** message are included in the **GetVideoEncoderConfigurationsResponse** message.
9. Check that VideoEncoderConfiguration parameters are same in **GetProfilesResponse** message and in **GetVideoEncoderConfigurationsResponse** message for each VideoEncoderConfiguration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT return two or more VideoEncoderConfigurations in **GetVideoEncoderConfigurationsResponse** message with the same ConfigurationToken.
- The DUT returned the **GetProfilesResponse** message with VideoEncoderConfigurations that were not included in the **GetVideoEncoderConfigurationsResponse** message.
- The DUT returned different parameters list and parameters values for the same VideoEncoderConfiguration in the **GetVideoEncoderConfigurationsResponse** message and in the GetProfilesResponse message.

### 5.2.9.2 VIDEO ENCODER CONFIGURATIONS AND VIDEO ENCODER CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-2-3-2

**Specification Coverage:** Get media profiles, Get video encoder configurations

**Feature Under Test:** GetVideoEncoderConfiguration, GetVideoEncoderConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoEncoderConfigurations command and GetVideoEncoderConfiguration command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of Video Encoder Configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken**) message to retrieve parameters of video encoder configuration from device.
6. The DUT sends **GetVideoEncoderConfigurationResponse** message.
7. Check that all parameters for video encoder configuration in the **GetVideoEncoderConfigurationsResponse** message and **GetVideoEncoderConfigurationResponse** message are the same.
8. Repeat steps 5-7 for each Video Encoder Configurations from the **GetVideoEncoderConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.

- The DUT did not send equal parameters for one or more VideoEncoderConfiguration in the **GetVideoEncoderConfigurationResponse** message and in the GetVideoEncoderConfigurationsResponse message.

### 5.2.9.3 VIDEO ENCODER CONFIGURATIONS AND VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-2-3-3

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration options

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetVideoEncoderConfigurations command and GetVideoEncoderConfigurationOptions command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of Video Encoder Configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationOptionsRequest** (**ConfigurationToken**) message to retrieve VideoEncoderConfigurationOptions from device.
6. The DUT sends **GetVideoEncoderConfigurationOptionsResponse** message.
7. Check that all parameters of video encoder configuration are available according to options for video encoder configuration.
8. Repeat steps 5-7 for other video encoder configurations from the **GetVideoEncoderConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT sent for at list one VideoEncoderConfiguration with inconsistent parameters:
- Encoding value from video encoder configuration is not included in options
- There is no pair “Resolution.Height, Resolution.Width” from video encoder configuration in AvailableResolution list for previously checked Encoder from options.
- Quality value for video encoder configuration is not between QualityRange.Min and QualityRange.Max from options.
- RateControl.FrameRateLimit for video encoder configuration is not between Encoding.FrameRateRange.Min and Encoding.FrameRateRange.Max for current encoding from options
- RateControl.EncodingInterval for video encoder configuration is not between EncodingIntervalRange.Min and EncodingIntervalRange.Max for current encoding from options.
- RateControl.BitrateLimit for video encoder configuration is not between BitrateRange.Min and BitrateRange.Max for current encoding from options.
- If Encoding is H.264, H264.H264Profile value does not exist in H264.H264ProfilesSupported list from options.
- If Encoding is MPEG4, MPEG4.MPEG4Profile value does not exist in MPEG4.MPEG4ProfilesSupported list from options.
- If Encoding is H264 or MPEG4, Enc.GovLength value is not between Enc.GovLengthRange.Min and Enc.GovLengthRange.Max for corresponded encoding from options.

### 5.2.9.4 PROFILES AND VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-2-3-4

**Specification Coverage:** Get media profile, Get video encoder configuration options

**Feature Under Test:** GetProfiles, GetVideoEncoderConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfiles command and GetVideoEncoderConfigurationOptions command are consistent.

**Pre-Requisite:** ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles from device.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetVideoEncoderConfigurationOptionsRequest** (**ProfileToken, ConfigurationToken**) message to retrieve VideoEncoderConfigurationOptions for video encoder configuration from the DUT.
6. The DUT sends **GetVideoEncoderConfigurationOptionsResponse** message.
7. Check that all parameters of video encoder configuration in profile are available according to options for video encoder configuration.
8. Repeat steps 5 -7 for other profiles from the **GetProfilesResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT sent for at list one VideoEncoderConfiguration with inconsistent parameters:
- Encoding value from video encoder configuration is not included in options
- There is no pair “Resolution.Height, Resolution.Width” from video encoder configuration in AvailableResolution list for previously checked Encoder from options.
- Quality value for video encoder configuration is not between QualityRange.Min and QualityRange.Max from options.
- RateControl.FrameRateLimit for video encoder configuration is not between Encoding.FrameRateRange.Min and Encoding.FrameRateRange.Max for current encoding from options
- RateControl.EncodingInterval for video encoder configuration is not between EncodingIntervalRange.Min and EncodingIntervalRange.Max for current encoding from options.
- RateControl.BitrateLimit for video encoder configuration is not between BitrateRange.min and BitrateRange.max for current encoding from options.
- If Encoding is H.264, H264.H264Profile value does not exist in H264.H264ProfilesSupported list from options.
- If Encoding is MPEG4, MPEG4.MPEG4Profile value does not exist in MPEG4.MPEG4ProfilesSupported list from options.
- If Encoding is H264 or MPEG4, Enc.GovLength value is not between Enc.GovLengthRange.Min and Enc.GovLengthRange.Max for corresponded encoding from options.

### 5.2.9.5 VIDEO ENCODER CONFIGURATION USE COUNT (CURRENT STATE)

**Test Case ID:** MEDIA-2-3-5

**Specification Coverage:** Get media profiles, Get video encoder configurations, Get video encoder configuration

**Feature Under Test:** GetProfiles, GetVideoEncoderConfigurations, GetVideoEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count.

**Pre-Requisite:** Media is supported by DUT. Profile with VideoEncoderConfiguration exists. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their video encoder configurations from device.
6. The DUT sends **GetProfilesResponse** message.
7. Check the **UseCount = usecount1** value for the first VideoEncoderConfiguration (**VEC1**) in the list on step 4.
8. Check that there are not more than usecount1 profiles with this VideoEncoderConfiguration in the list from step 6.
9. Check that **UseCount** value in **GetProfilesResponse** for every occurrence of this VideoEncoderConfiguration is **usecount1**.
10. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
11. The DUT sends **GetVideoEncoderConfigurationResponse** message.
12. Check the UseCount value in **GetVideoEncoderConfigurationResponse** (**UseCount = usecount1**).
13. Repeat steps 7-10 for all other VideoEncoderConfigurations from the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- The DUT sent UseCount value which is less than amount of profiles with VideoEncoderConfiguration.
- The DUT sent different UseCount values in GetProfilesResponse, GetVideoEncoderConfigurationsResponse and **GetVideoEncoderConfigurationResponse** messages.

### 5.2.9.6 VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODINGS

**Test Case ID:** MEDIA-2-3-12

**Specification Coverage:** None

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify whether all supported encodings can be set.

**Pre-Requisite:** Media is supported by DUT. Media Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke **GetVideoEncoderConfigurationsRequest** message to retrieve all DUT video encoder configurations.
4. Verify the **GetVideoEncoderConfigurationsResponse** message from the DUT.
5. ONVIF Client will invoke **GetVideoEncoderConfigurationOptionsRequest** message (ConfigurationToken = “Token1”, where “Token1” is a first video encoder configuration token from GetVideoEncoderConfigurationsResponse message, no ProfileToken) to retrieve supported video encoder configuration options.
6. Verify the GetVideoEncoderConfigurationOptionsResponse message from the DUT.
7. ONVIF Client will invoke **SetVideoEncoderConfigurationRequest** message (Configuration.token = “Token1”, where “Token1” is a first video encoder configuration token from GetVideoEncoderConfigurationsResponse message, Configuration.Encoding = “codec1”, where “codec1” is a first supported codec from GetVideoEncoderConfigurationOptionsResponse message, other Configuration parameters that are applicable for selected Encoding) to change video encoder configuration settings.
8. Verify the **SetVideoEncoderConfigurationResponse** message from the DUT.
9. ONVIF Client will invoke **GetVideoEncoderConfigurationRequest** message (ConfigurationToken = “Token1”, where “Token1” is a first video encoder configuration token from GetVideoEncoderConfigurationsResponse message) to retrieve DUT video encoder configuration for specified token.
10. Verify the **GetVideoEncoderConfigurationResponse** message from the DUT. Check that video encoder setting was applied.
11. Repeat steps 7-10 for the rest Video Encodings supported by selected configuration.
12. If number of Video encoder configurations in **GetVideoEncoderConfigurationsResponse** message is more than one, repeat steps 5-11 for the last video encoder configuration token from **GetVideoEncoderConfigurationsResponse** message.
13. If number of Video encoder configurations in **GetVideoEncoderConfigurationsResponse** message is more than two, Repeat steps 5-11 for the video encoder configuration token between the first and the last VEC tokens from **GetVideoEncoderConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **SetVideoEncoderConfigurationResponse** message.
- The DUT sent **GetVideoEncoderConfigurationResponse** message with Configuration.Encoding other then was set up with **SetVideoEncoderConfigurationRequest** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.9.7 VIDEO ENCODER CONFIGURATION USE COUNT (ADD SAME VIDEO ENCODER CONFIGURATION TO PROFILE TWICE)

**Test Case ID:** MEDIA-2-3-13

**Specification Coverage:** Add video encoder configuration to a profile, Get video encoder configurations, Get video encoder configuration.

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfiguration, AddVideoEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count after adding same video encoder configuration to profile twice.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles command

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.

3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
6. The DUT sends **GetVideoSourceConfigurationsResponse** message.
7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration from **profile1**
11. The DUT sends **RemoveVideoEncoderConfigurationResponse** message.
12. Execute steps 10-26.
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleVideoSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoSourceConfigurations for new profile.
16. The DUT sends **GetCompatibleVideoSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source configuration from the list on step 11 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
18. The DUT sends **AddVideoSourceConfigurationResponse** message.
19. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoEncoderConfigurations for new profile with VideoSourceConfiguration.
20. The DUT sends **GetCompatibleVideoEncoderConfigurationsResponse** message. If response does not contain any VideoEncoderConfiguration, repeat steps 12-15 with next

video source configuration from step 11. If there are no other video source configuration, skip other steps and go to the next test.

21. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ConfigurationToken = first video encoder configuration from the list on step 15 (VEC1), ProfileToken = New Profile**) message to add VideoEncoderConfiguration to the new profile.
22. The DUT sends **AddVideoEncoderConfigurationResponse** message.
23. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
24. The DUT sends **GetVideoEncoderConfigurationResponse** message.
25. Check the UseCount value in **GetVideoEncoderConfigurationResponse** message (**UseCount = usecount1+1**, usecount1 is the UseCount value for VEC1 from the list on step 4). If test step 8 is executed, the usecount1 value could be reduced by 1.
26. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = VEC1**) message to replace video encoder configuration in profile.
27. The DUT sends **AddVideoEncoderConfigurationResponse** message.
28. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
29. The DUT sends **GetVideoEncoderConfigurationResponse** message.
30. Check that **UseCount = usecount1+1**, in **GetVideoEncoderConfigurationResponse** message.
31. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleVideoEncoderConfigurationsResponse** message.

- The DUT did not send valid **GetCompatibleVideoSourceConfigurationsResponse** message.
- The DUT did not send valid **AddVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the VideoEncoderConfiguration to one more profile.
- UseCount value is changed after repeated adding of the VideoEncoderConfiguration in profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.9.8 VIDEO ENCODER CONFIGURATION USE COUNT (ADD DIFFERENT VIDEO ENCODER CONFIGURATIONS IN PROFILE)

**Test Case ID:** MEDIA-2-3-14

**Specification Coverage:** Add video encoder configuration to a profile, Get video encoder configurations, Get video encoder configuration.

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfiguration, AddVideoEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count after adding different video encoder configurations to profile.

**Pre-Requisite:** Media is supported by DUT. There are at least two VideoEncoderConfigurations. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles command

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.

4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. If there is only one VideoEncoderConfiguration in the **GetVideoEncoderConfigurationsResponse** message go to the next test.
6. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
7. The DUT sends **GetVideoSourceConfigurationsResponse** message.
8. ONVIF Client invokes **CreateProfileRequest** (**Name = New Name, Token = New Profile**) message to create profile.
9. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
10. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
11. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration from **profile1**
12. The DUT sends **RemoveVideoEncoderConfigurationResponse** message
13. Execute steps 10-29
14. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
15. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
16. ONVIF Client invokes **GetCompatibleVideoSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoSourceConfigurations for new profile.
17. The DUT sends **GetCompatibleVideoSourceConfigurationsResponse** message.
18. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source configuration from the list on step 11 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
19. The DUT sends **AddVideoSourceConfigurationResponse** message.
20. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoEncoderConfigurations for new profile with VideoSourceConfiguration.
21. The DUT sends **GetCompatibleVideoEncoderConfigurationsResponse** message. If response contains number of VideoEncoderConfiguration less than 2, repeat steps 12-

15 with next video source configuration from step 11. If there are no other video source configuration, skip other steps and go to the next test.

22. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ConfigurationToken = first video encoder configuration from the list on step 15 (VEC1), ProfileToken = New Profile**) message to add VideoEncoderConfiguration to the new profile.
23. The DUT sends **AddVideoEncoderConfigurationResponse** message.
24. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
25. The DUT sends **GetVideoEncoderConfigurationResponse** message.
26. Check the UseCount value in **GetVideoEncoderConfigurationResponse** (**UseCount = usecount1+1**, usecount1 is value of UseCount for VEC1 from the list on step 4).If step 8 is executed; the usecount1 could be reduced by 1.
27. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = VEC2, where VEC2 is other VEC from list on step 15**) message to replace video encoder configuration in profile.
28. The DUT sends **AddVideoEncoderConfigurationResponse** message.
29. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
30. The DUT sends **GetVideoEncoderConfigurationResponse** message.
31. Check that **UseCount = usecount1**, in **GetVideoEncoderConfigurationResponse**.
32. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC2**) message to retrieve video encoder configuration parameters.
33. The DUT sends **GetVideoEncoderConfigurationResponse** message.
34. Check that **UseCount = usecount2+1**, in **GetVideoEncoderConfigurationResponse** where usecount2 is UseCount value for VEC2 in the list on step 4.
35. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleVideoSourceConfigurationsResponse** message.
- UseCount value is not increased by 1 after adding of the VideoEncoderConfiguration to one more profile.
- UseCount value is not decreased by 1 after replacing of the VideoEncoderConfiguration in profile for removed VideoEncoderConfiguration and UseCount value is not increased by 1 after replacing of the VideoEncoderConfiguration in profile for new VideoEncoderConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.2.9.9 VIDEO ENCODER CONFIGURATION USE COUNT (REMOVE VIDEO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-2-3-15

**Specification Coverage:** Remove video encoder configuration from a profile, Get video encoder configurations, Get video encoder configuration.

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfiguration, RemoveVideoEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count after removing video encoder configuration from a profile.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
6. The DUT sends **GetVideoSourceConfigurationsResponse** message.
7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration from profile1
11. The DUT sends **RemoveVideoEncoderConfigurationResponse** message
12. Execute steps 10-23
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleVideoSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoSourceConfigurations for new profile.
16. The DUT sends **GetCompatibleVideoSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source configuration from the list on step 11 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
18. The DUT sends **AddVideoSourceConfigurationResponse** message.
19. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoEncoderConfigurations for new profile with VideoSourceConfiguration.

20. The DUT sends **GetCompatibleVideoEncoderConfigurationsResponse** message. If response does not contain any VideoEncoderConfiguration, repeat steps 12-15 with next video source configuration from step 11. If there are no other video source configuration, skip other steps and go to the next test.
21. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ConfigurationToken = first video encoder configuration from the list on step 15 (VEC1), ProfileToken = New Profile**) message to add VideoEncoderConfiguration to the new profile.
22. The DUT sends **AddVideoEncoderConfigurationResponse** message.
23. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** (**ProfileToken = New Profile**) message to remove video encoder configuration from NewProfileToken.
24. The DUT sends **RemoveVideoEncoderConfigurationResponse** message.
25. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
26. The DUT sends **GetVideoEncoderConfigurationResponse** message.
27. Check that **UseCount = usecount1**, in **GetVideoEncoderConfigurationResponse** where usecount1 is UseCount value for VEC1 in the list on step 4.If step 8 is executed, the usecount1 could be reduced by 1.
28. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **RemoveVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleVideoSourceConfigurationsResponse** message.

- UseCount value is not decreased by 1 after removing of the VideoEncoderConfiguration from profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.9.10 VIDEO ENCODER CONFIGURATION USE COUNT (PROFILE DELETION WITH VIDEO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-2-3-16

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration, Delete media profile

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfiguration, DeleteProfile

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count after deletion of the profile with video encoder configuration

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetVideoSourceConfigurationsRequest** message to retrieve list of video source configurations from device.
6. The DUT sends **GetVideoSourceConfigurationsResponse** message.
7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.

8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveVideoEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove VideoEncoderConfiguration from profile1
11. The DUT sends **RemoveVideoEncoderConfigurationResponse** message
12. Execute steps 10-23
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleVideoSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoSourceConfigurations for new profile.
16. The DUT sends **GetCompatibleVideoSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddVideoSourceConfigurationRequest** (**ConfigurationToken = first video source configuration from the list on step 11 (VSC1), ProfileToken = New Profile**) message to add VideoSourceConfiguration to the new profile.
18. The DUT sends **AddVideoSourceConfigurationResponse** message.
19. ONVIF Client invokes **GetCompatibleVideoEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible VideoEncoderConfigurations for new profile with VideoSourceConfiguration.
20. The DUT sends **GetCompatibleVideoEncoderConfigurationsResponse** message. If response does not contain any VideoEncoderConfiguration, repeat steps 12-15 with next video source configuration from step 11. If there are no other video source configuration, skip other steps and go to the next test.
21. ONVIF Client invokes **AddVideoEncoderConfigurationRequest** (**ConfigurationToken = first video encoder configuration from the list on step 15 (VEC1), ProfileToken = New Profile**) message to add VideoEncoderConfiguration to the new profile.
22. The DUT sends **AddVideoEncoderConfigurationResponse** message.
23. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with video encoder configuration.
24. The DUT sends **DeleteProfileResponse** message.

25. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
26. The DUT sends **GetVideoEncoderConfigurationResponse** message.
27. Check that **UseCount = usecount1**, in **GetVideoEncoderConfigurationResponse** where usecount1 is UseCount value for VEC1 in the list on step 4. If step 8 is executed, the usecount1 could be reduced by 1.
28. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **DeleteProfileResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleVideoSourceConfigurationsResponse** message.
- UseCount value is not decreased by 1 after deletion of the profile with the VideoEncoderConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.9.11 VIDEO ENCODER CONFIGURATION USE COUNT (SET VIDEO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-2-3-17

**Specification Coverage:** Get video encoder configurations, Get video encoder configuration, Modify a video encoder configuration.

**Feature Under Test:** GetVideoEncoderConfigurations, GetVideoEncoderConfiguration, SetVideoEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Video Encoder Configuration use count after setting video encoder configuration parameters.

**Pre-Requisite:** Media is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetVideoEncoderConfigurationsRequest** message to retrieve list of video encoder configurations from device.
4. The DUT sends **GetVideoEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **SetVideoEncoderConfigurationRequest** (**token = the first Video Encoder Configuration from the list on step 4(VEC1), UseCount = NewUseCount**) message to set parameters for video encoder configuration.
6. The DUT sends **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetVideoEncoderConfigurationRequest** (**ConfigurationToken = VEC1**) message to retrieve video encoder configuration parameters.
8. The DUT sends **GetVideoEncoderConfigurationResponse** message.
9. Check that **UseCount = usecount1**, in **GetVideoEncoderConfigurationResponse** where usecount1 is UseCount value for VEC1 in the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetVideoEncoderConfigurationsResponse** message.

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send **RemoveVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **RemoveVideoEncoderConfigurationResponse** message.
- The DUT did not send **GetVideoEncoderConfigurationResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.
- UseCount value is changed after trying to set UseCount value.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.2.9.12 VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODINGS (ALL VIDEO ENCODER CONFIGURATIONS)

**Test Case ID:** MEDIA-2-3-18

**Specification Coverage:** None

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify whether all supported encodings can be set.

**Pre-Requisite:** Media is supported by DUT. Media Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetVideoEncoderConfigurationsRequest** message to retrieve all DUT video encoder configurations.
4. Verify the **GetVideoEncoderConfigurationsResponse** message from the DUT.
5. ONVIF Client will invoke **GetVideoEncoderConfigurationOptionsRequest** message (**ConfigurationToken = “Token1”, where “Token1” is a first video encoder**

**configuration token from GetVideoEncoderConfigurationsResponse message, no ProfileToken**) to retrieve supported video encoder configuration options.

6. Verify the **GetVideoEncoderConfigurationOptionsResponse** message from the DUT.
7. ONVIF Client will invoke **SetVideoEncoderConfigurationRequest** message (**Configuration.token = "Token1", where "Token1" is a first video encoder configuration token from GetVideoEncoderConfigurationsResponse message, Configuration.Encoding = "codec1", where "codec1" is a first supported codec from GetVideoEncoderConfigurationOptionsResponse message, other Configuration parameters that are applicable for selected Encoding**) to change video encoder configuration settings.
8. Verify the **SetVideoEncoderConfigurationResponse** message from the DUT.
9. ONVIF Client will invoke **GetVideoEncoderConfigurationRequest** message (**ConfigurationToken = "Token1", where "Token1" is a first video encoder configuration token from GetVideoEncoderConfigurationsResponse message**) to retrieve DUT video encoder configuration for specified token.
10. Verify the **GetVideoEncoderConfigurationResponse** message from the DUT. Check that video encoder setting was applied.
11. Repeat steps 7-10 for the rest Video Encodings supported by selected configuration.
12. Repeat steps 5-11 for the rest Video Encoder Configuration supported by the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetVideoEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetVideoEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **SetVideoEncoderConfigurationResponse** message.
- The DUT sent **GetVideoEncoderConfigurationResponse** message with Configuration.Encoding other then was set up with SetVideoEncoderConfigurationRequest message.
- The DUT did not send valid **GetVideoEncoderConfigurationResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3 Audio Configuration

### 5.3.1 GET AUDIO SOURCE CONFIGURATION OPTIONS

**Test Case ID:** MEDIA-3-1-7

**Specification Coverage:** Get audio source configuration options

**Feature Under Test:** GetAudioSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of GetAudioSourceConfigurationOptions command with different parameters.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (empty) message.
4. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
5. Verify **GetAudioSourceConfigurationOptionsResponse** message.
6. ONVIF Client invokes **GetProfilesRequest** message to get list of existing profiles from device.
7. The DUT sends **GetProfilesResponse** message.
8. For each profile (*profile*) in **GetProfilesResponse**:
   - 8.1. ONVIF Client invokes **GetCompatibleAudioSourceConfigurations** message (ProfileToken = *profile*.@token) to get list of audio source configurations compatible with a profile.
   - 8.2. The DUT sends **GetCompatibleAudioSourceConfigurationsResponse** message with parameters
     - Configurations list =: *audioSourceConfList*

8.3. If *audioSourceConfList* is not empty, go to step 9.

9. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (ConfigurationToken = *audioSourceConfList*[0].@token) message.
10. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
11. Verify **GetAudioSourceConfigurationOptionsResponse** message.
12. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (ProfileToken = *profile*.@token) message.
13. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
14. Verify **GetAudioSourceConfigurationOptionsResponse** message.
15. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (ProfileToken = *profile*.@token, ConfigurationToken = *audioSourceConfList*[0].@token) message.
16. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
17. Verify **GetAudioSourceConfigurationOptionsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetAudioSourceConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationOptionsResponse** message for one or more **GetAudioSourceConfigurationOptionsRequest** messages.

## 5.3.2 G.711 AUDIO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-3-1-14

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration, Get audio encoder configuration options, Modify audio encoder configurations

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT G.711 Audio Encoder Configuration Setting

**Pre-Requisite:** Audio is supported by DUT

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurations** request.
4. DUT sends the list of supported audio encoder configurations in **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** Request (Audio Encoder Configuration token) to retrieve audio encoder configuration options for the specified audio encoder configuration.
6. DUT sends the range of configurable values for the received audio encoder configuration in the **GetAudioEncoderConfigurationOptionsResponse** message.
7. Test steps 5 & 6 have to be repeated for all audio encoder configurations until ONVIF Client finds a audio encoder configuration with G.711 encoding support
8. ONVIF Client invokes **SetAudioEncoderConfiguration** request with G.711 configuration values outside the range defined in the **GetAudioEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
9. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
10. ONVIF Client verifies the SOAP fault message sent by DUT.
11. ONVIF Client invokes **SetAudioEncoderConfiguration** request (Encoding = "G711", Bit Rate = r1, Sample Rate = r2 and force persistence = false). These values will be taken from the **GetAudioEncoderConfigurationOptionsResponse** message.

12. DUT modifies G.711 audio encoder configuration and responds with **SetAudioEncoderConfigurationResponse** message indicating success.
13. ONVIF Client verifies the G.711 audio Encoder Configuration settings on DUT by invoking **GetAudioEncoderConfiguration** request.
14. DUT sends modified G.711 audio Encoder Configuration in the **GetAudioEncoderConfigurationResponse** message (Encoding = “G711”, Bit Rate = r1, Sample Rate = r2).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.
- DUT doesn’t support G.711 audio encoding.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetAudioEncoderConfiguration** request.
- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not modify G.711 Audio Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.3 G.726 AUDIO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-3-1-15

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration, Get audio encoder configuration options, Modify audio encoder configurations

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT G.726 Audio Encoder Configuration Setting

**Pre-Requisite:** Audio is supported by DUT and G.726 is implemented by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurations** request.
4. DUT sends the list of supported audio encoder configurations in **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** Request (Audio Encoder Configuration token) to retrieve audio encoder configuration options for the specified audio encoder configuration.
6. DUT sends the range of configurable values for the received audio encoder configuration in the **GetAudioEncoderConfigurationOptionsResponse** message.
7. Test steps 5 & 6 have to be repeated for all audio encoder configurations until ONVIF Client finds a audio encoder configuration with G.726 encoding support
8. ONVIF Client invokes **SetAudioEncoderConfiguration** request with G.726 configuration values outside the range defined in the **GetAudioEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
9. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
10. ONVIF Client verifies the SOAP fault message sent by DUT.
11. ONVIF Client invokes **SetAudioEncoderConfiguration** request (Encoding = "G726", Bit Rate = r1, Sample Rate = r2 and force persistence = false). These values will be taken from the **GetAudioEncoderConfigurationOptionsResponse** message.
12. DUT modifies G.726 audio encoder configuration and responds with **SetAudioEncoderConfigurationResponse** message indicating success.
13. ONVIF Client verifies the G.726 audio Encoder Configuration settings on DUT by invoking **GetAudioEncoderConfiguration** request.
14. DUT sends the modified G.726 audio Encoder Configuration in the **GetAudioEncoderConfigurationResponse** message (Encoding = "G726", Bit Rate = r1, Sample Rate = r2).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.
- DUT doesn't support G.726 audio encoding.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetAudioEncoderConfiguration** request.
- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not modify G.726 Audio Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.4 AAC AUDIO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-3-1-16

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration, Get audio encoder configuration options, Modify audio encoder configurations

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT AAC Audio Encoder Configuration Setting

**Pre-Requisite:** Audio is supported by DUT and AAC is implemented by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurations** request.

4. DUT sends the list of supported audio encoder configurations in **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** Request (Audio Encoder Configuration token) to retrieve audio encoder configuration options for the specified audio encoder configuration.
6. DUT sends the range of configurable values for the received audio encoder configuration in the **GetAudioEncoderConfigurationOptionsResponse** message.
7. Test steps 5 & 6 have to be repeated for all audio encoder configurations until ONVIF Client finds a audio encoder configuration with AAC encoding support
8. ONVIF Client invokes **SetAudioEncoderConfiguration** request with AAC configuration values outside the range defined in the **GetAudioEncoderConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
9. DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
10. ONVIF Client verifies the SOAP fault message sent by DUT.
11. ONVIF Client invokes **SetAudioEncoderConfiguration** request (Encoding = "AAC", Bit Rate = r1, Sample Rate = r2 and force persistence = false). These values will be taken from the **GetAudioEncoderConfigurationOptionsResponse** message.
12. DUT modifies AAC audio encoder configuration and responds with **SetAudioEncoderConfigurationResponse** message indicating success.
13. ONVIF Client verifies the AAC audio Encoder Configuration settings on DUT by invoking **GetAudioEncoderConfiguration** request.
14. DUT sends modified AAC audio Encoder Configuration in the **GetAudioEncoderConfigurationResponse** message (Encoding = "AAC", Bit Rate = r1, Sample Rate = r2).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.

- DUT doesn't support AAC audio encoding.
- DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetAudioEncoderConfiguration** request.
- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not modify AAC Audio Encoder Configuration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.5 GET AUDIO SOURCE CONFIGURATION – INVALID CONFIGURATIONTOKEN

**Test Case ID:** MEDIA-3-1-17

**Specification Coverage:** Get audio source configuration

**Feature Under Test:** GetAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of GetAudioSourceConfiguration command in case of invalid ConfigurationToken.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (invalid ConfigurationToken) message.
4. The DUT will generate a SOAP 1.2 fault message (**InvalidArgVal/NoConfig**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** Other faults than specified in the test are acceptable but specified is preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.6 GET AUDIO SOURCE CONFIGURATION OPTIONS – INVALID PROFILETOKEN

**Test Case ID:** MEDIA-3-1-18

**Specification Coverage:** Get audio source configuration options

**Feature Under Test:** GetAudioSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of GetAudioSourceConfigurationOptions command in case of invalid ProfileToken.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (invalid ProfileToken) message.
4. The DUT will generate a SOAP 1.2 fault message (**InvalidArgVal/NoProfile**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** Other faults than specified in the test are acceptable but specified is preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.7 GET AUDIO SOURCE CONFIGURATION OPTIONS – INVALID CONFIGURATION TOKEN

**Test Case ID:** MEDIA-3-1-19

**Specification Coverage:** Get audio source configuration options

**Feature Under Test:** GetAudioSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of GetAudioSourceConfigurationOptions command in case of invalid ConfigurationToken.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (invalid ConfigurationToken) message.
4. The DUT will generate a SOAP 1.2 fault message (**InvalidArgVal/NoConfig**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.

- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** Other faults than specified in the test are acceptable but specified is preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.8 SET AUDIO SOURCE CONFIGURATION – INVALID TOKEN

**Test Case ID:** MEDIA-3-1-20

**Specification Coverage:** Modify an audio source configuration

**Feature Under Test:** SetAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of SetAudioSourceConfiguration command in case of invalid ConfigurationToken.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **SetAudioSourceConfigurationRequest** (invalid ConfigurationToken) message.
4. The DUT will generate a SOAP 1.2 fault message (**InvalidArgVal/NoConfig**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** Other faults than specified in the test are acceptable but specified is preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.9 SET AUDIO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-3-1-21

**Specification Coverage:** Create media profile, Add audio source configuration to a profile, Add audio encoder configuration to a profile, Remove audio source configuration from a profile, Remove audio encoder configuration from a profile, Delete media profile, Get audio source configurations, Get audio encoder configurations, Get compatible audio encoder configurations, Set audio encoder configuration

**Feature Under Test:** SetAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Audio Encoder Configuration Operations

**Pre-Requisite:** Audio is supported by DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreateProfileRequest** message (ProfileToken = 'testprofileX') to create new profile.
4. Verify **CreateProfileResponse** message or SOAP 1.2 fault message (**Action/ MaxNVTProfiles**) from the DUT. If fault was received execute Annex A.3.
5. ONVIF Client will invoke **GetCompatibleAudioSourceConfigurationsRequest** message (ProfileToken = 'testprofileX') to retrieve the list of audio source configurations compatible with profile.
6. ONVIF Client verifies the list of audio source configurations sent by DUT.
7. ONVIF Client invokes **AddAudioSourceConfigurationRequest** message (ProfileToken = 'testprofileX', ConfigurationToken as one of the tokens received in the **GetCompatibleAudioSourceConfigurationsResponse** message) to add audio source configuration to profile.

8. DUT adds the audio source configuration to the profile and sends the response.
9. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurationsRequest** message (ProfileToken = 'testprofileX') to retrieve audio encoder configurations compatible with profile.
10. DUT sends the list of audio encoder configurations compatible with the received media profile token.
11. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** message (ProfileToken = 'testprofileX', ConfigurationToken as one of the tokens received in the **GetCompatibleAudioEncoderConfigurationsResponse** message) to add audio encoder configuration to profile.
12. DUT adds the audio encoder configuration to the profile and sends the response.
13. ONVIF Client invokes **GetAudioEncoderConfigurationOptionsRequest** (ProfileToken = 'testprofileX') request to retrieve audio encoder options for specified profile.
14. DUT sends the audio encoder configuration options which could be applied to audio encoder from specified profile.
15. ONVIF Client invokes **SetAudioEncoderConfigurationRequest** message (ConfigurationToken, Encoding = [other than current], Bitrate = [other than current], SampleRate = [other than current], ForcePersistence = false, where all values was taken from audio encoder configuration options) to change
16. DUT sends **SetAudioEncoderConfigurationResponse** message.
17. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** message (ConfigurationToken) to get new audio encoder configuration parameters.
18. DUT sends **GetAudioEncoderConfigurationResponse** message with parameters specified in set request.
19. ONVIF Client checks that Audio configuration in **GetAudioEncoderConfigurationResponse** message is the same as in **SetAudioEncoderConfigurationRequest** message.
20. If used created Media Profile then ONVIF Client invokes **DeleteProfileRequest** message (ProfileToken = 'testprofileX'). Otherwise ONVIF client skip rest steps and restore profile settings.
21. DUT deletes the media profile and sends the response.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **CreateProfileResponse** message.
- The DUT did not send valid **GetCompatibleAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT sent Audio configuration in **GetAudioEncoderConfigurationResponse** message different with **SetAudioEncoderConfigurationRequest** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.10 AUDIO SOURCE CONFIGURATION

**Test Case ID:** MEDIA-3-1-22

**Specification Coverage:** Create media profile, Add audio source configuration to a profile, Remove audio source configuration from a profile, Delete media profile, Get audio sources, Get audio source configurations, Get audio source configuration, Get compatible audio source configurations, Get audio source configuration options, Modify an audio source configuration.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Audio Source Configuration Operations

**Pre-Requisite:** Audio is supported by the DUT

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **CreateProfile** request with ProfileToken = 'testprofileX'.
4. Verify CreateProfileResponse message or SOAP 1.2 fault message (**Action/ MaxNVTProfiles**) from the DUT. If fault was received execute Annex A.3.
5. ONVIF Client invokes **GetAudioSources** request to retrieve the existing audio sources of DUT.
6. ONVIF Client verifies the list of audio sources sent by DUT.
7. ONVIF Client invokes **GetAudioSourceConfigurations** request to retrieve the list of audio source configurations supported by the DUT.
8. ONVIF Client verifies the list of audio source configurations sent by DUT.
9. ONVIF Client invokes **GetCompatibleAudioSourceConfigurations** request with 'testprofileX' as ProfileToken.
10. The DUT sends the list of audio source configurations compatible with the received media profile token.
11. ONVIF Client invokes **AddAudioSourceConfiguration** request message with ProfileToken as 'testprofileX' and ConfigurationToken as one of the tokens in **GetCompatibleAudioSourceConfigurationsResponse**.
12. The DUT adds the audio source configuration to the media profile and sends the response.
13. ONVIF Client invokes **GetAudioSourceConfigurationOptions** request with ConfigurationToken as the same token sent in the **AddAudioSourceConfiguration** request.
14. The DUT sends the configurable options supported for the received audio source configuration.
15. ONVIF Client invokes **SetAudioSourceConfiguration** request with audio source configuration values outside the range defined in **GetAudioSourceConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
16. The DUT send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
17. ONVIF Client verifies the SOAP fault message sent by DUT.
18. ONVIF Client invokes **SetAudioSourceConfiguration** request with audio source configuration values as defined in **GetAudioSourceConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.

19. The DUT modifies the audio source configuration and sends the **SetAudioSourceConfigurationResponse** message indicating success.
20. ONVIF Client verifies the modified audio source configuration by invoking the **GetAudioSourceConfiguration** request.
21. The DUT sends the modified audio source configuration in **GetAudioSourceConfigurationResponse**.
22. ONVIF Client invokes **RemoveAudioSourceConfiguration** request with ProfileToken as 'testprofileX'.
23. The DUT removes the audio source configuration token from media profile and sends the response.
24. If profile with profile token = testprofileX was created during test execution, then ONVIF Client invokes **DeleteProfile** request with ProfileToken as 'testprofileX'. Otherwise, ONVIF client skip the rest steps and restore profile settings.
25. The DUT deletes the media profile and sends the response.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateProfileResponse** message.
- The DUT did not send valid **GetAudioSourcesResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetCompatibleAudioSourceConfigurationsResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **GetAudioSourceConfigurationOptionsResponse** message.
- The DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetAudioSourceConfiguration** request.
- The DUT did not send **SetAudioSourceConfigurationResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not modify audio source configuration correctly.

- The DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- The DUT did not send **DeleteProfileResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

**NOTE:** If profile was deleted during Annex A.3 execution, ONVIF Client restores the deleted profile and profile settings.

## 5.3.11 AUDIO ENCODER CONFIGURATION

**Test Case ID:** MEDIA-3-1-23

**Specification Coverage:** Create media profile, Add audio source configuration to a profile, Add audio encoder configuration to a profile, Remove audio source configuration from a profile, Remove audio encoder configuration from a profile, Delete media profile, Get audio source configurations, Get audio encoder configurations, Get compatible audio encoder configurations.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Audio Encoder Configuration Operations

**Pre-Requisite:** Audio is supported by the DUT

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreateProfile** request with ProfileToken = ‘testprofileX’.
4. Verify **CreateProfileResponse** message or SOAP 1.2 fault message (**Action/ MaxNVTProfiles**) from the DUT. If fault was received, execute Annex A.3.
5. ONVIF Client will invoke **GetAudioSourceConfigurations** request to retrieve the list of audio source configurations supported by DUT.
6. ONVIF Client verifies the list of audio source configurations sent by DUT.
7. ONVIF Client invokes **AddAudioSourceConfiguration** request message with ProfileToken as ‘testprofileX’ and ConfigurationToken as one of the tokens received in the **GetAudioSourceConfigurationsResponse**.

8. The DUT adds the audio source configuration to the profile and sends the response.
9. ONVIF Client will invoke **GetAudioEncoderConfigurations** request to retrieve the list of audio encoder configurations supported by DUT.
10. ONVIF Client verifies the list of audio encoder configurations sent by DUT.
11. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurations** request with 'testprofileX' as ProfileToken.
12. The DUT sends the list of audio encoder configurations compatible with the received media profile token.
13. ONVIF Client invokes **AddAudioEncoderConfiguration** request message with ProfileToken as 'testprofileX' and ConfigurationToken as one of the tokens received in the **GetCompatibleAudioEncoderConfigurationsResponse**.
14. The DUT adds the audio encoder configuration to the profile and sends the response.
15. ONVIF Client invokes **RemoveAudioEncoderConfiguration** request with ProfileToken as 'testprofileX'.
16. The DUT removes the audio encoder configuration token from media profile and sends the response.
17. ONVIF Client invokes **RemoveAudioSourceConfiguration** request with ProfileToken as 'testprofileX'.
18. The DUT removes the audio source configuration token from media profile and sends the response.
19. If profile with profile token = testprofileX was created during test execution, then ONVIF Client invokes **DeleteProfile** request with ProfileToken as 'testprofileX'. Otherwise, ONVIF client skip rest steps and restore profile settings.
20. The DUT deletes the media profile and sends the response.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateProfileResponse** message.

- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send **GetCompatibleAudioEncoderConfigurationsResponse** message.
- The DUT did not send **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send **RemoveAudioEncoderConfigurationResponse** message.
- The DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- DUT did not send **DeleteProfileResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

**NOTE:** If profile was deleted during Annex A.3 execution, ONVIF Client restores the deleted profile and profile settings.

## 5.3.12 Audio Source Configuration

### 5.3.12.1 AUDIO SOURCE CONFIGURATIONS AND PROFILES CONSISTENCY

**Test Case ID:** MEDIA-3-2-1

**Specification Coverage:** Get media profiles, Get audio source configurations

**Feature Under Test:** GetProfiles, GetAudioSourceConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioSourceConfigurations command and GetProfiles command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their configurations.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
6. The DUT sends **GetAudioSourceConfigurationsResponse** message.
7. Check that each AudioSourceConfiguration from **GetAudioSourceConfigurationsResponse** message has unique token.
8. Check that each AudioSourceConfiguration from the **GetProfilesResponse** message are included in the **GetAudioSourceConfigurationsResponse** message.
9. Check that AudioSourceConfiguration parameters are same in the **GetProfilesResponse** message and in the **GetAudioSourceConfigurationsResponse** message for each AudioSourceConfiguration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT return two or more AudioSourceConfiguration in **GetAudioSourceConfigurationsResponse** message with the same ConfigurationToken.
- The DUT returned the **GetProfilesResponse** message with AudioSourceConfigurations that were not included in the **GetAudioSourceConfigurationsResponse** message.
- The DUT returned different parameters list and parameters values in the **GetAudioSourceConfigurationsResponse** message and in the **GetProfilesResponse** message for the same AudioSourceConfiguration.

### 5.3.12.2 AUDIO SOURCE CONFIGURATIONS AND AUDIO SOURCE CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-3-2-2

**Specification Coverage:** Get audio source configurations, Get audio source configuration

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioSourceConfigurations command and GetAudioSourceConfiguration command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (ConfigurationToken) message to retrieve parameters of Audio Source Configuration from device.
6. The DUT sends **GetAudioSourceConfigurationResponse** message.
7. Verify that all parameters and their values for AudioSourceConfigurationToken from **GetAudioSourceConfigurationsResponse** message and **GetAudioSourceConfigurationResponse** message are same.
8. Repeat steps 5-7 for other AudioSourceConfigurations from the **GetAudioSourceConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- The DUT did not send equal parameters for AudioSourceConfiguration in the **GetAudioSourceConfigurationResponse** message and in the **GetAudioSourceConfigurationsResponse** message.

### 5.3.12.3 AUDIO SOURCE CONFIGURATIONS AND AUDIO SOURCE CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-2-3

**Specification Coverage:** Get audio source configurations, Get audio source configuration options

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioSourceConfigurations command and AudioSourceConfigurationOptions command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (ConfigurationToken) message to retrieve available audio source configuration options from device.

6. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
7. Check that InputTokensAvailable list from the **GetAudioSourceConfigurationOptionsResponse** message contains only unique tokens.
8. Check that SourceToken for the audio source configuration from GetAudioSourceConfigurationsResponse message exists in InputTokensAvailable list from the **GetAudioSourceConfigurationOptionsResponse** message.
9. Repeat steps 5-8 for each audio source configuration from the **GetAudioSourceConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetAudioSourceConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationOptionsResponse** message.
- The DUT sent InputTokensAvailable list in the **GetAudioSourceConfigurationOptionsResponse** message that contains not unique tokens.
- The DUT did not send current SourceToken for the audio source configuration from the GetAudioSourceConfigurationsResponse message in InputTokensAvailable list in the **GetAudioSourceConfigurationOptionsResponse** message.

## 5.3.12.4 PROFILES AND AUDIO SOURCE CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-2-4

**Specification Coverage:** Get media profiles, Get audio source configuration options,

**Feature Under Test:** GetProfiles, GetAudioSourceConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfiles command and GetAudioSourceConfigurationOptions command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of existing profiles from device.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationOptionsRequest** (ProfileToken, ConfigurationToken) message to retrieve available audio source configuration options for profile and audio source configuration from device.
6. The DUT sends **GetAudioSourceConfigurationOptionsResponse** message.
7. Check that InputTokensAvailable list from the **GetAudioSourceConfigurationOptionsResponse** message contains only unique tokens.
8. Check that SourceToken for the audio source configuration from GetProfilesResponse message exists in InputTokensAvailable list from the **GetAudioSourceConfigurationOptionsResponse** message.
9. Repeat steps 5-8 for each audio source configuration from the **GetProfilesResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioSourceConfigurationOptionsResponse** message.

- The DUT did not send valid **GetAudioSourceConfigurationOptionsResponse** message.
- The DUT sent InputTokensAvailable list in the **GetAudioSourceConfigurationOptionsResponse** message that contains not unique tokens.
- The DUT did not send current SourceToken for the audio source configuration from **GetProfilesResponse** message in InputTokensAvailable list in the **GetAudioSourceConfigurationOptionsResponse** message.

### 5.3.12.5 AUDIO SOURCE CONFIGURATIONS AND AUDIO SOURCES CONSISTENCY

**Test Case ID:** MEDIA-3-2-5

**Specification Coverage:** Get audio source configurations, Get audio sources

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSources

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioSourceConfigurations command and GetAudioSources command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourcesRequest** message to retrieve list of Audio Sources from the DUT.
6. The DUT sends **GetAudioSourcesResponse** message.
7. Check that each AudioSource from **GetAudioSourcesResponse** message has unique token.

8. Check that every AudioSourceConfiguration.SourceToken from **GetAudioSourceConfigurationsResponse** message exists in **GetAudioSourcesResponse** message (AudioSource.token).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetAudioSourcesResponse** message.
- The DUT did not send valid **GetAudioSourcesResponse** message.
- The DUT return two or more AudioSources in **GetAudioSourcesResponse** message with the same VideoSourceToken.
- The DUT returned the **GetAudioSourceConfigurationsResponse** message with AudioSources that were not included in the **GetAudioSourceConfigurationsResponse** message.

## 5.3.12.6 AUDIO SOURCE CONFIGURATION USE COUNT (CURRENT STATE)

**Test Case ID:** MEDIA-3-2-6

**Specification Coverage:** Get media profiles, Get audio source configurations, Get audio source configuration

**Feature Under Test:** GetProfiles, GetAudioSourceConfigurations, GetAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. Profile with AudioSourceConfiguration exists. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their audio source configurations from device.
6. The DUT sends **GetProfilesResponse** message.
7. Check the **UseCount = usecount1** value for the first AudioSourceConfiguration (ASC1) in the list on step 3.
8. Check that there are not more than **usecount1** profiles with this AudioSourceConfiguration in the list from step 6.
9. Check that **UseCount** value in **GetProfilesResponse** for every occurrence of this AudioSourceConfiguration is **usecount1**.
10. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = the first AudioSourceConfiguration from list on step 3**) message to retrieve audio source configuration parameters.
11. The DUT sends **GetAudioSourceConfigurationResponse** message.
12. Check the UseCount value in **GetAudioSourceConfigurationResponse** (**UseCount = usecount1**).
13. Repeat steps 7-10 for all other AudioSourceConfigurations from the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- The DUT sent UseCount value which is less than amount of profiles with AudioSourceConfiguration.
- The DUT sent different UseCount values in **GetProfilesResponse**, **GetAudioSourceConfigurationsResponse** and **GetAudioSourceConfigurationResponse** messages.

### 5.3.12.7 AUDIO SOURCE CONFIGURATION USE COUNT (ADD SAME AUDIO SOURCE CONFIGURATION TO PROFILE TWICE)

**Test Case ID:** MEDIA-3-2-12

**Specification Coverage:** Get media profiles, Get media profile, Add audio source configuration to a profile, Get audio source configurations, Get audio source configuration

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSourceConfiguration, AddAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count after adding same audio source configuration to profile twice.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles Command

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.

5. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
8. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** and **RemoveAudioSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioEncoderConfiguration and AudioSourceConfiguration from **profile1**
9. The DUT sends **RemoveAudioEncoderConfigurationResponse** and **RemoveAudioSourceConfigurationResponse** message
10. Execute steps 8 - 17.
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source from the list on step 4 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
14. The DUT sends **AddAudioSourceConfigurationResponse** message.
15. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
16. The DUT sends **GetAudioSourceConfigurationResponse** message.
17. Check the UseCount value in **GetAudioSourceConfigurationResponse** message (**UseCount = usecount1+1**, usecount1 is value of UseCount for **ASC1** from the list on step 4). If test step 6 is executed, the usecount1 could be reduced by 1.
18. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = ASC1**) message to replace audio source configuration in profile.
19. The DUT sends **AddAudioSourceConfigurationResponse** message.
20. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
21. The DUT sends **GetAudioSourceConfigurationResponse** message.

22. Check that **UseCount = usecount1+1**, in **GetAudioSourceConfigurationResponse** message.
23. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with audio source configuration.
24. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the AudioSourceConfiguration to one more profile.
- UseCount value is changed after repeated adding of the AudioSourceConfiguration in profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.12.8 AUDIO SOURCE CONFIGURATION USE COUNT (ADD DIFFERENT AUDIO SOURCE CONFIGURATIONS IN PROFILE)

**Test Case ID:** MEDIA-3-2-13

**Specification Coverage:** Get media profiles, Get media profile, Add audio source configuration to a profile, Get audio source configurations, Get audio source configuration

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSourceConfiguration, AddAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count after adding different audio source configurations to profile.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. There are at least two AudioSourceConfigurations. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. If there is only one AudioSourceConfiguration in the **GetAudioSourceConfigurationsResponse** message go to the next test.
6. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
7. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
8. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
9. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** and **RemoveAudioSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioSourceConfiguration from **profile1**
10. The DUT sends **RemoveAudioEncoderConfigurationResponse** and **RemoveAudioSourceConfigurationResponse** message
11. Execute steps 9-21
12. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
13. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
14. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source from the list on step 4 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.

15. The DUT sends **AddAudioSourceConfigurationResponse** message.
16. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
17. The DUT sends **GetAudioSourceConfigurationResponse** message.
18. Check the UseCount value in **GetAudioSourceConfigurationResponse** (**UseCount = usecount1+1**, usecount1 is value of UseCount for **ASC1** from the list on step 4). If test step 7 is executed, the usecount1 value could be reduced by 1.
19. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = ASC2, where ASC2 is other ASC from list on step 4**) message to replace audio source configuration in **profile**.
20. The DUT sends **AddAudioSourceConfigurationResponse** message.
21. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
22. The DUT sends **GetAudioSourceConfigurationResponse** message.
23. Check that **UseCount = usecount1**, in **GetAudioSourceConfigurationResponse**.
24. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC2**) message to retrieve audio source configuration parameters.
25. The DUT sends **GetAudioSourceConfigurationResponse** message.
26. Check that **UseCount = usecount2+1**, in **GetAudioSourceConfigurationResponse** where usecount2 is UseCount value for ASC2 in the list on step 4.
27. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with audio source configuration.
28. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.

- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the AudioSourceConfiguration to one more profile.
- UseCount value is not decreased by 1 after replacing of the AudioSourceConfiguration in profile for removed AudioSourceConfiguration and UseCount value is not increased by 1 after replacing of the AudioSourceConfiguration in profile for new AudioSourceConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.12.9 AUDIO SOURCE CONFIGURATION USE COUNT (REMOVE AUDIO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-3-2-14

**Specification Coverage:** Remove audio source configuration from a profile, Get audio source configurations, Get audio source configuration

**Feature Under Test:** GetProfiles, GetAudioSourceConfigurations, GetAudioSourceConfiguration, RemoveAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count after removing audio source configuration from profile.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles are received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.

4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
8. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** and **RemoveAudioSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioSourceConfiguration from **profile1**
9. The DUT sends **RemoveAudioEncoderConfigurationResponse** and **RemoveAudioSourceConfigurationResponse** message
10. Execute steps 8-14
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source from the list on step 4 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
14. The DUT sends **AddAudioSourceConfigurationResponse** message.
15. ONVIF Client invokes **RemoveAudioSourceConfigurationRequest** (**ProfileToken = New Profile**) message to remove audio source configuration from profile1.
16. The DUT sends **RemoveAudioSourceConfigurationResponse** message.
17. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
18. The DUT sends **GetAudioSourceConfigurationResponse** message.
19. Check that **UseCount = usecount1**, in **GetAudioSourceConfigurationResponse** message where usecount1 is UseCount value for ASC1 in the list on step 4. If test step 6 is executed, the usecount1 value could be reduced by 1.
20. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with audio source configuration.
21. The DUT sends **DeleteProfileResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- The DUT did not send valid **RemoveAudioSourceConfigurationResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- UseCount value is not decreased by 1 after removing of the AudioSourceConfiguration from profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.12.10 AUDIO SOURCE CONFIGURATION USE COUNT (PROFILE DELETION WITH AUDIO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-3-2-15

**Specification Coverage:** Get media profiles, Get audio source configurations, Get audio source configuration, Delete media profile

**Feature Under Test:** GetProfiles, GetProfiles, GetAudioSourceConfigurations, GetAudioSourceConfiguration, SetAudioSourceConfiguration, DeleteProfile

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count after deletion of profile with audio source configuration in it.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of profiles is received by GetProfiles command/ List of profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
6. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
7. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
8. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** and **RemoveAudioSourceConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioSourceConfiguration from **profile1**
9. The DUT sends **RemoveAudioEncoderConfigurationResponse** and **RemoveAudioSourceConfigurationResponse** message
10. Execute steps 8-14
11. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
12. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
13. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source from the list on step 4 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
14. The DUT sends **AddAudioSourceConfigurationResponse** message.
15. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with audio source configuration.
16. The DUT sends **DeleteProfileResponse** message.
17. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.

18. The DUT sends **GetAudioSourceConfigurationResponse** message.
19. Check that **UseCount = usecount1**, in **GetAudioSourceConfigurationResponse** where usecount1 is UseCount value for ASC1 in the list on step 4. If test step 6 is executed, the usecount1 value could be reduced by 1.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **AddAudioSourceConfigurationResponse** message.
- The DUT did not send valid **AddAudioSourceConfigurationResponse** message.
- The DUT did not send **DeleteProfileResponse** message.
- The DUT did not send valid **DeleteProfileResponse** message.
- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- UseCount value is not decreased by 1 after deletion of the profile with the AudioSourceConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.12.11 AUDIO SOURCE CONFIGURATION USE COUNT (SET AUDIO SOURCE CONFIGURATION)

**Test Case ID:** MEDIA-3-2-16

**Specification Coverage:** Get audio source configurations, Get audio source configuration, Modify an audio source configuration.

**Feature Under Test:** GetAudioSourceConfigurations, GetAudioSourceConfiguration, SetAudioSourceConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Source Configuration use count after changing of audio source configuration parameters.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
4. The DUT sends **GetAudioSourceConfigurationsResponse** message.
5. ONVIF Client invokes **SetAudioSourceConfigurationRequest** (**token = the first Audio Source Configuration from the list on step 4(ASC1), UseCount = NewUseCount**) message to set parameters for audio source configuration.
6. The DUT sends **SetAudioSourceConfigurationResponse** message.
7. ONVIF Client invokes **GetAudioSourceConfigurationRequest** (**ConfigurationToken = ASC1**) message to retrieve audio source configuration parameters.
8. The DUT sends **GetAudioSourceConfigurationResponse** message.
9. Check that **UseCount = usecount1**, in **GetAudioSourceConfigurationResponse** where usecount1 is UseCount value for ASC1 in the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationsResponse** message.
- The DUT did not send **SetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **SetAudioSourceConfigurationResponse** message.

- The DUT did not send **GetAudioSourceConfigurationResponse** message.
- The DUT did not send valid **GetAudioSourceConfigurationResponse** message.
- UseCount value was changed after trying to set UseCount value.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.13 Audio Encoder Configuration

### 5.3.13.1 AUDIO ENCODER CONFIGURATIONS AND PROFILES CONSISTENCY

**Test Case ID:** MEDIA-3-3-1

**Specification Coverage:** Get media profiles, Get audio encoder configurations

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that Audio Encoder Configurations and Audio Encoder Profiles are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their audio encoder configurations.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
6. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
7. Check that each GetAudioEncoderConfiguration from **GetAudioEncoderConfigurationsResponse** message has unique token.

8. Check that each AudioEncoderConfiguration from the **GetProfilesResponse** message are included in the **GetAudioEncoderConfigurationsResponse** message.
9. Check that AudioEncoderConfiguration parameters are same in the **GetProfilesResponse** message and in the **GetAudioEncoderConfigurationsResponse** message for corresponding AudioEncoderConfiguration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT return two or more AudioEncoderConfigurations in **GetAudioEncoderConfigurationsResponse** message with the same ConfigurationToken.
- The DUT returned the **GetProfilesResponse** message with AudioEncoderConfigurations that were not included in the **GetAudioEncoderConfigurationsResponse** message.
- The DUT returned different parameters list and parameters values in the **GetAudioEncoderConfigurationsResponse** message and in the **GetProfilesResponse** message for the same AudioEncoderConfiguration.

## 5.3.13.2 AUDIO ENCODER CONFIGURATIONS AND AUDIO ENCODER CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-3-3-2

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration

**Feature Under Test:** GetAudioEncoderConfigurations, GetAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioEncoderConfigurations command and GetAudioEncoderConfiguration command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of Audio Source Configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (ConfigurationToken) message to retrieve parameters of audio encoder configuration from the DUT.
6. The DUT sends **GetAudioEncoderConfigurationResponse** message.
7. Verify that all parameters for audio encoder configuration from the GetAudioEncoderConfigurationsResponse message and from the **GetAudioEncoderConfigurationResponse** message are the same.
8. Repeat steps 5-7 for each Audio Encoder Configurations from the **GetAudioEncoderConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send equal parameters for one or more AudioEncoderConfiguration in the **GetAudioEncoderConfigurationResponse** message and in the **GetAudioEncoderConfigurationsResponse** message.

### 5.3.13.3 AUDIO ENCODER CONFIGURATIONS AND AUDIO ENCODER CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-3-3

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration options

**Feature Under Test:** GetAudioEncoderConfigurations, GetAudioEncoderConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioEncoderConfiguration command and GetAudioEncoderConfigurationOptions command are consistent.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of Audio Encoder Configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationOptionsRequest** (ConfigurationToken) message to retrieve available audio encoder configuration options from device.
6. The DUT sends **GetAudioEncoderConfigurationOptionsResponse** message.
7. Check that Encoding value for audio encoder configuration exists in Encoding parameter of one of the Option sections from list in the **GetAudioEncoderConfigurationOptionsResponse** message, and Bitrate and SampleRate exist in BitrateList and SampleRateList for this encoding in the **GetAudioEncoderConfigurationOptionsResponse** message.
8. Repeat steps 5-7 for each audio encoder configuration from the GetAudioEncoderConfigurationsResponse message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT sent non-available Encoding value for one or more AudioSourceConfigurations
- The DUT sent not available combination of Encoding and Bitrate or Encoding and SampleRate for one or more AudioEncoderConfigurations.

### 5.3.13.4 PROFILES AND AUDIO ENCODER CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-3-4

**Specification Coverage:** Get media profile, Get audio encoder configuration options

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfiles command and GetAudioEncoderConfigurationOptions command are consistency.

**Pre-Requisite:** Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles from device.

4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetAudioEncoderConfigurationOptionsRequest** (ProfileToken, ConfigurationToken) message to retrieve AudioEncoderConfigurationOptions for audio encoder configuration and profile from the DUT.
6. The DUT sends **GetAudioEncoderConfigurationOptionsResponse** message.
7. Check that Encoding value for audio encoder configuration exists in Encoding parameter of one of the Option sections from list in the **GetAudioEncoderConfigurationOptionsResponse** message, and Bitrate and SampleRate exist in BitrateList and SampleRateList for this encoding in the **GetAudioEncoderConfigurationOptionsResponse** message.
8. Repeat steps 5-7 for each audio encoder configuration from the GetProfilesResponse message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT sent non-available Encoding value for one or more AudioSourceConfigurations
- The DUT sent not available combination of Encoding and Bitrate or Encoding and SampleRate for one or more AudioEncoderConfigurations.

## 5.3.13.5 AUDIO ENCODER CONFIGURATION USE COUNT (CURRENT STATE)

**Test Case ID:** MEDIA-3-3-5

**Specification Coverage:** Get media profiles, Get audio encoder configurations, Get audio encoder configuration

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurations, GetAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their audio encoder configurations from device.
6. The DUT sends **GetProfilesResponse** message.
7. Check the **UseCount = usecount1** value for the first AudioEncoderConfiguration (**AEC1**) in the list on step 3.
8. Check that there are not more than usecount1 profiles with this AudioEncoderConfiguration in the list from step 6.
9. Check that UseCount value in **GetProfilesResponse** message for every occurrence of this AudioEncoderConfiguration is usecount1.
10. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
11. The DUT sends **GetAudioEncoderConfigurationResponse** message.
12. Check the UseCount value in **GetAudioEncoderConfigurationResponse** (**UseCount = usecount1**) message.
13. Repeat steps 7-10 for all other AudioEncoderConfigurations from the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT sent UseCount value which is less than amount of profiles with AudioEncoderConfiguration.
- The DUT sent different UseCount values in **GetProfilesResponse**, **GetAudioEncoderConfigurationsResponse** and **GetAudioEncoderConfigurationResponse** messages.

### 5.3.13.6 AUDIO ENCODER CONFIGURATION OPTIONS AND AUDIO ENCODER CONFIGURATIONS CONSISTENCY (BITRATE AND SAMPLERATE)

**Test Case ID:** MEDIA-3-3-11

**Specification Coverage:** Get audio encoder configuration options

**Feature Under Test:** GetAudioEncoderConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To that verify bitrate and samplerate values (the output bitrate [kbps], the output sample rate [kHz]) in the GetAudioEncoderConfigurations command are consistent with options and real settings.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. Media Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client
2. Start the DUT
3. ONVIF Client invokes **GetAudioEncoderConfigurationOptionsRequest** message to retrieve list of Audio Encoder Configuration Options from device.
4. Verify the **GetAudioEncoderConfigurationOptionsResponse** message from the DUT. Check that all values of Options.BitrateList.Items are less than 10000. Check that all values of Options.SampleRateList.Items are less than 1000.
5. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of Audio Encoder Configurations from device.
6. Verify the **GetAudioEncoderConfigurationsResponse** message from the DUT.
7. ONVIF Client invokes **GetAudioEncoderConfigurationOptionsRequest** message (**ConfigurationToken = 'token1' where "Token1" is a first audio encoder configuration token from GetAudioEncoderConfigurationsResponse message**) to retrieve list of Audio Encoder Configuration Options for specified Audio Encoder Configuration from device.
8. Verify the **GetAudioEncoderConfigurationOptionsResponse** message from the DUT. Check that all values of Options.BitrateList.Items are less than 10000 and listed in Options.BitrateList of **GetAudioEncoderConfigurationOptionsResponse** message from step 4. Check that all values of Options.SampleRateList.Items are less than 1000 listed in Options.SampleRateList of **GetAudioEncoderConfigurationOptionsResponse** message from step 4.
9. Repeat steps 7-8 for the rest Audio Encoder Configurations supported by the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT sent at least one Options.BitrateList.Item that is greater than 10000.
- The DUT sent at least one Options.SampleRateList.Item that is greater than 1000.

- The DUT sent at least one Options.BitrateList.Item in **GetAudioEncoderConfigurationOptionsResponse** message for specific Audio Encoder Configuration that is not listed in general **GetAudioEncoderConfigurationOptionsResponse** message.
- The DUT sent at least one Options.SampleRateList.Item in **GetAudioEncoderConfigurationOptionsResponse** message for specific Audio Encoder Configuration that is not listed in general **GetAudioEncoderConfigurationOptionsResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

**NOTE:** Bitrate shall be specified in kbps, so it shall not exceed 10000.

**NOTE:** SampleRate shall be specified in kHz, so it shall not exceed 1000.

## 5.3.13.7 AUDIO ENCODER CONFIGURATION USE COUNT (ADD SAME AUDIO ENCODER CONFIGURATION TO PROFILE TWICE)

**Test Case ID:** MEDIA-3-3-12

**Specification Coverage:** Get media profiles, Get media profile, Add audio encoder configuration to a profile, Get audio encoder configurations, Get audio encoder configuration

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurations, GetAudioEncoderConfiguration, AddAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count after adding same audio encoder configuration to profile twice.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.

3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
6. The DUT sends **GetAudioSourceConfigurationsResponse** message.
7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioEncoderConfiguration from **profile1**
11. The DUT sends **RemoveAudioEncoderConfigurationResponse** message
12. Execute steps 10-26
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleAudioSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioSourceConfigurations for the new profile.
16. The DUT sends **GetCompatibleAudioSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source configuration from the list on step 11 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
18. The DUT sends **AddAudioSourceConfigurationResponse** message.
19. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioEncoderConfigurations for the new profile with Audio Source Configuration.
20. The DUT sends **GetCompatibleAudioEncoderConfigurationsResponse** message. If response does not contain any AudioEncoderConfiguration, repeat steps 12-15 with next

audio source configuration from step 11. If there are no other audio source configuration, skip other steps and go to the next test.

21. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ConfigurationToken = first audio encoder from the list on step 15 (AEC1), ProfileToken = New Profile**) message to add AudioEncoderConfiguration to the new profile.
22. The DUT sends **AddAudioEncoderConfigurationResponse** message.
23. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
24. The DUT sends **GetAudioEncoderConfigurationResponse** message.
25. Check the UseCount value in **GetAudioEncoderConfigurationResponse** message (**UseCount = usecount1+1**, usecount1 is value of UseCount for AEC1 from the list on step 4). If test step 8 is executed, the usecount1 value could be reduced by 1.
26. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = AEC1**) message to replace audio encoder configuration in profile.
27. The DUT sends **AddAudioEncoderConfigurationResponse** message.
28. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
29. The DUT sends **GetAudioEncoderConfigurationResponse** message.
30. Check that **UseCount = usecount1+1**, in **GetAudioEncoderConfigurationResponse** message.
31. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleAudioEncoderConfigurationsResponse** message.

- The DUT did not send valid **GetCompatibleAudioSourceConfigurationsResponse** message.
- The DUT did not send valid **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- UseCount value is not increased by 1 after adding of the AudioEncoderConfiguration to one more profile.
- UseCount value is changed after repeated adding of the AudioEncoderConfiguration in profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.13.8 AUDIO ENCODER CONFIGURATION USE COUNT (ADD DIFFERENT AUDIO ENCODER CONFIGURATIONS IN PROFILE)

**Test Case ID:** MEDIA-3-3-13

**Specification Coverage:** Get media profiles, Get media profile, Add audio encoder configuration to a profile, Get audio encoder configurations, Get audio encoder configuration

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurations, GetAudioEncoderConfiguration, AddAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count after adding different audio encoder configurations to profile.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. There are at least two AudioEncoderConfigurations. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.

4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. If there is only one AudioEncoderConfiguration in the **GetAudioEncoderConfigurationsResponse** message go to the next test.
6. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
7. The DUT sends **GetAudioSourceConfigurationsResponse** message.
8. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
9. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
10. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
11. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioEncoderConfiguration from **profile1**
12. The DUT sends **RemoveAudioEncoderConfigurationResponse** message
13. Execute steps 11-30
14. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
15. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
16. ONVIF Client invokes **GetCompatibleAudioSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioSourceConfigurations for the new profile.
17. The DUT sends **GetCompatibleAudioSourceConfigurationsResponse** message.
18. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source configuration from the list on step 11 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
19. The DUT sends **AddAudioSourceConfigurationResponse** message.
20. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioEncoderConfigurations for the new profile with Audio Source Configuration.
21. The DUT sends **GetCompatibleAudioEncoderConfigurationsResponse** message. If response does not contain at least two AudioEncoderConfiguration, repeat steps 12-15 with

next audio source configuration from step 11. If there are no other audio source configuration, skip other steps and go to the next test.

22. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ConfigurationToken = first audio encoder from the list on step 15 (AEC1), ProfileToken = New Profile**) message to add AudioEncoderConfiguration to the new profile.
23. The DUT sends **AddAudioEncoderConfigurationResponse** message.
24. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
25. The DUT sends **GetAudioEncoderConfigurationResponse** message.
26. Check the UseCount value in **GetAudioEncoderConfigurationResponse** (**UseCount = usecount1+1**, usecount1 is value of UseCount for **AEC1** from the list on step 4). If test step 9 is executed, the usecount1 value could be reduced by 1.
27. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ProfileToken = New Profile, ConfigurationToken = AEC2, where AEC2 is other AEC from list on step 4**) message to replace audio encoder configuration in profile.
28. The DUT sends **AddAudioEncoderConfigurationResponse** message.
29. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
30. The DUT sends **GetAudioEncoderConfigurationResponse** message.
31. Check that **UseCount = usecount1**, in **GetAudioEncoderConfigurationResponse**.
32. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC2**) message to retrieve audio encoder configuration parameters.
33. The DUT sends **GetAudioEncoderConfigurationResponse** message.
34. Check that **UseCount = usecount2+1**, in **GetAudioEncoderConfigurationResponse** where usecount2 is UseCount value for AEC2 in the list on step 4.
35. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetCompatibleAudioSourceConfigurationsResponse** message.
- UseCount value is not increased by 1 after adding of the AudioEncoderConfiguration to one more profile.
- UseCount value is not decreased by 1 after replacing of the AudioEncoderConfiguration in profile for removed AudioEncoderConfiguration and
- UseCount value is not increased by 1 after replacing of the AudioEncoderConfiguration in profile for new AudioEncoderConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.13.9 AUDIO ENCODER CONFIGURATION USE COUNT (REMOVE AUDIO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-3-3-14

**Specification Coverage:** Get media profiles, Remove audio encoder configuration from a profile, Get audio encoder configurations, Get audio encoder configuration

**Feature Under Test:** GetProfiles, GetAudioEncoderConfigurations, GetAudioEncoderConfiguration, RemoveAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count after removing audio encoder configuration from profile.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
6. The DUT sends **GetAudioSourceConfigurationsResponse** message.
7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioEncoderConfiguration from **profile1**
11. The DUT sends **RemoveAudioEncoderConfigurationResponse** message
12. Execute steps 10-22
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleAudioSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioSourceConfigurations for the new profile.
16. The DUT sends **GetCompatibleAudioSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source configuration from the list on step 11 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
18. The DUT sends **AddAudioSourceConfigurationResponse** message.

19. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioEncoderConfigurations for the new profile with Audio Source Configuration.
20. The DUT sends **GetCompatibleAudioEncoderConfigurationsResponse** message. If response does not contain any AudioEncoderConfiguration, repeat steps 12-15 with next audio source configuration from step 11. If there are no other audio source configuration, skip other steps and go to the next test.
21. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ConfigurationToken = first audio encoder from the list on step 15 (AEC1), ProfileToken = New Profile**) message to add AudioEncoderConfiguration to the new profile.
22. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** (**ProfileToken = New Profile**) message to remove audio encoder configuration from profile1.
23. The DUT sends **RemoveAudioEncoderConfigurationResponse** message.
24. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
25. The DUT sends **GetAudioEncoderConfigurationResponse** message.
26. Check that **UseCount = usecount1**, in **GetAudioEncoderConfigurationResponse** where usecount1 is UseCount value for AEC1 in the list on step 4. If test step 8 is executed, the usecount1 value could be reduced by 1.
27. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **RemoveAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleAudioEncoderConfigurationsResponse** message.

- The DUT did not send valid **GetCompatibleAudioSourceConfigurationsResponse** message.
- UseCount value is not decreased by 1 after removing of the AudioEncoderConfiguration from profile.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.13.10 AUDIO ENCODER CONFIGURATION USE COUNT (DELETION PROFILE WITH AUDIO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-3-3-15

**Specification Coverage:** Get audio encoder configurations, Get audio encoder configuration, Delete media profile

**Feature Under Test:** GetAudioEncoderConfigurations, GetAudioEncoderConfiguration, DeleteProfile.

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count after deletion profile with audio encoder configuration in it.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command. List of media profiles is received by GetProfiles command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioSourceConfigurationsRequest** message to retrieve list of audio source configurations from device.
6. The DUT sends **GetAudioSourceConfigurationsResponse** message.

7. ONVIF Client invokes **CreateProfileRequest** (**Name = NewName, Token = New Profile**) message to create profile.
8. If the DUT will generate a SOAP 1.2 fault message (**Action/MaxNVTProfiles**),
9. If there is profile with “fixed” attribute value “false” in profiles list (**profile1**)
10. ONVIF Client invokes **RemoveAudioEncoderConfigurationRequest** (**ProfileToken = profile1**) message to remove AudioEncoderConfiguration from **profile1**
11. The DUT sends **RemoveAudioEncoderConfigurationResponse** message
12. Execute steps 10-22
13. If there is no profile with “fixed” attribute value “false” in profiles list, end test.
14. If the DUT sends **CreateProfileResponse** message, validate **CreateProfileResponse** message from the DUT.
15. ONVIF Client invokes **GetCompatibleAudioSourceConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioSourceConfigurations for the new profile.
16. The DUT sends **GetCompatibleAudioSourceConfigurationsResponse** message.
17. ONVIF Client invokes **AddAudioSourceConfigurationRequest** (**ConfigurationToken = first audio source configuration from the list on step 11 (ASC1), ProfileToken = New Profile**) message to add AudioSourceConfiguration to the new profile.
18. The DUT sends **AddAudioSourceConfigurationResponse** message.
19. ONVIF Client invokes **GetCompatibleAudioEncoderConfigurationsRequest** (**ProfileToken = New Profile**) message to get compatible AudioEncoderConfigurations for the new profile with Audio Source Configuration.
20. The DUT sends **GetCompatibleAudioEncoderConfigurationsResponse** message. If response does not contain any AudioEncoderConfiguration, repeat steps 12-15 with next audio source configuration from step 11. If there are no other audio source configuration, skip other steps and go to the next test.
21. ONVIF Client invokes **AddAudioEncoderConfigurationRequest** (**ConfigurationToken = first audio encoder from the list on step 15 (AEC1), ProfileToken = New Profile**) message to add AudioEncoderConfiguration to the new profile.
22. ONVIF Client invokes **DeleteProfileRequest** (**ProfileToken = New Profile**) message to remove profile with audio encoder configuration.

23. The DUT sends **DeleteProfileResponse** message.
24. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
25. The DUT sends **GetAudioEncoderConfigurationResponse** message.
26. Check that **UseCount = usecount1**, in **GetAudioEncoderConfigurationResponse** where usecount1 is UseCount value for AEC1 in the list on step 4.If test step 8 is executed, the usecount1 value could be reduced by 1.
27. ONVIF Client restores DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **AddAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **DeleteProfileResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetCompatibleAudioEncoderConfigurationsResponse** message
- The DUT did not send valid **GetCompatibleAudioSourceConfigurationsResponse** message.
- UseCount value is not decreased by 1 after deletion of the profile with the AudioEncoderConfiguration.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.13.11 AUDIO ENCODER CONFIGURATION USE COUNT (SET AUDIO ENCODER CONFIGURATION)

**Test Case ID:** MEDIA-3-3-16

**Specification Coverage:** Get media profiles, Get audio encoder configurations, Get audio encoder configuration, Modify audio encoder configurations.

**Feature Under Test:** GetAudioEncoderConfigurations, GetAudioEncoderConfiguration, SetAudioEncoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check Audio Encoder Configuration use count after setting audio encoder configuration parameters.

**Pre-Requisite:** Media is supported by DUT. Audio is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioEncoderConfigurationsRequest** message to retrieve list of audio encoder configurations from device.
4. The DUT sends **GetAudioEncoderConfigurationsResponse** message.
5. ONVIF Client invokes **SetAudioEncoderConfigurationRequest** (**token = the first Audio Encoder Configuration from the list on step 4(AEC1), UseCount = NewUseCount**) message to set parameters for audio encoder configuration.
6. The DUT sends **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetAudioEncoderConfigurationRequest** (**ConfigurationToken = AEC1**) message to retrieve audio encoder configuration parameters.
8. The DUT sends **GetAudioEncoderConfigurationResponse** message.
9. Check that **UseCount = usecount1**, in **GetAudioEncoderConfigurationResponse** where usecount1 is UseCount value for AEC1 in the list on step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationsResponse** message.

- The DUT did not send **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **GetAudioEncoderConfigurationResponse** message.
- The DUT did not send **SetAudioEncoderConfigurationResponse** message.
- The DUT did not send valid **SetAudioEncoderConfigurationResponse** message.
- UseCount value changed after trying to set UseCount value.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.3.14 Audio Output Configuration

### 5.3.14.1 SET AUDIO OUTPUT CONFIGURATION

**Test Case ID:** MEDIA-3-4-4

**Specification Coverage:** Get audio output configurations, Get audio output configuration, Get audio output configuration options, Modify audio output configuration

**Feature Under Test:** GetAudioOutputConfigurations, GetAudioOutputConfiguration, GetAudioOutputConfigurationOptions, SetAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check SetAudioOutputConfiguration behavior.

**Pre-Requisite:** Media is supported by DUT. Audio output is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioOutputConfigurationsRequest** message to retrieve list of audio output configurations from device.
4. The DUT sends **GetAudioOutputConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioOutputConfigurationOptionsRequest** (ConfigurationToken = AOC1, where AOC1 is the first configuration from the list on step 4) message to retrieve options for AOC1 from device.

6. The DUT sends **GetAudioOutputConfigurationOptionsResponse** message.
7. ONVIF Client invokes **SetAudioOutputConfigurationRequest** (Name = NewName, token = AOC1, OutputToken = OT1, where OT1 is one of OutputTokensAvailable from GetAudioOutputConfigurationResponse, SendPrimacy = SendPrimacyMode1, where SendPrimacyMode1 is one of modes from SendPrimacyOptions from Response on step 6, OutputLevel = OL1, where OL1 is between OutputRange.min and OutputRange.max from response on step 6, ForcePersistence = false) message to set parameters for audio output configuration.
8. The DUT sends **SetAudioOutputConfigurationResponse** message.
9. Validate **SetAudioOutputConfigurationResponse**
10. ONVIF Client invokes **GetAudioOutputConfigurationRequest** (ConfigurationToken = AOC1) message to retrieve settings of AOC1 from device.
11. The DUT sends **GetAudioOutputConfigurationResponse** message.
12. Validate AOC1 settings. Check, that all settings are as were set on step 7.
13. Repeat steps 5-12 for other Audio Output Configurations from step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioOutputConfigurationsResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationsResponse** message.
- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationOptionsResponse** message.
- The DUT did not send **GetAudioOutputConfigurationResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationResponse** message.
- The DUT did not send **SetAudioOutputConfigurationResponse** message.
- The DUT did not send valid **SetAudioOutputConfigurationResponse** message.
- At least for one Audio Output Configuration after modifying settings, there is at least one of following items:

- Name is not equal to NewName;
- OutputToken is not equal to OT1;
- SendPrimacy is not equal to SendPrimacyMode1;
- OutputLevel is not equal OL1;
- ForcePersistence is not false.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.3.14.2 SET AUDIO OUTPUT CONFIGURATION – INVALID OUTPUTTOKEN

**Test Case ID:** MEDIA-3-4-6

**Specification Coverage:** Modify audio output configuration

**Feature Under Test:** SetAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check SetAudioOutputConfiguration command behavior in case of incorrect configuration token.

**Pre-Requisite:** Media is supported by DUT. Audio output is supported by DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioOutputConfigurationsRequest** message to retrieve list of audio output configurations from device.
4. The DUT sends **GetAudioOutputConfigurationsResponse** message.
5. ONVIF Client invokes **GetAudioOutputConfigurationOptionsRequest** (ConfigurationToken = AOC1, where AOC1 is the first configuration from the list on step 4) message to retrieve options for AOC1 from device.

6. The DUT sends **GetAudioOutputConfigurationOptionsResponse** message.
7. ONVIF Client invokes **SetAudioOutputConfigurationRequest** (Name = NewName, token = AOC1, OutputToken = InvalidOT1, where InvalidOT1 is not one of OutputTokensAvailable from GetAudioOutputConfigurationResponse, SendPrimacy = InvalidSendPrimacyMode1, where InvalidSendPrimacyMode1 is not one of modes from SendPrimacyOptions from Response on step 6, OutputLevel = InvalidOL1, where OL1 is not between OutputRange.min and OutputRange.max from response on step 6) message with incorrect parameters.
8. The DUT will generate a SOAP 1.2 fault message (**Sender/InvalidArgVal/ConfigModify**).
9. ONVIF Client invokes **GetAudioOutputConfigurationRequest** (ConfigurationToken = AOC1) message to retrieve settings of AOC1 from device.
10. The DUT sends **GetAudioOutputConfigurationResponse** message.
11. Validate AOC1 settings. Check, that all settings are as settings of AOC1 in the list on step 4.
12. Repeat steps 5-12 for other Audio Output Configurations from step 4.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioOutputConfigurationsResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationsResponse** message.
- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationOptionsResponse** message.
- The DUT did not send **GetAudioOutputConfigurationResponse** message.
- The DUT did not send valid **GetAudioOutputConfigurationResponse** message.
- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).
- Settings are changed after incorrect **SetAudioOutputConfigurationRequest**.

**NOTE:** Other faults than specified in the test are acceptable but specified is preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.3.14.3 SET AUDIO OUTPUT CONFIGURATION – INVALID CONFIGURATION

**Test Case ID:** MEDIA-3-4-7

**Specification Coverage:** 11.12.5 Modify audio output configuration

**Feature Under Test:** SetAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check SetAudioOutputConfiguration command behavior in case of incorrect configuration token.

**Pre-Requisite:** Media is supported by the DUT. Audio output is supported by the DUT. ONVIF Client gets the Media Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **SetAudioOutputConfigurationRequest** (**token = InvalidAudioOutput**) message with incorrect ConfigurationToken.
4. The DUT will generate a SOAP 1.2 fault message (**Sender/InvalidArgVal/NoConfig**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent an incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** Other faults than specified in the test are acceptable though the specified are preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

### 5.3.14.4 GET AUDIO OUTPUT CONFIGURATION OPTIONS

**Test Case ID:** MEDIA-3-4-8

**Specification Coverage:** Audio Output Configuration (ONVIF Media Service Specification)

**Feature Under Test:** GetAudioOutputConfigurations, GetAudioOutputConfiguration, GetAudioOutputConfigurationOptions, SetAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check GetAudioOutputConfigurationOptions behavior.

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client generates creation or selection of profile with name (in *profileName*) and with token (out *profileToken*) and retrieves media profile token to delete (out *profileToDelete*) or media profile to restore (out *profileToRestore*) by following the procedure mentioned in Annex A.6.
4. ONVIF Client invokes **GetCompatibleAudioOutputConfigurations** with parameters:
   - ProfileToken := *profileToken*
5. The DUT responds with **GetCompatibleAudioOutputConfigurationsResponse** with parameters:
   - Configurations =: *compatibleConfigurationsList*
6. If *compatibleConfigurationsList* is empty, FAIL the test and skip other steps.
7. Set the following:
   - *configurationToken* := *compatibleConfigurationsList*[0].token
8. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters
   - ConfigurationToken := *configurationToken*

- ProfileToken := *profileToken*

9. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
   - Options
10. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters
    - ConfigurationToken skipped
    - ProfileToken := *profileToken*
11. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
    - Options
12. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters
    - ConfigurationToken := *configurationToken*
    - ProfileToken skipped
13. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
    - Options
14. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters
    - ConfigurationToken skipped
    - ProfileToken skipped
15. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
    - Options
16. If *profileToRestore* was returned by HelperCreateEmptyProfile, ONVIF Client restore media profile (in *profileToRestore*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.
17. If *profileToDelete* was returned by HelperCreateEmptyProfile, ONVIF Cliend delete media profile (in *profileToDelete*) by following the procedure mentioned in Annex A.8 to restore DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetCompatibleAudioOutputConfigurationsResponse** message.
- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.

## 5.3.14.5 AUDIO OUTPUT CONFIGURATION

**Test Case ID:** MEDIA-3-4-9

**Specification Coverage:** Audio Output Configuration (ONVIF Media Service Specification), Add audio output configuration to a profile (ONVIF Media Specification), Remove audio output configuration from a profile (ONVIF Media Specification), Get audio outputs (ONVIF Media Specification), Get audio output configurations (ONVIF Media Specification), Get audio output configuration (ONVIF Media Specification), Get compatible audio output configurations (ONVIF Media Specification), Get audio output configuration options (ONVIF Media Specification), Modify an audio output configuration (ONVIF Media Specification).

**Feature Under Test:** GetAudioOutputs, GetAudioOutputConfigurations, GetAudioOutputConfiguration, GetCompatibleAudioOutputConfigurations, AddAudioOutputConfiguration, GetAudioOutputConfigurationOptions, SetAudioOutputConfiguration, RemoveAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify Audio Output Configuration Operations

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Set the following:
   - *profileName* := “TestProfile”

4. ONVIF Client generates creation or selection of profile with name (in *profileName*) and with token (out *profileToken*) and retrieves media profile token to delete (out *profileToDelete*) or media profile to restore (out *profileToRestore*) by following the procedure mentioned in Annex A.6.
5. ONVIF Client invokes **GetAudioOutputs** to retrieve the existing audio outputs of DUT.
6. The DUT responds with **GetAudioOutputsResponse** with parameters:
   - AudioOutputs := *audioOutputsList*
7. If *audioOutputsList* is empty, FAIL the test and go to step 32.
8. ONVIF Client invokes **GetCompatibleAudioOutputConfigurations** with parameters:
   - ProfileToken := *profileToken*
9. The DUT responds with **GetCompatibleAudioOutputConfigurationsResponse** with parameters:
   - Configurations =: *compatibleConfigurationsList*
10. If *compatibleConfigurationsList* is empty, FAIL the test and go to step 32.
11. Set the following:
    - *configurationToken* := *compatibleConfigurationsList*.token[0]
12. ONVIF Client invokes **AddAudioOutputConfiguration** with parameters:
    - ProfileToken := *profileToken*
    - ConfigurationToken := *configurationToken*
13. The DUT responds with **AddAudioOutputResponse** message.
14. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters:
    - ProfileToken skipped
    - ConfigurationToken := *configurationToken*
15. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** with parameters:
    - Options =: *options*
16. Set the following:

- *invalidOutputLevel* := *options*.OutputLevelRange.Min – 1.

17. ONVIF Client invokes **SetAudioOutputConfiguration** with parameters:
    - Configuration.Name := *compatibleConfigurationsList*[0].token.Name
    - Configuration.UseCount := *compatibleConfigurationsList*[0].token.UseCount
    - Configuration.Token := *configurationToken*
    - Configuration.OutputToken := *compatibleConfigurationsList*[0].token.OutputToken
    - Configuration.SendPrimacy := *compatibleConfigurationsList*[0].token.SendPrimacy
    - Configuration.OutputLevel := *invalidOutputLevel*
    - ForcePersistence := false
18. The DUT responds with **env:Sender\ter: InvalidArgVal \ter:ConfigModify** SOAP 1.2 fault.
19. Set the following:
    - *validOutputLevel* := *options*.OutputLevelRange.Max.
20. ONVIF Client invokes **SetAudioOutputConfiguration** with parameters:
    - Configuration.Name := *compatibleConfigurationsList*[0].token.Name
    - Configuration.UseCount := *compatibleConfigurationsList*[0].token.UseCount
    - Configuration.Token := *configurationToken*
    - Configuration.OutputToken := *compatibleConfigurationsList*[0].token.OutputToken
    - Configuration.SendPrimacy := *compatibleConfigurationsList*[0].token.SendPrimacy
    - Configuration.OutputLevel := *validOutputLevel*
    - ForcePersistence := false
21. The DUT responds with **SetAudioOutputConfigurationResponse** message.
22. ONVIF Client invokes **GetAudioOutputConfiguration** with parameters:
    - ConfigurationToken := *configurationToken*
23. DUT responds with **GetAudioOutputConfigurationResponse** with parameters:
    - Configuration =: *updatedConfiguration*

24. If *updatedConfiguration*.Name does not equal to *compatibleConfigurationsList*[0].token.Name, FAIL the test and go to step 32.
25. If *updatedConfiguration*.Token does not equal to *configurationToken*, FAIL the test and go to step 32.
26. If *updatedConfiguration*.OutputToken does not equal to *compatibleConfigurationsList*[0].token.OutputToken, FAIL the test and go to step 32.
27. If *updatedConfiguration*.SendPrimacy does not equal to *compatibleConfigurationsList*[0].token. SendPrimacy, FAIL the test and go to step 32.
28. If *updatedConfiguration*.OutputLevel does not equal to *validOutputLevel*, FAIL the test and go to step 32.
29. ONVIF Client invokes **RemoveAudioOutputConfiguration** with parameters:
    - ProfileToken := *profileToken*
30. The DUT responds with **RemoveAudioOutputConfigurationResponse** message.
31. If *profileToRestore* was returned by HelperCreateEmptyProfile, ONVIF Client restore media profile (in *profileToRestore*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.
32. If *profileToDelete* was returned by HelperCreateEmptyProfile, ONVIF Client delete media profile (in *profileToDelete*) by following the procedure mentioned in Annex A.8 to restore DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioOutputsResponse** message.
- The DUT did not send **GetCompatibleAudioOutputConfigurationsResponse** message.
- The DUT did not send **AddAudioOutputConfigurationResponse** message.
- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.
- The DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetAudioOutputConfiguration** request.

- The DUT did not send **SetAudioOutputConfigurationResponse** message.
- The DUT did not send **GetAudioOutputConfigurationResponse** message.
- The DUT did not send **RemoveAudioOutputConfigurationResponse** message.

### 5.3.14.6 AUDIO DECODER CONFIGURATION

**Test Case ID:** MEDIA-3-4-10

**Specification Coverage:** Audio Decoder Configuration (ONVIF Media Service Specification), Get compatible audio decoder configurations (ONVIF Media Specification), Add audio decoder configuration to a profile (ONVIF Media Specification), GetProfile (ONVIF Media Specification), Remove audio decoder configuration from a profile (ONVIF Media Specification).

**Feature Under Test:** GetAudioDecoderConfigurations, GetAudioDecoderConfiguration, GetCompatibleAudioDecoderConfigurations, AddAudioDecoderConfiguration, GetProfile, RemoveAudioDecoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify Add Audio Decoder Configuration and Remove Audio Decoder Configuration Operations

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. Set the following:
   - *profileName* := “TestProfile”
4. ONVIF Client generates creation or selection of profile with name (in *profileName*) and with token (out *profileToken*) and retrieves media profile token to delete (out *profileToDelete*) or media profile to restore (out *profileToRestore*) by following the procedure mentioned in Annex A.6.
5. ONVIF Client invokes **GetCompatibleAudioOutputConfigurations** with parameters:

- ProfileToken := *profileToken*

6. The DUT responds with **GetCompatibleAudioOutputConfigurationsResponse** with parameters:
   - Configurations =: *compatibleAudioOutputConfigs*
7. If *compatibleAudioOutputConfigs* is empty, FAIL the test and and go to step 30.
8. Set the following:
   - *audioOutputConfigToken* := *compatibleAudioOutputConfigs*[0].token
9. ONVIF Client invokes **AddAudioOutputConfiguration** with parameters:
   - ProfileToken := *profileToken*
   - ConfigurationToken := *audioOutputConfigToken*
10. The DUT responds with **AddAudioOutputConfigurationResponse** message.
11. ONVIF Client invokes **GetAudioDecoderConfigurations**.
12. The DUT responds with **GetAudioDecoderConfigurationsResponse** with parameters:
    - Configurations =: *audioDecoderConfigsList*
13. If *audioDecoderConfigsList* is empty, FAIL the test and go to step 30.
14. ONVIF Client invokes **GetCompatibleAudioDecoderConfigurations** with parameters:
    - ProfileToken := *profileToken*
15. The DUT responds with **GetCompatibleAudioDecoderConfigurationsResponse** with parameters:
    - Configuration =: *compatibleAudioDecoderConfigs*
16. If *compatibleAudioDecoderConfigs* is empty, FAIL the test and go to step 30.
17. Set the following:
    - *audioDecoderConfigToken* := *compatibleAudioDecoderConfigs*[0].token
18. ONVIF Client invokes **AddAudioDecoderConfiguration** with parameters:
    - ProfileToken := *profileToken*
    - ConfigurationToken := *audioDecoderConfigToken*

19. The DUT responds with **AddAudioDecoderConfigurationResponse** message.
20. ONVIF Client invokes **GetProfile** with parameters:
    - ProfileToken := *profileToken*
21. The DUT responds with **GetProfileResponse** with parameters:
    - Profile =: *profile*
22. If *profile* does not contain Extension.AudioOutputConfiguration element, FAIL the test and go to step 30.
23. If *profile*.Extension.AudioOutputConfiguration.token value does not equal to *audioOutputConfigToken*, FAIL the test and go to step 30.
24. If *profile* does not contain Extension.AudioDecoderConfiguration element, FAIL the test and go to step 30.
25. If *profile*.Extension.AudioDecoderConfiguration.token value does not equal to *audioDecoderConfigToken*, FAIL the test and go to step 30.
26. ONVIF Client invokes **RemoveAudioDecoderConfiguration** with parameters:
    - ProfileToken := *profileToken*
27. The DUT responds with **RemoveAudioDecoderConfigurationResponse** message.
28. ONVIF Client invokes **RemoveAudioOutputConfiguration** with parameters:
    - ProfileToken := *profileToken*
29. The DUT responds with **RemoveAudioOutputConfigurationResponse** message.
30. If *profileToRestore* was returned by HelperCreateEmptyProfile, ONVIF Cliend restore media profile (in *profileToRestore*) by following the procedure mentioned in Annex A.7 to restore DUT configuration.
31. If *profileToDelete* was returned by HelperCreateEmptyProfile, ONVIF Cliend delete media profile (in *profileToDelete*) by following the procedure mentioned in Annex A.8 to restore DUT configuration.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetCompatibleAudioOutputConfigurationsResponse** message.
- The DUT did not send **AddAudioOutputConfigurationResponse** message.
- The DUT did not send **GetAudioDecoderConfigurationsResponse** message.
- The DUT did not send **GetCompatibleAudioDecoderConfigurationsResponse** message.
- The DUT did not send **AddAudioDecoderConfiguration** message.
- The DUT did not send **GetProfile** message.
- The DUT did not send **RemoveAudioDecoderConfigurationResponse** message.
- The DUT did not send **RemoveAudioOutputConfigurationResponse** message.

### 5.3.14.7 AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUT CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-4-11

**Specification Coverage:** Get Audio Output Configurations (ONVIF Media Specification), Get Audio Output Configuration Options (ONVIF Media Specification).

**Feature Under Test:** GetAudioOutputConfigurations, GetAudioOutputConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioOutputConfigurations command and GetAudioOutputConfigurationOptions command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioOutputConfigurations**.
4. The DUT responds with **GetAudioOutputConfigurationsResponse** with parameters:

- Configurations list =: *audioOutputConfigsList*

5. If *audioOutputConfigsList* is empty, FAIL the test and skip other steps.
6. For each audio output configuration (*audioOutputConfig*) from *audioOutputConfigsList* repeat the following steps :

6.1. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters

- ConfigurationToken := *audioOutputConfig*.token
- ProfileToken skipped

6.2. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** with parameters:

- Options =: *audioOutputOptions*

6.3. If *audioOutputOptions*.OutputTokensAvailable list contains at least two equal tokens, FAIL the test and skip other steps.

6.4. If *audioOutputOptions*.OutputTokensAvailable list does not contain *audioOutputConfig*.OutputToken value, FAIL the test and skip other steps.

6.5. If *audioOutputConfig* contains SendPrimacy element, check the following:

6.5.1. If *audioOutputOptions* does not contain at least one SendPrimacyOptions element, FAIL the test and skip other steps.

6.5.2. If *audioOutputOptions*.SendPrimacyOptions list does not contain *audioOutputConfig*.SendPrimacy value, FAIL the test and skip other steps.

6.6. If *audioOutputConfig*.OutputLevel value is less than *audioOutputOptions*.OutputLevelRange.Min, FAIL the test and skip other steps.

6.7. If *audioOutputConfig*.OutputLevel value is more than *audioOutputOptions*.OutputLevelRange.Max, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioOutputConfigurationsResponse** message.

- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.

### 5.3.14.8 PROFILES AND AUDIO OUTPUT CONFIGURATION OPTIONS CONSISTENCY

**Test Case ID:** MEDIA-3-4-12

**Specification Coverage:** Get Profiles (ONVIF Media Specification), Get Audio Output Configuration Options (ONVIF Media Specification).

**Feature Under Test:** GetProfiles, GetAudioOutputConfigurationOptions

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetProfiles command and GetAudioOutputConfigurationOptions command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles**.
4. DUT responds with **GetProfilesResponse** with parameters:
   - Profiles =: *profilesList*
5. If *profilesList* is empty, FAIL the test and skip other steps.
6. For each profile item with Extension.AudioOutputConfiguration element (*profile*) from *profilesList* repeat the following steps:
   6.1. ONVIF Client invokes **GetAudioOutputConfigurationOptions** with parameters
      - ConfigurationToken := *profile*.Extension.AudioOutputConfiguration.token
      - ProfileToken := *profile*.token
   6.2. The DUT responds with **GetAudioOutputConfigurationOptionsResponse** with parameters:

- Options =: *audioOutputOptions*

6.3. If *audioOutputOptions*.OutputTokensAvailable list contains at least two equal tokens, FAIL the test and skip other steps.

6.4. If *audioOutputOptions*.OutputTokensAvailable token list does not contain *profile*.Extension.AudioOutputConfiguration.OutputToken value, FAIL the test and skip other steps.

6.5. If *profile*.Extension.AudioOutputConfiguration contains SendPrimacy element, check the following:

6.5.1. If *audioOutputOptions* does not contain at least one SendPrimacyOptions element, FAIL the test and skip other steps.

6.5.2. If *audioOutputOptions*.SendPrimacyOptions list does not contain *profile*.Extension.AudioOutputConfiguration.SendPrimacy value, FAIL the test and skip other steps.

6.6. If *profile*.Extension.AudioOutputConfiguration.OutputLevel value is less than *audioOutputOptions*.OutputLevelRange.Min, FAIL the test and skip other steps.

6.7. If *profile*.Extension.AudioOutputConfiguration.OutputLevel value is more than *audioOutputOptions*.OutputLevelRange.Max, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.

## 5.3.14.9 PROFILES AND AUDIO OUTPUT CONFIGURATIONS CONSISTENCY

**Test Case ID:** MEDIA-3-4-13

**Specification Coverage:** Get media profiles (ONVIF Media Specification), Get audio output configurations (ONVIF Media Specification).

**Feature Under Test:** GetProfiles, GetAudioOutputConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioOutputConfigurations command and GetProfiles command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles**.
4. DUT responds with **GetProfilesResponse** with parameters:
   - Profiles =: *profilesList*
5. If *profilesList* is empty, FAIL the test and skip other steps.
6. ONVIF Client invokes **GetAudioOutputConfigurations**.
7. DUT responds with **GetAudioOutputConfigurationsResponse** with parameters:
   - Configurations list =: *audioOutputConfigsList*
8. If *audioOutputConfigsList* is empty, FAIL the test and skip other steps.
9. If *audioOutputConfigsList* contains at least two AudioOutputConfiguration items with equal token, FAIL the test and skip other steps.
10. For each profile item with Extension.AudioOutputConfiguration element (*profile*) from *profilesList* repeat the following steps:
    10.1. If *audioOutputConfigsList* does not contain AudioOutputConfiguration item with token value equals to *profile*.Extension.AudioOutputConfiguration.token, FAIL the test and skip other steps.
    10.2. If *profile*.Extension.AudioOutputConfiguration item does not have equal field values to *audioOutputConfigsList* [token = *profile*.Extension.AudioOutputConfiguration.token] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetAudioOutputConfigurationsResponse** message.

**NOTE:** The following fields are compared at step 2:

- AudioOutputConfiguration:
  - Name
  - UseCount
  - token
  - OutputToken
  - SendPrimacy
  - OutputLevel

## 5.3.14.10 AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUT CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-3-4-14

**Specification Coverage:** Get Audio Output Configurations (ONVIF Media Specification), Get Audio Output Configuration (ONVIF Media Specification).

**Feature Under Test:** GetAudioOutputConfigurations, GetAudioOutputConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioOutputConfigurations command and GetAudioOutputConfiguration command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioOutputConfigurations**.
4. DUT responds with **GetAudioOutputConfigurationsResponse** with parameters:
   - Configurations =: *audioOutputConfigsList*
5. If *audioOutputConfigsList* is empty, FAIL the test and skip other steps.
6. For each AudioOutputConfiguration item (*audioOutputConfig*) from *audioOutputConfigsList* repeat the following steps :
   6.1. ONVIF Client invokes **GetAudioOutputConfiguration** with parameters
      - ConfigurationToken := *audioOutputConfig*.token
   6.2. DUT responds with **GetAudioOutputConfigurationResponse** with parameters:
      - Configuration =: *configuration*
   6.3. If *audioOutputConfig* item does not have equal field values to configuration item, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioOutputConfigurationsResponse** message.
- DUT did not send **GetAudioOutputConfigurationResponse** message.

**NOTE:** The following fields are compared at step 3:

- AudioOutputConfiguration:
  - Name
  - UseCount
  - token

- OutputToken
- SendPrimacy
- OutputLevel

### 5.3.14.11 AUDIO OUTPUT CONFIGURATIONS AND AUDIO OUTPUTS CONSISTENCY

**Test Case ID:** MEDIA-3-4-15

**Specification Coverage:** Get Audio Output Configurations (ONVIF Media Specification), Get Audio Outputs (ONVIF Media Specification).

**Feature Under Test:** GetAudioOutputConfigurations, GetAudioOutputs

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioOutputConfigurations command and GetAudioOutputs command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioOutputConfigurations**.
4. DUT responds with **GetAudioOutputConfigurationsResponse** with parameters:
   - Configurations =: *audioOutputConfigsList*
5. If *audioOutputConfigsList* is empty, FAIL the test and skip other steps.
6. ONVIF Client invokes **GetAudioOutputs**.
7. DUT responds with **GetAudioOutputsResponse** with parameters:
   - AudioOutputs list =: *audioOutputsList*

8. If *audioOutputsList* is empty, FAIL the test and skip other steps.
9. If *audioOutputsList* contains at least two items with equal tokens, FAIL the test and skip other steps.
10. For each AudioOutputConfiguration item (*audioOutputConfig*) from *audioOutputConfigsList* repeat the following:
    10.1. If *audioOutputsList* does not contain token equals to *audioOutputConfig*.OutputToken, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioOutputConfigurationsResponse** message.
- DUT did not send **GetAudioOutputsResponse** message.

## 5.3.14.12 PROFILES AND AUDIO DECODER CONFIGURATIONS CONSISTENCY

**Test Case ID:** MEDIA-3-4-16

**Specification Coverage:** Get media profiles (ONVIF Media Specification), Get audio decoder configurations (ONVIF Media Specification).

**Feature Under Test:** GetProfiles, GetAudioDecoderConfigurations

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioDecoderConfigurations command and GetProfiles command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client invokes **GetProfiles**.
4. DUT responds with **GetProfilesResponse** with parameters:
   - Profiles =: *profilesList*
5. If *profilesList* is empty, FAIL the test and skip other steps.
6. ONVIF Client invokes **GetAudioDecoderConfigurations**.
7. DUT responds with **GetAudioDecoderConfigurationsResponse** with parameters:
   - Configurations list =: *audioDecoderConfigsList*
8. If *audioDecoderConfigsList* is empty, FAIL the test and skip other steps.
9. If *audioDecoderConfigsList* contains at least two AudioDecoderConfiguration items with equal token, FAIL the test and skip other steps.
10. For each profile item with Extension.AudioDecoderConfiguration element (*profile*) from *profilesList* repeat the following steps:
    10.1. If *audioDecoderConfigsList* does not contain AudioDecoderConfiguration item with token value equals to *profile*.Extension.AudioDecoderConfiguration.token, FAIL the test and skip other steps.
    10.2. If *profile*.Extension.AudioDecoderConfiguration item does not have equal field values to *audioDecoderConfigsList* [token = *profile*.Extension.AudioDecoderConfiguration.token] item, FAIL the test and skip other steps.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetAudioDecoderConfigurationsResponse** message.

**NOTE:** The following fields are compared at step 2:

- AudioDecoderConfiguration:

- Name
- UseCount
- token

### 5.3.14.13 AUDIO DECODER CONFIGURATIONS AND AUDIO DECODER CONFIGURATION CONSISTENCY

**Test Case ID:** MEDIA-3-4-17

**Specification Coverage:** Get Audio Decoder Configurations (ONVIF Media Specification), Get Audio Decoder Configuration (ONVIF Media Specification).

**Feature Under Test:** GetAudioDecoderConfigurations, GetAudioDecoderConfiguration

**WSDL Reference:** media.wsdl

**Test Purpose:** To check that GetAudioDecoderConfigurations command and GetAudioDecoderConfiguration command are consistent

**Pre-Requisite:** Media service is supported by the DUT. Audio output is supported by the DUT. Media Service is received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetAudioDecoderConfigurations**.
4. The DUT responds with **GetAudioDecoderConfigurationsResponse** with parameters:
   - Configurations list =: *audioDecoderConfigsList*
5. If *audioDecoderConfigsList* is empty, FAIL the test and skip other steps.
6. For each AudioDecoderConfiguration item (*audioDecoderConfig*) from *audioDecoderConfigsList* repeat the following steps :
   6.1. ONVIF Client invokes **GetAudioDecoderConfiguration** with parameters
      - ConfigurationToken := *audioDecoderConfig*.token

6.2. The DUT responds with **GetAudioDecoderConfigurationResponse** with parameters:

- Configuration =: *configuration*

6.3. If *audioDecoderConfig* item does not have equal field values to *configuration* item, FAIL the test.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetAudioDecoderConfigurationsResponse** message.
- The DUT did not send **GetAudioDecoderConfigurationResponse** message.

**NOTE:** The following fields are compared at step 3:

- AudioDecoderConfiguration:
  - Name
  - UseCount
  - token

## 5.3.15 PTZ Configuration

## 5.3.16 PTZ CONFIGURATIONS AND PROFILES CONSISTENCY

**Test Case ID:** MEDIA-4-1-2

**Specification Coverage:** Get media profiles, GetConfigurations.

**Feature Under Test:** GetProfiles, GetConfigurations

**WSDL Reference:** media.wsdl, ptz.wsdl.

**Test Purpose:** To check that GetPTZConfigurations command and GetProfiles command are consistent.

**Pre-Requisite:** PTZ is supported by DUT. ONVIF Client gets the Media Service entry point and PTZ Service entry point by GetCapabilities command.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfilesRequest** message to retrieve list of profiles and their PTZ configurations.
4. The DUT sends **GetProfilesResponse** message.
5. ONVIF Client invokes **GetConfigurations** message to retrieve list of PTZ Configurations from device.
6. The DUT sends **GetConfigurationsResponse** message.
7. Check that each PTZConfiguration from **GetConfigurationsResponse** message has unique token.
8. Check that all PTZConfigurations from the **GetProfilesResponse** message are included in list the **GetConfigurationsResponse** message.
9. Check that PTZConfiguration parameters are same in the **GetProfilesResponse** message and in the **GetConfigurationsResponse** message.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **GetProfilesResponse** message.
- The DUT did not send valid **GetProfilesResponse** message.
- The DUT did not send **GetConfigurationsResponse** message.
- The DUT did not send valid **GetConfigurationsResponse** message.
- The DUT return two or more PTZConfigurations in **GetConfigurationsResponse** message with the same ConfigurationToken.

- The DUT returned the **GetProfilesResponse** message with PTZConfigurations that were not included in the **GetConfigurationsResponse** message.
- The DUT returned different parameters list and parameters values for the same PTZConfiguration in the **GetConfigurationsResponse** message and in the **GetProfilesResponse** message.

## 5.4 Metadata Configuration

### 5.4.1 METADATA CONFIGURATION

**Test Case ID:** MEDIA-5-1-3

**Specification Coverage:** Create media profile, Add metadata configuration to a profile, Remove metadata configuration from a profile, Delete media profile, Get metadata configurations, Get metadata configuration, Get compatible metadata configurations, Get metadata configuration options, Modify a metadata configuration.

**Feature Under Test:** None

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify DUT Metadata Configuration Operations

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreateProfile** request with ProfileToken = ‘testprofileX’.
4. Verify **CreateProfileResponse** message or SOAP 1.2 fault message (**Action/ MaxNVTProfiles**) from the DUT. If fault was received, execute Annex A.3.
5. ONVIF Client will invoke **GetMetadataConfigurations** request to retrieve the list of metadata configurations supported by the DUT.
6. ONVIF Client verifies the list of metadata configurations sent by the DUT.
7. ONVIF Client invokes **GetCompatibleMetadataConfigurations** request with ‘testprofileX’ as ProfileToken.

8. The DUT sends the list of metadata configurations compatible with the received media profile token.
9. ONVIF Client invokes **AddMetadataConfiguration** request message with ProfileToken as 'testprofileX' and ConfigurationToken as one of the metadata configuration tokens received in the **GetCompatibleMetadataConfigurationsResponse**.
10. The DUT adds the Metadata configuration to the profile and sends the response.
11. ONVIF Client invokes **GetMetadataConfigurationOptions** request with ConfigurationToken as same token in **AddMetadataConfiguration** request.
12. The DUT sends the configurable options supported for the received metadata configuration.
13. ONVIF Client invokes **SetMetadataConfiguration** request with metadata configuration values outside the range defined in **GetMetadataConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
14. The DUT sends the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**).
15. ONVIF Client verifies the SOAP fault message sent by DUT.
16. ONVIF Client invokes **SetMetadataConfiguration** request with metadata configuration values within the range defined in **GetMetadataConfigurationOptionsResponse** and 'ForcePersistence' flag as 'FALSE'.
17. The DUT modifies the metadata configuration and sends the **SetMetadataConfiguration** indicating success.
18. ONVIF Client verifies the modified Metadata configuration by invoking the **GetMetadataConfigurations** request.
19. The DUT sends the modified Metadata configuration in **GetMetadataConfigurationResponse**.
20. ONVIF Client invokes **RemoveMetadataConfiguration** request with ProfileToken as 'testprofileX'.
21. The DUT removes the Metadata configuration token from media profile and sends the response.
22. If profile with profile token = testprofileX was created during test execution, then ONVIF Client invokes DeleteProfile request with ProfileToken as 'testprofileX'. Otherwise, ONVIF client skip rest steps and restore profile settings.
23. The DUT deletes the media profile and sends the response.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send **CreateProfileResponse** message.
- The DUT did not send **GetMetadataConfigurationsResponse** message.
- The DUT did not send **GetCompatibleMetadataConfigurationsResponse** message.
- The DUT did not send **AddMetadataConfigurationResponse** message.
- The DUT did not send **GetMetadataConfigurationOptionsResponse** message.
- The DUT did not send the SOAP 1.2 fault message (**InvalidArgVal/ConfigModify**) for invalid **SetMetadataConfiguration** request.
- The DUT did not send **SetMetadataConfigurationResponse** message.
- The DUT did not send **GetMetadataConfigurationResponse** message.
- The DUT did not modify metadata configuration correctly.
- The DUT did not send **RemoveMetadataConfigurationResponse** message.
- The DUT did not send **DeleteProfileResponse** message.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

**NOTE:** If profile was deleted during Annex A.3 execution, ONVIF Client restores the deleted profile and profile settings.

## 5.5 Media Streaming

### 5.5.1 SNAPSHOT URI

**Test Case ID:** MEDIA-6-1-1

**Specification Coverage:** Get media profiles, Request snapshot URI.

**Feature Under Test:** GetSnapshotUri

**WSDL Reference:** media.wsdl

**Test Purpose:** To retrieve snapshot URI of DUT for given media profile

**Pre-Requisite:** None

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** request to retrieve the list of existing media profiles on DUT.
4. ONVIF Client validates the **GetProfilesResponse** message sent by the DUT. At least one media profile should be present with video source and video encoder.
5. ONVIF Client invokes **GetSnapshotUri** request with ProfileToken as one of the media profile tokens received in **GetProfilesResponse** message.
6. DUT sends HTTP URI and parameters defining the lifetime of the URI like ValidUntilConnect, ValidUntilReboot and Timeout in the GetSnapshotUriResponse message.
7. ONVIF Client invokes HTTP GET request on the snapshot URI sent by DUT.
8. DUT sends 200 OK message and the single shot JPEG image data.
9. ONVIF Client verifies the JPEG image sent by the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetSnapshotUriResponse** message.
- DUT did not send one or more mandatory parameters in the **GetSnapshotUriResponse** message (mandatory parameters –Uri, ValidUntilConnect, ValidUntilReboot and Timeout).
- DUT did not send 200 OK message.

- DUT did not send valid JPEG image data.

## 5.6 Error Handling

### 5.6.1 SOAP FAULT MESSAGE

**Test Case ID:** MEDIA-7-1-4

**Specification Coverage:** Request stream URI.

**Feature Under Test:** GetStreamUri

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify that DUT generates SOAP 1.2 fault message to the invalid GetStreamUriRequest message (Invalid Media Profile).

**Pre-Requisite:** Media service is supported by DUT. Real-time Streaming is supported by DUT. Media service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetStreamUriRequest** message with invalid media profile.
4. DUT will generate the SOAP 1.2 fault message (**InvalidArgVal/NoProfile**).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT did not send correct SOAP 1.2 fault message (fault code, namespace etc).

**NOTE:** See Annex A.2 for correct syntax for the StreamSetup element in GetStreamUri requests.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.6.2 START MULTICAST - INVALID PROFILE TOKEN

**Test Case ID:** MEDIA-7-1-5

**Specification Coverage:** Start multicast streaming

**Feature Under Test:** StartMulticastStreaming

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify the behavior of StartMulticastStreaming command in case of invalid Profile Token.

**Pre-Requisite:** Media service is supported by DUT. Real-time Streaming is supported by DUT. Media service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **StartMulticastStreamingRequest** message (**invalid ProfileToken**).
4. The DUT will generate a SOAP 1.2 fault message (**InvalidArgVal/NoProfile** or **ActionNotSupported** (for case when multicast is not supported)).

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

**NOTE:** See Annex in [ONVIF Base Test] for Invalid SOAP 1.2 fault message definition.

**NOTE:** Other faults than specified in the test are acceptable but specified are preferable.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## 5.7 Capabilities

### 5.7.1 MEDIA SERVICE CAPABILITIES

**Test Case ID:** MEDIA-8-1-1

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServiceCapabilities (for Media Service)

**WSDL Reference:** media.wsdl

**Test Purpose:** To verify Media Service Capabilities of the DUT.

**Pre-Requisite:** Media Service was received from the DUT.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilitiesRequest** message to retrieve media service capabilities of the DUT.
4. Verify the **GetServiceCapabilitiesResponse** from the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetServiceCapabilitiesResponse**.

### 5.7.2 GET SERVICES AND GET MEDIA SERVICE CAPABILITIES CONSISTENCY

**Test Case ID:** MEDIA-8-1-2

**Specification Coverage:** Capability exchange

**Feature Under Test:** GetServices, GetServiceCapabilities (for Media Service)

**WSDL Reference:** devicemgmt.wsdl, media.wsdl

**Test Purpose:** To verify Get Services and Media Service Capabilities consistency.

**Pre-Requisite:** None.

**Test Configuration:** ONVIF Client and DUT

**Test Procedure:**

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServicesRequest** message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
4. Verify the **GetServicesResponse** message from the DUT.
5. ONVIF Client will invoke **GetServiceCapabilitiesRequest** message to retrieve Media service capabilities of the DUT.
6. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

**Test Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- The DUT did not send valid **GetServicesResponse** message.
- The DUT did not send valid **GetServiceCapabilitiesResponse** message.
- The DUT sent different Capabilities in **GetServicesResponse** message and in **GetServiceCapabilitiesResponse** message.

**NOTE:** Service will be defined as Media service if it will have Namespace element that is equal to “http://www.onvif.org/ver10/media/wsdl”.

# Annex A Helper Procedures and Additional Notes

This section describes the meaning of the following definitions. These definitions are used in the test case description.

## A.1 Invalid Media Profile

Media profile token is a string of max length value of 64.

Invalid Media Profile:

a. a) A string which is not formed according to the rules of RFC 952.

b. b) A string which exceeds max length value of 64.

## A.2 StreamSetup syntax for GetStreamUri

The following media stream setups for GetStreamUri are covered in this Test Specification:

1. RTP Unicast over UDP
2. RTP over RTSP over HTTP over TCP
3. RTP over RTSP over TCP

The correct syntax for the StreamSetup element for these media stream setups are as follows:

1. RTP Unicast over UDP

```
<?xml version="1.0"?>
<StreamSetup>
<StreamType>RTP_unicast</StreamType>
<Transport>
<Protocol>UDP</Protocol>
</Transport>
</StreamSetup>
```

2. RTP over RTSP over HTTP over TCP

```
<?xml version="1.0"?>
<StreamSetup>
<StreamType>RTP_unicast</StreamType>
<Transport>
<Protocol>HTTP</Protocol>
</Transport>
</StreamSetup>
```

3. RTP over RTSP over TCP

```
<?xml version="1.0"?>
<StreamSetup>
<StreamType>RTP_unicast</StreamType>
<Transport>
<Protocol>RTSP</Protocol>
</Transport>
</StreamSetup>
```

## A.3 Create Empty Profile

For the execution of test cases with profile configurations, ONVIF Client has to find, create or configure empty media profile.

ONVIF Client follows the following procedure to find, create or configure empty media profile.

If there is profile with fixed attribute equal to false, then follows the following procedure:

If there are no profiles with fixed = “false” remove all configurations from one fixed profile and use this profile for test. If there are no profiles skip other steps and fail test.

1. ONVIF Client will invoke **GetProfilesRequest** message to retrieve complete profiles list.
2. Verify the **GetProfilesResponse** message from the DUT.
3. ONVIF Client will invoke **DeleteProfileRequest** message (ProfileToken = Token1, where Token1 is the first ProfileToken from the **GetProfilesResponse** with fixed = false) to delete Profile.

4. Verify the **DeleteProfileResponse** message from the DUT.
5. Return to step with profile creation of the test procedure.

**NOTE:** See Annex A.5 for Name and Token Parameters Length limitations.

## A.4 Get Guaranteed Number of Video Encoder Instances and Get Video Encoder Configuration Options Consistency Verification Description

The following general rules (see below) were used to verify consistency between:

1. GetVideoEncoderConfigurationOptionsResponse message requested without Profile token and Video Encoder Configuration token (general options for the DUT).
2. GetGuaranteedNumberOfVideoEncoderInstancesResponse messages for all Video Source Configurations supported by the DUT.

For **JPEG** encoding:

- If DUT supports **JPEG encoding** (i.e. GetVideoEncoderConfigurationOptionsResponse message **contains Options.JPEG** element), then the following behavior
  - will be assumed as **correct**:
    - **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no JPEG elements or contains a JPEG element with value > 0**.
  - will be assumed as **incorrect**:
    - **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains a JPEG element with value = 0**
- If DUT **does not support JPEG** encoding (i.e. GetVideoEncoderConfigurationOptionsResponse message **does not contain Options.JPEG** element), then the following behavior
  - will be assumed as **correct**:
    - **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no JPEG elements or contains a JPEG element with value = 0**.

- will be assumed as **incorrect**:
  - **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains a JPEG element with value > 0**.

For **MPEG-4** encoding:

- If DUT **supports MPEG-4** encoding (i.e. GetVideoEncoderConfigurationOptionsResponse message **contains Options.MPEG** element), then the following behavior
  - will be assumed as **correct**:
    - **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no MPEG elements or contains an MPEG element with value > 0**.
  - will be assumed as **incorrect**:
    - **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains an MPEG element with value = 0**
- If DUT **does not support MPEG-4** encoding (i.e. GetVideoEncoderConfigurationOptionsResponse message **does not contain Options.MPEG** element), then the following behavior
  - will be assumed as **correct**:
    - **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no MPEG elements or contains an MPEG element with value = 0**.
  - will be assumed as **incorrect**:
    - **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains an MPEG element with value > 0**.

For **H.264** encoding:

- If DUT **supports H.264** encoding (i.e. GetVideoEncoderConfigurationOptionsResponse message **contains Options.H264** element), then the following behavior
  - will be assumed as **correct**:

- **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no H264 elements or contains an H264 element with value > 0**.

- will be assumed as **incorrect**:

- **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains an H264 element with value = 0**

- If DUT **does not support H.264** encoding (i.e. GetVideoEncoderConfigurationOptionsResponse message **does not contain Options.H264** element), then the following behavior

- will be assumed as **correct**:

- **For all** Video Source Configurations GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains no H264 elements or contains an H264 element with value = 0**.

- will be assumed as **incorrect**:

- **For at least one** Video Source Configuration GetGuaranteedNumberOfVideoEncoderInstancesResponse message **contains an H264 element with value > 0**.

## A.5 Name and Token Parameters

There are following limitations on maximum length of Name and Token parameters that sell be used during tests by ONVIF Device Test Tool to prevent faults from DUT:

1. Name shall be less than or equal to 64 characters (only readable characters accepted).
2. Token shall be less than or equal to 64 characters (only readable characters accepted).

UTF-8 character set shall be used for Name and Token.

**NOTE:** these limitations will not be used, if ONVIF Device Test Tool reuses values that were received from the DUT.

## A.6 Create or Configure Empty Profile

**Name:** HelperCreateEmptyProfile

**Procedure Purpose:** Helper procedure to create or configure empty media profile.

**Pre-requisite:** Media Service is supported by the DUT.

**Input:** The profile name (*profileName*) of profile to be created.

**Returns:** The profile token (*profileToken*), Media Profile to restore (*profileToRestore*) or Media Profile token to delete (*profileToDelete*).

**Procedure:**

1. ONVIF Client invokes **CreateProfile** with with parameters
   - Name := *profileName*
   - Token skipped
2. If DUT returns **CreateProfileResponse** message with parameters
   - Profile := *newProfile*

   2.1. Set the following:
   - *profileToken* := *newProfile*.token
   - *profileToDelete* := *newProfile*.token

   2.2. Skip other steps.
3. If DUT returns **env:Receiver\ter:Action\ter:MaxNVTProfiles** SOAP 1.2 fault:

   3.1. If profile was deleted during HelperCreateEmptyProfile procedure, FAIL the test and skip other steps.

   3.2. ONVIF Client will invoke **GetProfiles** message.

   3.3. DUT responds with **GetProfilesResponse** with parameters:
   - Profiles =: *profileList*

   3.4. If *profileList* is empty, FAIL the test and skip other steps.

   3.5. If *profileList* contains profile item with Profile.fixed = false:

   3.5.1. Set the following:
   - *profileToRestore* := the first *profileList*[0].Profile with fixed = false

   3.5.2. Set the following:
   - *profileToDelete* := *profileToRestore*.token

3.5.3. ONVIF Client invokes **DeleteProfile** message with parameters

- ProfileToken := *profileToDelete*

3.5.4. DUT responds with **DeleteProfileResponse**.

3.5.5. Go to the step 1.

3.6. If *profileList* does not contain profile item with Profile.fixed = false

3.6.1. Set the following:

- *profileToConfigure* := *profileList*[0].Profile
- *profileToRestore* := *profileList*[0].Profile

3.6.2. If *profileToConfigure* contains VideoSourceConfiguration:

3.6.2.1. ONVIF Client invokes **RemoveVideoSourceConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.2.2. DUT responds with **RemoveVideoSourceConfigurationResponse**

3.6.3. If *profileToConfigure* contains AudioSourceConfiguration

3.6.3.1. ONVIF Client invokes **RemoveAudioSourceConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.3.2. DUT responds with **RemoveAudioSourceConfigurationResponse**

3.6.4. If *profileToConfigure* contains VideoEncoderConfiguration

3.6.4.1. ONVIF Client invokes **RemoveVideoEncoderConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.4.2. DUT responds with **RemoveVideoEncoderConfigurationResponse**

3.6.5. If *profileToConfigure* contains AudioEncoderConfiguration

3.6.5.1. ONVIF Client invokes **RemoveAudioEncoderConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.5.2. DUT responds with **RemoveAudioEncoderConfigurationResponse**

3.6.6. If *profileToConfigure* contains VideoAnalyticsConfiguration

3.6.6.1. ONVIF Client invokes **RemoveVideoAnalyticsConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.6.2. DUT responds with **RemoveVideoAnalyticsConfigurationResponse**

3.6.7. If *profileToConfigure* contains PTZConfiguration

3.6.7.1. ONVIF Client invokes **RemovePTZConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.7.2. DUT responds with **RemovePTZConfigurationResponse**

3.6.8. If *profileToConfigure* contains MetadataConfiguration

3.6.8.1. ONVIF Client invokes **RemoveMetadataConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.8.2. DUT responds with **RemoveMetadataConfigurationResponse**

3.6.9. If *profileToConfigure* contains AudioOutputConfiguration

3.6.9.1. ONVIF Client invokes **RemoveAudioOutputConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.9.2. DUT responds with **RemoveAudioOutputConfigurationResponse**

3.6.10. If *profileToConfigure* contains AudioDecoderConfiguration

3.6.10.1. ONVIF Client invokes **RemoveAudioDecoderConfiguration** with parameters:

- ProfileToken := *profileToConfigure*.token

3.6.11. DUT responds with **RemoveAudioDecoderConfigurationResponse**

4. If DUT returns any other SOAP 1.2 fault, FAIL the test and skip other steps.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **CreateProfileResponse** message.
- DUT did not send **GetProfilesResponse** message.
- DUT did not send **DeleteProfileResponse** message.
- DUT did not send **RemoveVideoSourceConfigurationResponse** message.
- DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- DUT did not send **RemoveVideoEncoderConfigurationResponse** message.
- DUT did not send **RemoveAudioEncoderConfigurationResponse** message.
- DUT did not send **RemoveVideoAnalyticsConfigurationResponse** message.
- DUT did not send **RemovePTZConfigurationResponse** message.
- DUT did not send **RemoveMetadataConfigurationResponse** message.

## A.7 Restore Media Profile

**Name:** HelperRestoreMediaProfile

**Procedure Purpose:** Helper procedure to restore media profile.

**Pre-requisite:** Media Service is received from the DUT.

**Input:** Media Profile (*profileToRestore*).

**Returns:** None.

**Procedure:**

1. If *profileToRestore* was deleted during Annex A.6:
   1.1. ONVIF client invokes **CreateProfile** with parameters
      - Name := *profileToRestore*.Name
      - Token := *profileToRestore*
   1.2. The DUT responds with **CreateProfileResponse** message with parameters
2. ONVIF clients restore all *profileToRestore* configurations.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **CreateProfileResponse** message.
- DUT did not send **GetProfilesResponse** message.
- DUT did not send **DeleteProfileResponse** message.
- DUT did not send **RemoveVideoSourceConfigurationResponse** message.
- DUT did not send **RemoveAudioSourceConfigurationResponse** message.
- DUT did not send **RemoveVideoEncoderConfigurationResponse** message.
- DUT did not send **RemoveAudioEncoderConfigurationResponse** message.
- DUT did not send **RemoveVideoAnalyticsConfigurationResponse** message.
- DUT did not send **RemovePTZConfigurationResponse** message.
- DUT did not send **RemoveMetadataConfigurationResponse** message.

## A.8 Delete Media Profile

**Name:** HelperDeleteMediaProfile

**Procedure Purpose:** Helper procedure to delete media profile.

**Pre-requisite:** Media Service is received from the DUT.

**Input:** Media Profile token (*profileToDelete*).

**Returns:** None.

**Procedure:**

1. ONVIF Client invokes **DeleteProfile** with parameters:
   - ProfileToken =: *profileToDelete*
2. DUT responds with **DeleteProfileResponse**.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **DeleteProfileResponse** message.

## A.9 Get Video Source Configurations List

**Name:** HelperGetVideoSourceConfigurationsList

**Procedure Purpose:** Helper procedure to retrieve Video Source Configurations List.

**Pre-requisite:** Media Service is received from the DUT.

**Input:** None.

**Returns:** Video Source Configurations list (*videoSourceConfList*).

**Procedure:**

1. ONVIF Client invokes **GetVideoSourceConfigurations**.
2. DUT responds with **GetVideoSourceConfigurationsResponse** with parameters:
   - Configurations list =: *videoSourceConfList*
3. If *videoSourceConfList* is empty, FAIL the test.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetVideoSourceConfigurationsResponse** message.

## A.10 Get Metadata Configurations List

**Name:** HelperGetMetadataConfigurationsList

**Procedure Purpose:** Helper procedure to retrieve Metadata Configurations List.

**Pre-requisite:** Media Service is received from the DUT.

**Input:** None.

**Returns:** Metadata Configurations list (*metadataConfList*).

**Procedure:**

1. ONVIF Client invokes **GetMetadataConfigurations**.
2. DUT responds with **GetMetadataConfigurationsResponse** with parameters:
   - Configurations list =: *metadataConfList*

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetMetadataConfigurationsResponse** message.

## A.11 Get Audio Source Configurations List

**Name:** HelperGetAudioSourceConfigurationsList

**Procedure Purpose:** Helper procedure to retrieve Audio Source Configurations List.

**Pre-requisite:** Media Service is received from the DUT. Audio is supported by the DUT.

**Input:** None.

**Returns:** Audio Source Configurations list (*audioSourceConfList*).

**Procedure:**

1. ONVIF Client invokes **GetAudioSourceConfigurations**.
2. DUT responds with **GetAudioSourceConfigurationsResponse** with parameters:
   - Configurations list =: *audioSourceConfList*
3. If *audioSourceConfList* is empty, FAIL the test.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioSourceConfigurationsResponse** message.

## A.12 Get Audio Output Configurations List

**Name:** HelperGetAudioOutputConfigurationsList

**Procedure Purpose:** Helper procedure to retrieve Audio Output Configurations List.

**Pre-requisite:** Media Service is received from the DUT. Audio Outputs is supported by the DUT.

**Input:** None.

**Returns:** Audio Output Configurations list (*audioOutputConfList*).

**Procedure:**

1. ONVIF Client invokes **GetAudioOutputConfigurations**.
2. DUT responds with **GetAudioOutputConfigurationsResponse** with parameters:
   - Configurations list =: *audioOutputConfList*
3. If *audioOutputConfList* is empty, FAIL the test.

**Procedure Result:**

**PASS –**

- DUT passes all assertions.

**FAIL –**

- DUT did not send **GetAudioOutputConfigurationsResponse** message.