Driving IP-based physical security through global standardization

ONVIF^{TM}

ONVIF Media 2 Configuration Test Specification

Version 16.07

August 8, 2016

Driving IP-based physical security through global standardization

THIS DOCUMENT IS PROVIDED "AS IS," AND THE CORPORATION AND ITS MEMBERS AND THEIR AFFILIATES, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

IN NO EVENT WILL THE CORPORATION OR ITS MEMBERS OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT, WHETHER OR NOT (1) THE CORPORATION, MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR (2) SUCH DAMAGES WERE REASONABLY FORESEEABLE, AND ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS AND THEIR RESPECTIVE AFFILIATES TO THE CORPORATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE CORPORATION.

Revision History

Ver.	Date	Description
16.06	Feb 12, 2016	Original publication
16.06	Feb 19, 2016	The step 11 and diagram have been updated in test 4.1.3 updated according to the feedback of Bhetanabottla Sriram
16.06	Feb 25, 2016	The issue in last step in tests MEDIA2-4-2-3 has been fixed
16.06	Mar 8, 2016	The tests 4.1.1 – 4.1.3 have been updated according to the feedback from F2F Tokyo. OSD configuration tests have been added
16.06	Mar 15, 2016	The tests MEDIA2-4-4-1 - MEDIA2-4-4-4 have been updated according to Fredrik’s feedback.
16.06	Apr 4, 2016	The response for SetOSD and DeleteOSD requests have been updated according to the notes from Sano Hiroyuki
16.07	June 23, 2016	Test numbering and document version have been fixed
16.07	July 13, 2016	F2F commnets implemented.
16.07	July 14, 2016	Notes from F. Svensson implemented. Tests sequences updated.
16.07	July 27, 2016	Review comments implemented
16.07	August 8, 2016	Comments from Nicolas Brochu implemented

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 Media2 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 Media2 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 is to provide test cases to test individual requirements of ONVIF devices according to ONVIF Media2 Service which is defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

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

This specification does not address the following.

1. Product use cases and non-functional (performance and regression) testing.
2. SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
3. Network protocol implementation Conformance test for HTTP, HTTPS, RTP and RTSP protocol.
4. Poor streaming performance test (audio/video distortions, missing audio/video frames, incorrect lib synchronization etc.).
5. 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 Media2 Service and some of the optional requirements.

This ONVIF Media2 Test Specification covers ONVIF Media2 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.

1.1.1 Media Configuration

Media Configuration section covers the test cases needed for the verification of media2 service features as mentioned in [ONVIF Network Interface Specs]. Media2 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. H.264
    b. H.265
3. Audio Codec:
    a. G.711
    b. AAC

2 Terms and Definitions

2.1 Definitions

This section defines terms that are specific to the ONVIF Media2 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.

2.2 Abbreviations

This section describes abbreviations used in this document.

AAC	Advanced Audio Coding
JPEG	Joint Photographic Experts Group
TTL	Time To Live

3 Test Overview

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

3.1 Test Setup

3.1.1 Network Configuration for DUT

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

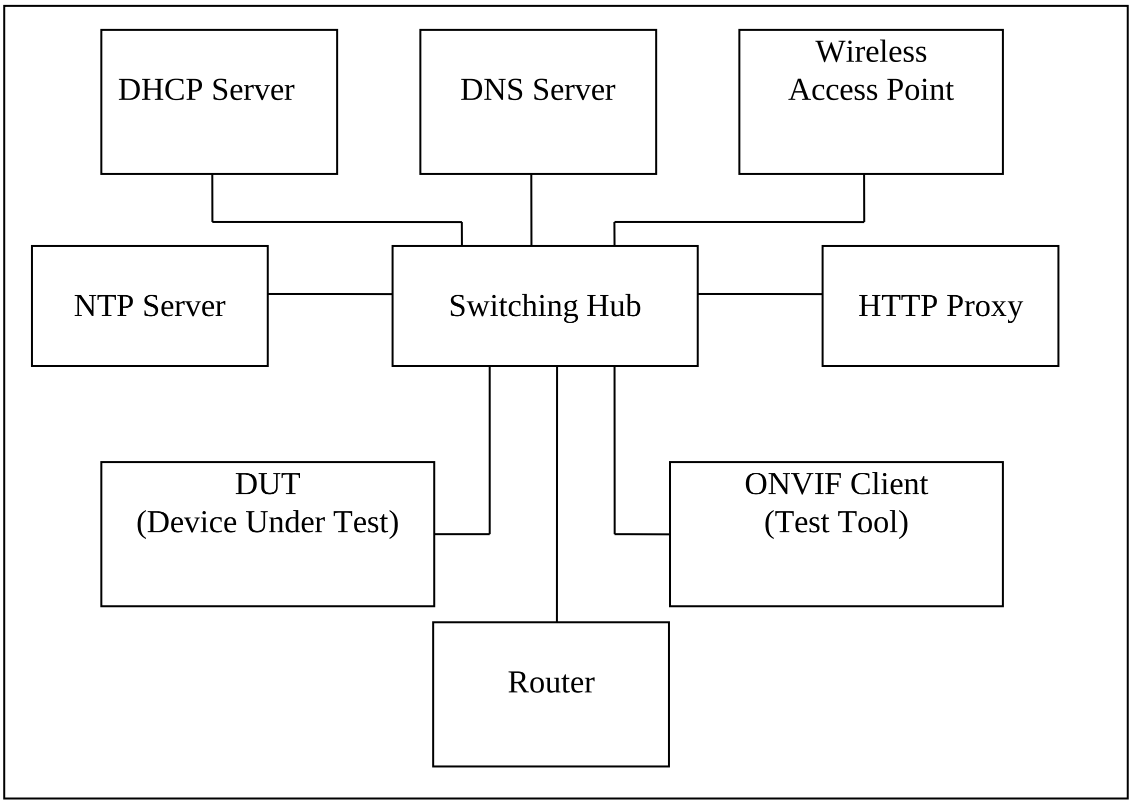


Figure 1: Test Configuration for DUT

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.

Router: provides router advertisements for IPv6 configuration.

3.2 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.

3.3 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.

3.3.1 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 either H264 or H265 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 configurations 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 4 for Media Configuration Test Cases.

4 Media Configuration Test Cases

4.1 Media Profile

4.1.1 READY TO USE MEDIA PROFILE FOR VIDEO STREAMING

Test Label: Media Configuration ready-to-use Media Service 2.0 Profile.

Test Case ID: MEDIA2-4-1-1

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles

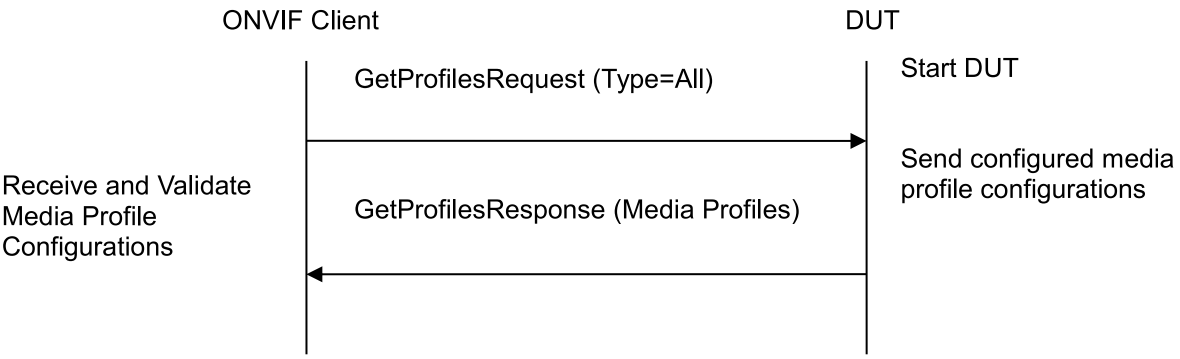
WSDL Reference: media2.wsdl

Test Purpose: To verify that DUT has a ready-to-use Media Service 2.0 Profile for streaming video (either H.264 or H.265).

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetProfilesRequest (Type=All)
Start DUT
Send configured media profile configurations
Receive and Validate Media Profile Configurations
GetProfilesResponse (Media Profiles)

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetProfilesRequest (Type=All) message to retrieve existing media2 profiles of the DUT.
4. Verifies that the DUT returns at least one media profile with video configuration (video source and video encoder) and video encoder encoding equals to the one of values ("H264" or "H265”) 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 did not provide at least one media profile, which meets the following conditions: configurations video source is not empty AND configurations video encoder is not empty AND configurations video encoder encoding equals to one of the following values:

- "H264"
- "H265”

4.1.2 CREATE MEDIA PROFILE WITH PRE-DEFINED CONFIGURATION

Test Label: Media Configuration Create Media Profile with Pre-Defined Configuration.

Test Case ID: MEDIA2-4-1-2

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles, CreateProfile, DeleteProfile

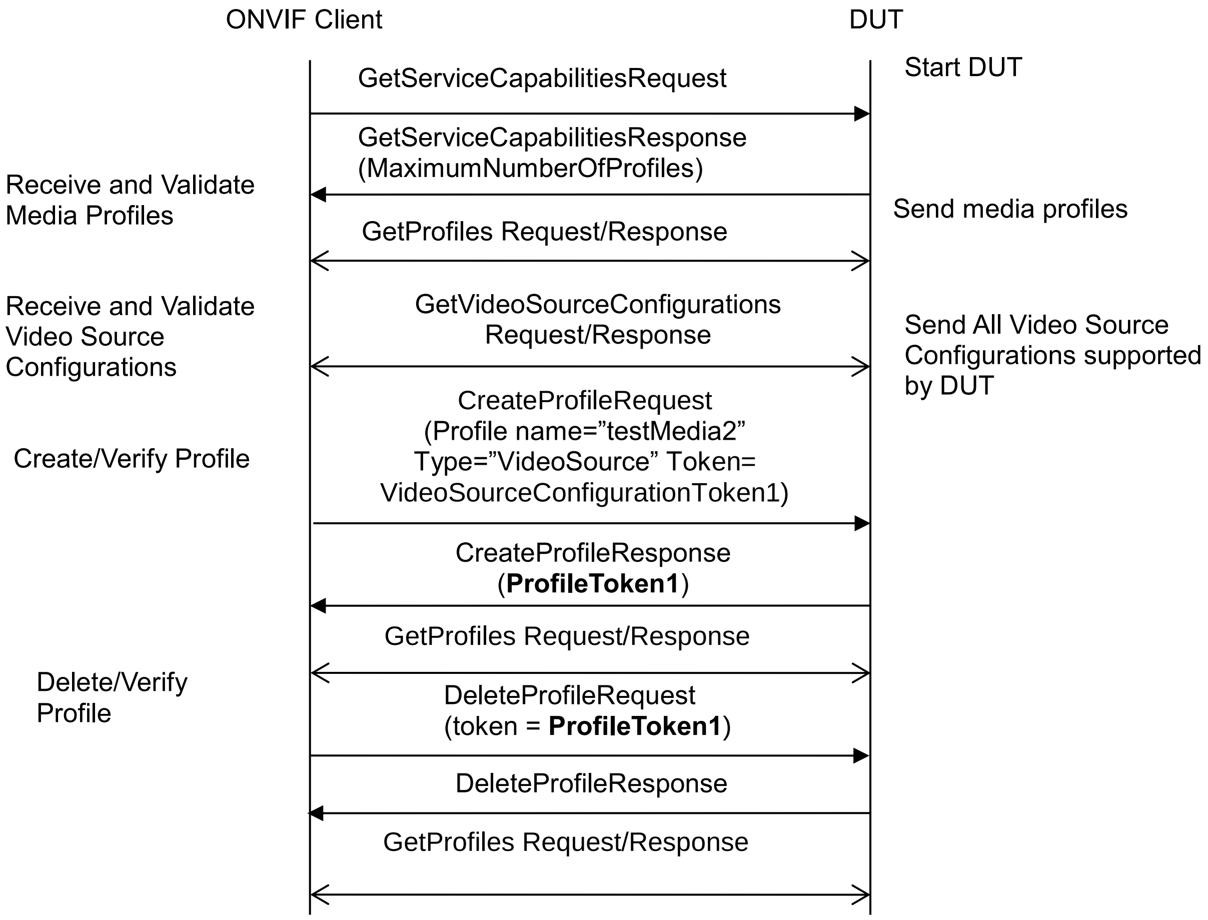
WSDL Reference: media2.wsdl

Test Purpose: To verify the DUT can create media profile with populated configuration parameter.

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetServiceCapabilitiesRequest
GetServiceCapabilitiesResponse (MaximumNumberOfProfiles)
Receive and Validate Media Profiles
Send media profiles
GetProfiles Request/Response
Receive and Validate Video Source Configurations
GetVideoSourceConfigurations Request/Response
Send All Video Source Configurations supported by DUT
Create/Verify Profile
CreateProfileRequest (Profile name="testMedia2" Type="VideoSource" Token= VideoSourceConfigurationToken1)
CreateProfileResponse (ProfileToken1)
GetProfiles Request/Response
Delete/Verify Profile
DeleteProfileRequest (token = ProfileToken1)
DeleteProfileResponse
GetProfiles Request/Response

Test Procedure:

1 Start an ONVIF Client.

2 Start the DUT.

3 ONVIF Client invokes GetServiceCapabilitiesRequest message to retrieve maximum number of profiles (**MaximumNumberOfProfiles**) supported by DUT.

4 ONVIF Client invokes GetProfilesRequest message to retrieve existing media profiles configurations of the DUT.

5 ONVIF Client invokes GetVideoSourceConfigurationsRequest with no parameters. DUT returns the list of available Video Source Configurations in GetVideoSourceConfigurationsResponse.

6 ONVIF Client selects the first configuration from this list and saves token of this configuration in **VideoSourceConfigurationToken1** variable.

7 ONVIF Client invokes CreateProfileRequest message to create a new media profile with pre-defined configuration using the following parameters: Profile name="testMedia2" and configuration list contains item with Type="VideoSource" and Token=**VideoSourceConfigurationToken1**.

8 ONVIF Client receives CreateProfileResponse response and save profile token from CreateProfileResponse in **ProfileToken1** variable.

9 ONVIF Client invokes GetProfilesRequest (Profile Token = **ProfileToken1**).

10 The DUT returns GetProfilesResponse with only one profile in the list of profiles.

11 ONVIF Client verifies that the list of profiles contains one media profile (the profile with token= **ProfileToken1**), if the list contains more than one profile or the list does not contain the profile with token= **ProfileToken1** then the test should FAIL.

12 ONVIF Client verifies that video source configuration token for the profile from GetProfilesResponse equals to **VideoSourceConfigurationToken1**.

13 ONVIF Client invokes DeleteProfileRequest with token = **ProfileToken1**. It retrieves DeleteProfileResponse from DUT to verify that the profile has been deleted successfully.

14 ONVIF Client invokes GetProfilesRequest (Profile Token = **ProfileToken1**) message to check the existence of deleted media profile.

15 The DUT generates SOAP 1.2 fault message (InvalidArgVal / NoProfile).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetProfilesResponse message.

Maximum Number of Profiles is reached and GetProfilesResponse does not contain at least one profile with Fixed = “false”.

The DUT did not send CreateProfileResponse message.

The maximum number of profiles is reached and there is no profile with Fixed=”false”

The DUT sent more than one profile in GetProfilesResponse if the profile token is specified by ONVIF Client in GetProfilesRequest.

The DUT did not set ‘fixed’ attribute of created media profile to ‘false’

The DUT did not send DeleteProfileResponse message.

The DUT did not send SOAP 1.2 fault message (InvalidArgVal/NoProfile).

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

Note: See Annex A.2 for Name and Token Parameters Length limitations.

4.1.3 DYNAMIC MEDIA PROFILE CONFIGURATION

Test Label: Media Configuration Dynamic Media Profile Configuration.

Test Case ID: MEDIA2-4-1-3

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles, CreateProfile, DeleteProfile, AddConfiguration, RemoveConfiguration, GetVideoEncoderConfigurations, GetVideoSourceConfigurations.

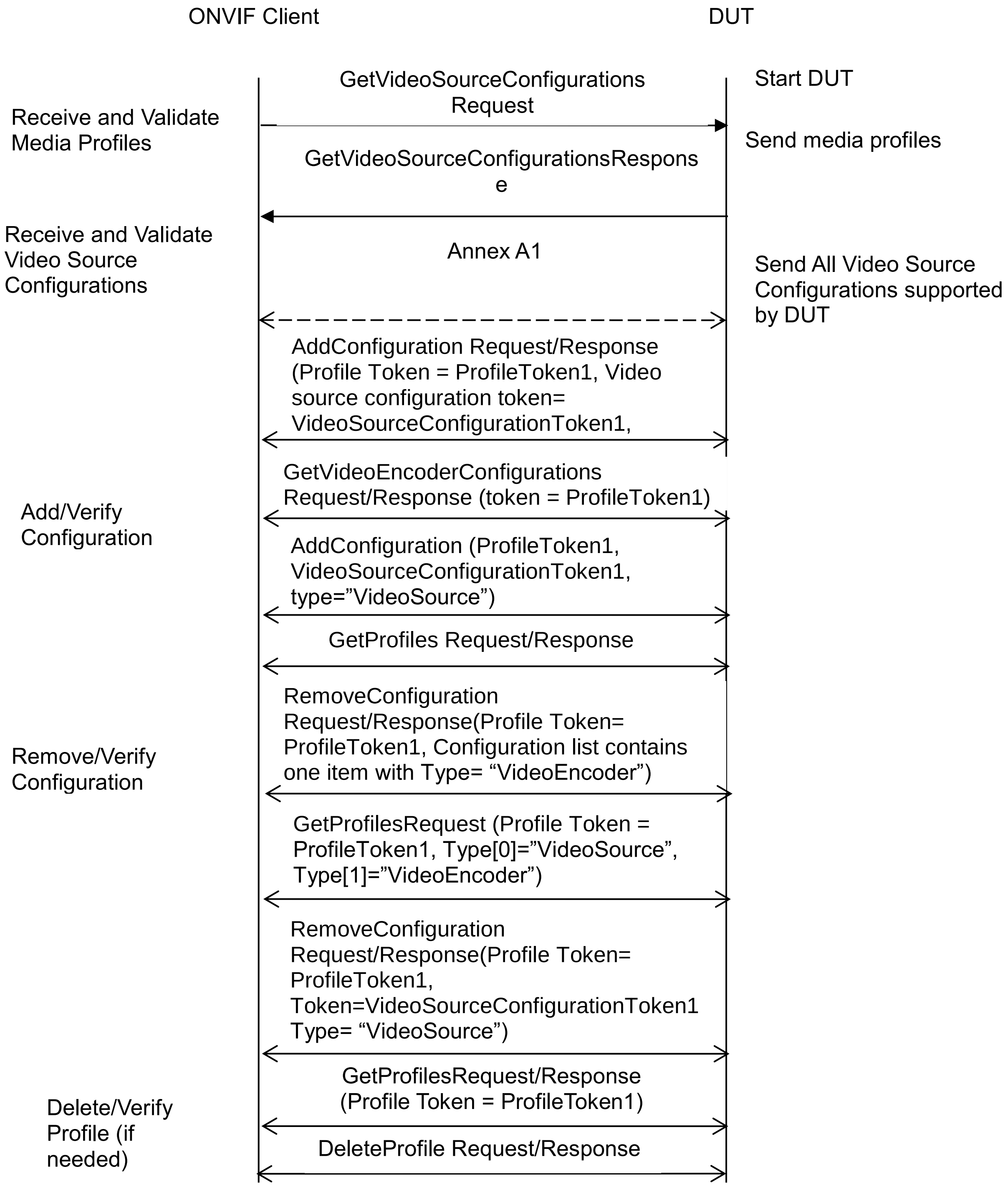
WSDL Reference: media2.wsdl

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

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetVideoSourceConfigurations Request
Receive and Validate Media Profiles
Send media profiles
GetVideoSourceConfigurationsRespons e
Receive and Validate Video Source Configurations
Annex A1
Send All Video Source Configurations supported by DUT
AddConfiguration Request/Response (Profile Token = ProfileToken1, Video source configuration token= VideoSourceConfigurationToken1,
GetVideoEncoderConfigurations Request/Response (token = ProfileToken1)
Add/Verify Configuration
AddConfiguration (ProfileToken1, VideoSourceConfigurationToken1, type=”VideoSource”)
GetProfiles Request/Response
RemoveConfiguration Request/Response(Profile Token= ProfileToken1, Configuration list contains one item with Type= “VideoEncoder”)
Remove/Verify Configuration
GetProfilesRequest (Profile Token = ProfileToken1, Type[0]=”VideoSource”, Type[1]=”VideoEncoder”)
RemoveConfiguration Request/Response(Profile Token= ProfileToken1, Token=VideoSourceConfigurationToken1 Type= “VideoSource”)
GetProfilesRequest/Response (Profile Token = ProfileToken1)
Delete/Verify Profile (if needed)
DeleteProfile Request/Response

Test Procedure:

1 Start an ONVIF Client.

2 Start the DUT.

3 ONVIF Client invokes GetVideoSourceConfigurations. DUT responds with GetVideoSourceConfigurationsResponse.

4 ONVIF Client verifies GetVideoSourceConfigurationsResponse, the response should contain at least one Video Source Configuration.

5 For each Video Source Configuration in GetVideoSourceConfigurationsResponse, ONVIF Client saves token of this configuration in **VideoSourceConfigurationToken1** variable and runs the steps 5.1 - 5.8:

5.1 ONVIF Client executes Annex **A.1** and saves recently created / updated profile token in **ProfileToken1** variable.

5.2 ONVIF Client invokes AddConfigurationRequest message (Profile Token = **ProfileToken1**, Video source configuration token= **VideoSourceConfigurationToken1**, type=”VideoSource”) to add video source configuration to the profile.

5.3 The DUT sends AddConfigurationResponse message indicating successful addition of video source configuration.

5.4 ONVIF Client invokes GetVideoEncoderConfigurationsRequest with Profile token = **ProfileToken1** as input parameter.

5.5 The DUT sends GetVideoEncoderConfigurationsResponse with list of available video encoder configurations for given profile.

5.6 ONVIF Client selects first video encoder configuration in GetVideoEncoderConfigurationsResponse, saves the token of this configuration in **VideoEncoderConfigurationToken1** variable and runs the steps 5.6.1 - 5.6.10:

5.6.1 ONVIF Client invokes AddConfigurationRequest message (Profile Token = **ProfileToken1**, Video encoder configuration token = **VideoEncoderConfigurationToken1**, type=”VideoEncoder”) to add video encoder configuration to the profile.

5.6.2 The DUT sends AddConfigurationResponse message indicating successful addition of video source configuration.

5.6.3 ONVIF Client invokes GetProfilesRequest (Profile Token = **ProfileToken1**) to get Profile with video source and video encoder configurations.

5.6.4 The DUT returns GetProfilesResponse with only one profile in the list of profiles.

5.6.5 ONVIF Client verifies that the list of profiles contains only one media profile (the profile with token= **ProfileToken1**), if the list contains more than one profile or the list does not contain the profile with token= **ProfileToken1** then the test should FAIL.

5.6.6 ONVIF Client verifies that video source configuration token for the profile from GetProfilesResponse equals to **VideoSourceConfigurationToken1**.

5.6.7 ONVIF Client verifies that video encoder configuration token for the profile from GetProfilesResponse equals to **VideoEncoderConfigurationToken1**.

5.6.8 ONVIF Client invokes RemoveConfigurationRequest message (Profile Token= **ProfileToken1,** Configuration list contains one item with Type= “VideoEncoder”) to remove video encoder configuration from a media profile.

5.6.9 The DUT sends RemoveConfigurationResponse message indicating successfully removal of video source and video encoder configuration.

5.6.10 ONVIF client invokes GetProfilesRequest (Profile Token = **ProfileToken1**, Type[0]=”VideoSource”, Type[1]=”VideoEncoder”). The DUT returns GetProfilesResponse with only one profile in the list of profiles. ONVIF Client verifies that the video encoder configuration was removed from the profile and video source configuration is still exists in media profile.

5.7 ONVIF Client invokes RemoveConfigurationRequest message (Profile Token= **ProfileToken1,** Configuration list contains one item with Token=**VideoSourceConfigurationToken1** and Type= “VideoSource”) to remove video source configuration from a media profile.

5.8 ONVIF client invokes GetProfilesRequest (Profile Token = **ProfileToken1**, Type=”VideoSource”). The DUT returns GetProfilesResponse with only one profile in the list of profiles. ONVIF Client verifies that the video source configuration was removed from the profile.

6 If profile with Profile Token = **ProfileToken1** was created during test execution, then ONVIF Client invokes DeleteProfileRequest (Profile Token = **ProfileToken1**) message to delete the newly created media profile. Otherwise, ONVIF client skip the steps 7 - 9.

7 The DUT deletes the media profile and send DeleteProfileResponse message.

8 ONVIF Client invokes GetProfilesRequest (Profile Token = **ProfileToken1**) message to check the existence of deleted media profile.

9 The DUT generates SOAP 1.2 fault message (InvalidArgVal/NoProfile).

Test Result:

PASS –

The DUT passed all assertions.

FAIL –

The DUT did not send GetProfilesResponse message.

The GetProfilesResponse did not contain any Video Source Configuration.

The DUT did not send CreateProfileResponse message.

The DUT did not send AddConfigurationResponse message.

The DUT sent more than one profile in GetProfilesResponse if the profile token is specified by ONVIF Client in GetProfilesRequest.

The DUT did not set ‘fixed’ attribute of created media profile to ‘false’

The DUT did not send RemoveConfigurationResponse message.

The DUT did not send DeleteProfileResponse message.

The DUT did not send SOAP 1.2 fault message (InvalidArgVal/NoProfile).

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

Note: See Annex A.2 for Name and Token Parameters Length limitations.

Note: If profile was deleted during Annex A.1 execution, ONVIF Client restores the deleted profile and profile settings.

4.2 Video Configuration

4.2.1 VIDEO ENCODER CONFIGURATION

Test Label: Media Configuration Video Encoder Configuration

Test Case ID: MEDIA2-4-2-1

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles, GetVideoEncoderConfigurations

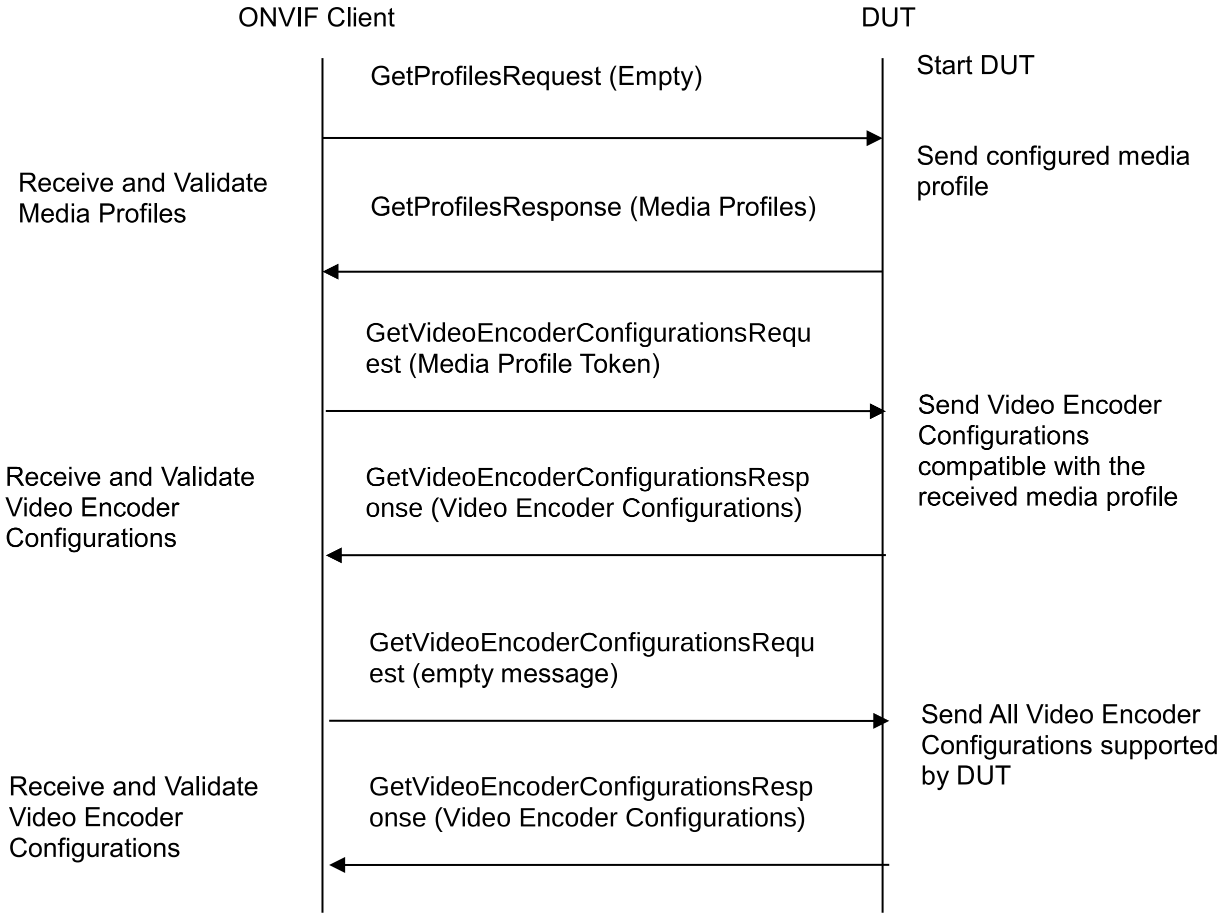
WSDL Reference: media2.wsdl

Test Purpose: To verify DUT sends All Video Encoder Configurations and Video Encoder Configurations compatible with specific profile.

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
GetProfilesRequest (Empty)
Start DUT
Send configured media profile
Receive and Validate Media Profiles
GetProfilesResponse (Media Profiles)
GetVideoEncoderConfigurationsRequest (Media Profile Token)
Send Video Encoder Configurations compatible with the received media profile
Receive and Validate Video Encoder Configurations
GetVideoEncoderConfigurationsResponse (Video Encoder Configurations)
GetVideoEncoderConfigurationsRequest (empty message)
Send All Video Encoder Configurations supported by DUT
Receive and Validate Video Encoder Configurations
GetVideoEncoderConfigurationsResponse (Video Encoder Configurations)

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetProfilesRequest.

4. DUT sends the list of existing media profiles in GetProfilesResponse message.

5. For each media profile from GetProfilesResponse, ONVIF Client saves this profile in **Profile1** variable and runs the steps 5.1 - 5.5.

   5.1. ONVIF Client invokes GetVideoEncoderConfigurations request with **Profile1** token argument to get the configurations, which are compatible with Profile1.

   5.2. DUT sends GetVideoEncoderConfigurationsResponse with the list of video encoder configurations, compatible with the received media profile token.

   5.3. ONVIF Client verifies the list of video encoder configurations sent by DUT.

   5.4. All video encoder configurations in GetVideoEncoderConfigurationsResponse should have unique token. If there are at least two configurations with identical tokens, then the test should FAIL.

5.5. Verify that video encoder configuration for **Profile1** is included in the GetVideoEncoderConfigurationsResponse message.

6. ONVIF Client invokes GetVideoEncoderConfigurationsRequest to retrieve all video encoder configurations supported by the DUT.

7. DUT sends the list of all video encoder configurations supported by DUT.

8. ONVIF Client verifies the list of video encoder configurations sent by DUT.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetProfilesResponse message.

DUT did not send valid GetVideoEncoderConfigurationsResponse message.

DUT returned GetVideoEncoderConfigurationsResponse with duplicate video encoder configuration tokens.

4.2.2 VIDEO ENCODER CONFIGURATIONS AND VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY

Test Label: Media Video Encoder Configurations and Video Encoder Configuration Options Consistency Validation.

Test Case ID: MEDIA2-4-2-2

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoEncoderConfigurations, GetVideoEncoderConfigurationOptions

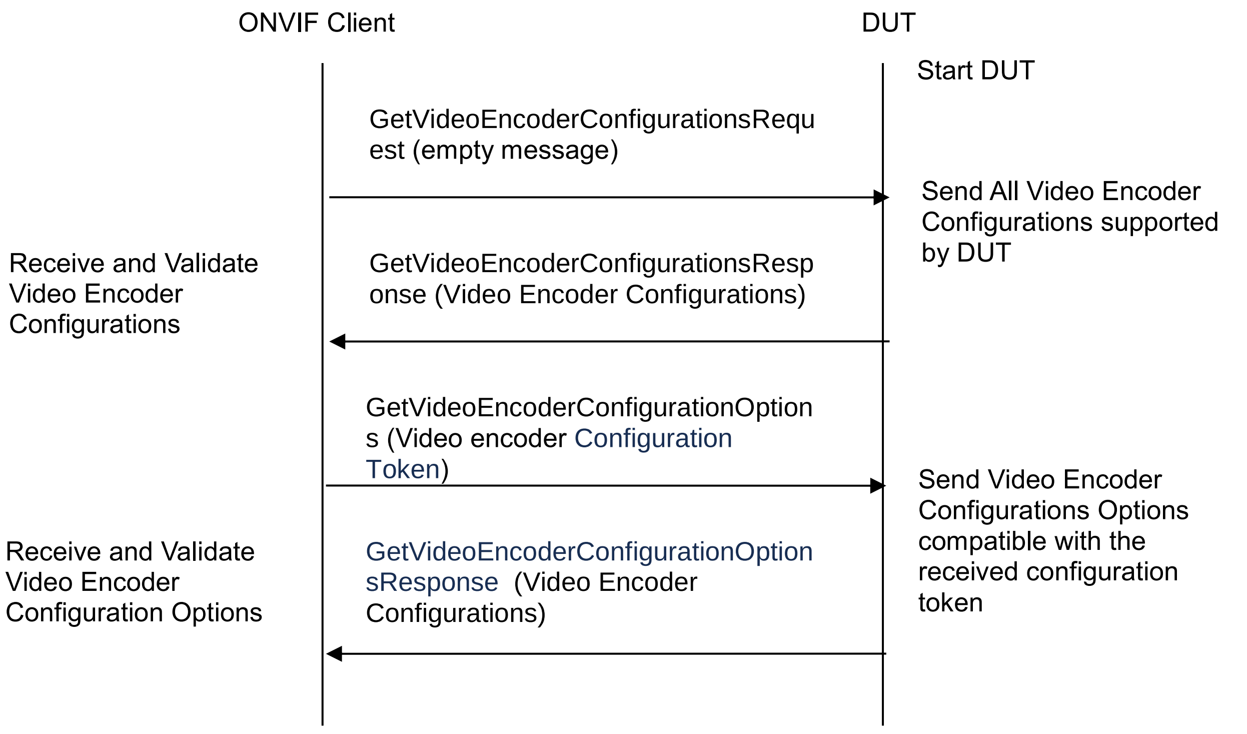
WSDL Reference: media2.wsdl

Test Purpose: To verify all video encoder configurations are consistent to video encoder configurations options

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetVideoEncoderConfigurationsRequest (empty message)
Send All Video Encoder Configurations supported by DUT
Receive and Validate Video Encoder Configurations
GetVideoEncoderConfigurationsResponse (Video Encoder Configurations)
GetVideoEncoderConfigurationOptions (Video encoder Configuration Token)
Send Video Encoder Configurations Options compatible with the received configuration token
Receive and Validate Video Encoder Configuration Options
GetVideoEncoderConfigurationOptionsResponse (Video Encoder Configurations)

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetVideoEncoderConfigurationsRequest to get all video encoder configurations from DUT.
4. DUT sends the list of video encoder configurations.
5. ONVIF Client verifies the list of video encoder configurations sent by DUT.
6. For each Video Encoder Configuration in GetVideoEncoderConfigurationsResponse, ONVIF Client saves this configuration in **Configuration1** and runs the steps 6.1 - 6.3

    6.1. ONVIF Client invokes GetVideoEncoderConfigurationOptionsRequest with **Configuration1** token as input argument.

    6.2. DUT sends GetVideoEncoderConfigurationOptionsResponse with the list of video encoder configuration options available for **Configuration1**.

    6.3. ONVIF Client verifies that **Configuration1** parameters are consistent to at least one option from GetVideoEncoderConfigurationOptionsResponse. See details of fields mapping in Annex A.3 VideoEncoderConfigurationOptions and VideoEncoderConfiguration mapping.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetVideoEncoderConfigurationsResponse message.

DUT did not send valid GetVideoEncoderConfigurationOptionsResponse message.

DUT failed consistency check according to Annex A.3

4.2.3 PROFILES AND VIDEO ENCODER CONFIGURATION OPTIONS CONSISTENCY

Test Label: Profile’s Video Encoder Configurations and Video Encoder Configuration Options Consistency Validation.

Test Case ID: MEDIA2-4-2-3

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles, GetVideoEncoderConfigurationOptions

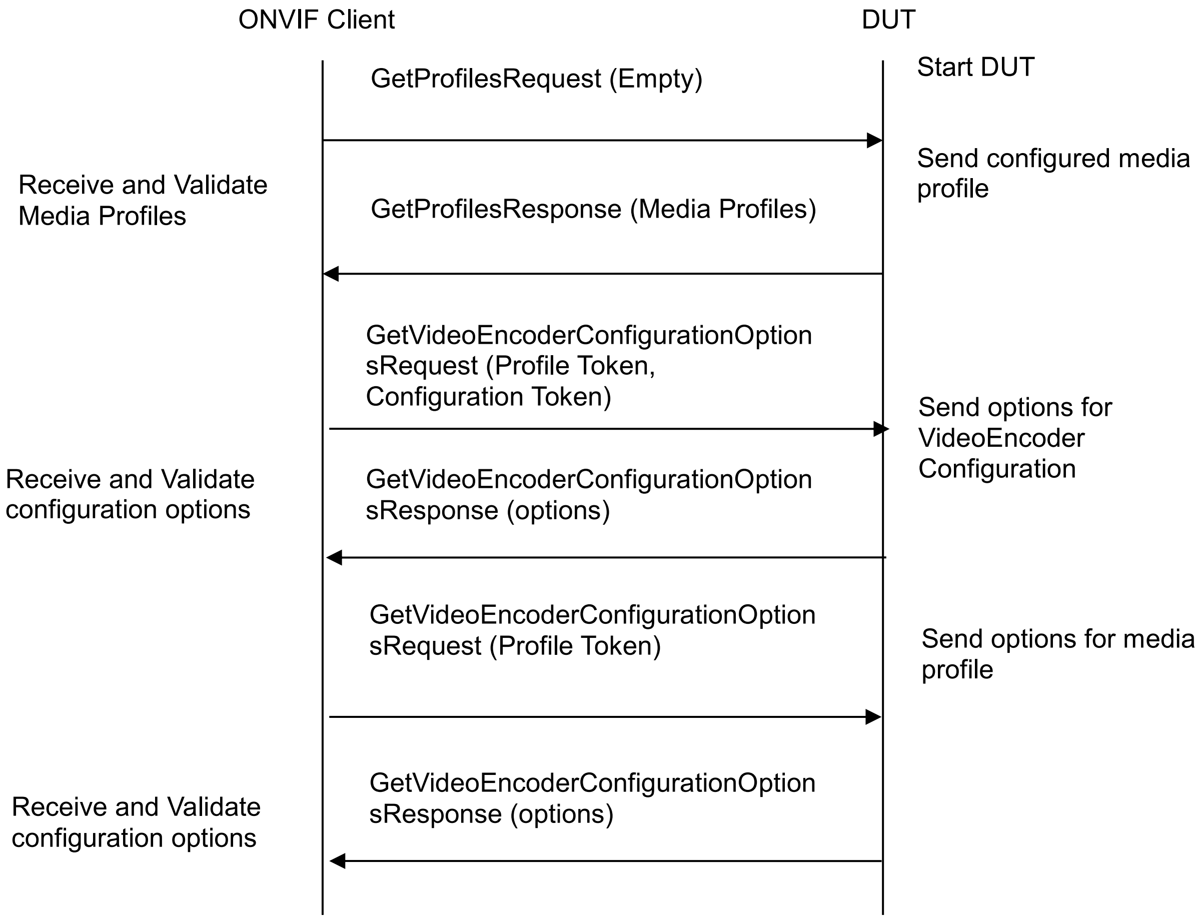
WSDL Reference: media2.wsdl

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

Pre-Requisite: Media2 Service feature is supported by DUT

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetProfilesRequest (Empty)
Receive and Validate Media Profiles
GetProfilesResponse (Media Profiles)
Send configured media profile
GetVideoEncoderConfigurationOptionsRequest (Profile Token, Configuration Token)
Send options for VideoEncoder Configuration
Receive and Validate configuration options
GetVideoEncoderConfigurationOptionsResponse (options)
GetVideoEncoderConfigurationOptionsRequest (Profile Token)
Send options for media profile
Receive and Validate configuration options
GetVideoEncoderConfigurationOptionsResponse (options)

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetProfilesRequest with Type="VideoEncoder" as input parameter.

4. DUT sends the list of existing media profiles in GetProfilesResponse message.

5. For each media profile from GetProfilesResponse, ONVIF Client saves this profile in **Profile1** variable, saves **Profile1** Configurations VideoEncoder configuration in **Configuration1** variable and runs the following steps

    5.1. ONVIF Client invokes GetVideoEncoderConfigurationOptionsRequest with **Configuration1** token and **Profile1** token as input arguments.

    5.2. DUT sends GetVideoEncoderConfigurationOptionsResponse with the list of configuration options.

    5.3. ONVIF Client verifies that **Configuration1** parameters are consistent to at least one option from GetVideoEncoderConfigurationOptionsResponse. See details of fields mapping in Annex A.3 VideoEncoderConfigurationOptions and VideoEncoderConfiguration mapping. ONVIF Client saves this option in **Option1** variable.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetProfilesResponse message.

DUT did not send valid GetVideoEncoderConfigurationsResponse message.

DUT failed consistency check according to Annex A.3

4.2.4 VIDEO ENCODER CONFIGURATIONS – ALL SUPPORTED VIDEO ENCODER CONFIGURATIONS

Test Label: Set All Supported Video Encoder Configuration.

Test Case ID: MEDIA2-4-2-4

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoEncoderConfigurationOptions, GetVideoEncoderConfigurations, SetVideoEncoderConfiguration

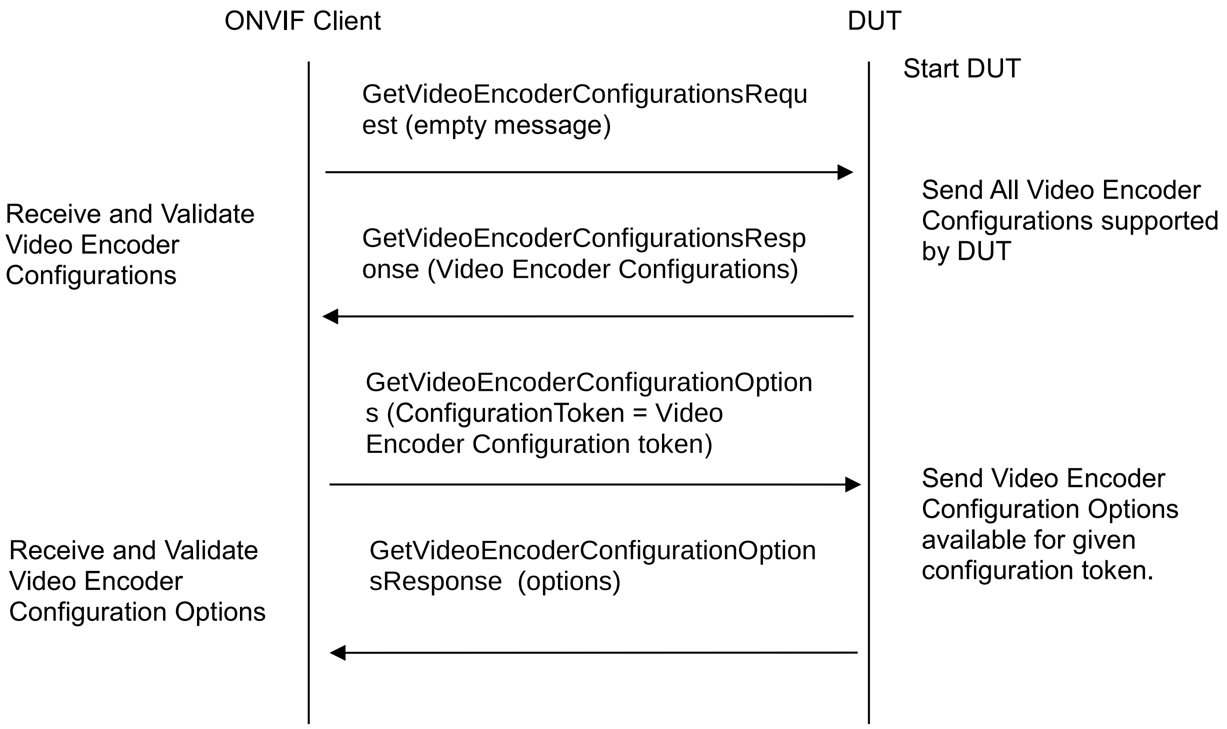
WSDL Reference: media2.wsdl

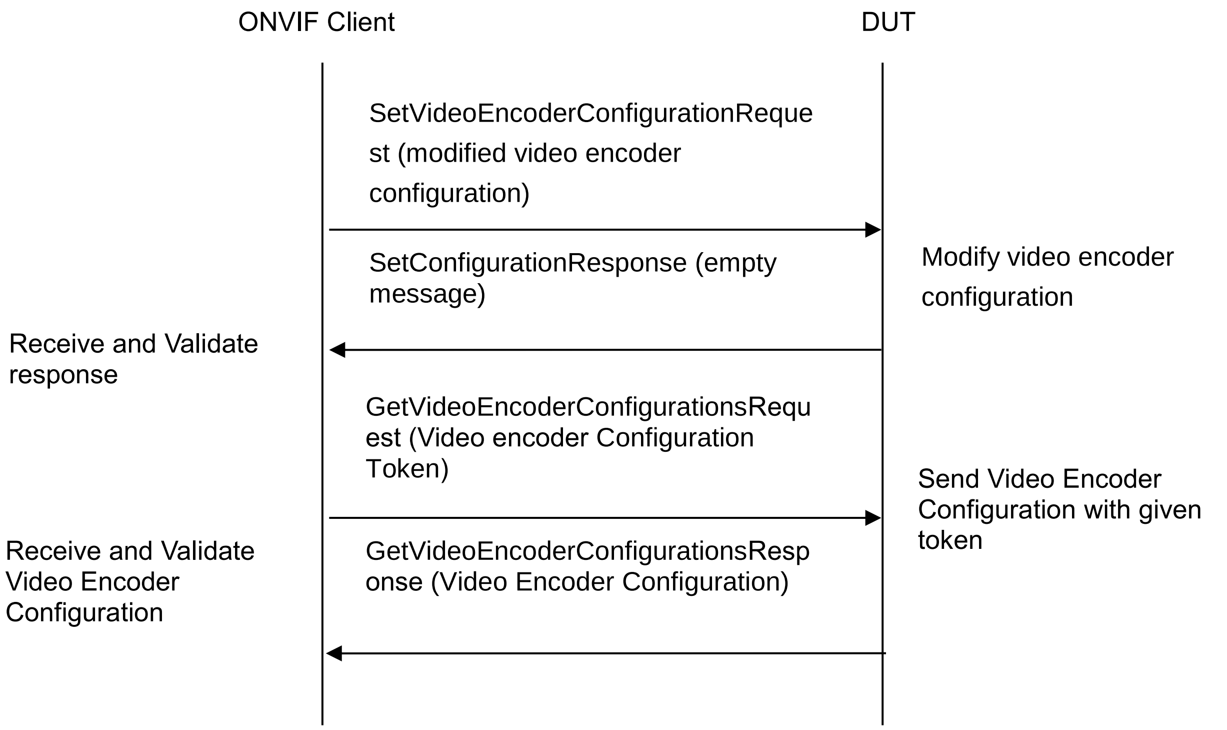
Test Purpose: To verify whether all supported options can be set.

Pre-Requisite: Media2 Service feature is supported by DUT. DUT has at least one video encoder configuration.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
GetVideoEncoderConfigurationsRequest (empty message)
Send All Video Encoder Configurations supported by DUT
Receive and Validate Video Encoder Configurations
GetVideoEncoderConfigurationsResponse (Video Encoder Configurations)
GetVideoEncoderConfigurationOptions (ConfigurationToken = Video Encoder Configuration token)
Send Video Encoder Configuration Options available for given configuration token.
Receive and Validate Video Encoder Configuration Options
GetVideoEncoderConfigurationOptionsResponse (options)


ONVIF Client
DUT
SetVideoEncoderConfigurationRequest (modified video encoder configuration)
Modify video encoder configuration
SetConfigurationResponse (empty message)
Receive and Validate response
GetVideoEncoderConfigurationsRequest (Video encoder Configuration Token)
Send Video Encoder Configuration with given token
GetVideoEncoderConfigurationsResponse (Video Encoder Configuration)
Receive and Validate Video Encoder Configuration

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetVideoEncoderConfigurations request to get all video encoder configurations from DUT.

4. DUT sends the list of video encoder configurations.

5. ONVIF Client verifies the list of video encoder configurations sent by DUT.

6. For each first video encoder configuration in GetVideoEncoderConfigurationsResponse, ONVIF Client saves this configuration in **Configuration1** variable and runs the following steps:

   6.1. ONVIF Client invokes GetVideoEncoderConfigurationOptionsRequest with ConfigurationToken = **Configuration1** token as input parameter to retrieve all configuration options available for the configuration.

   6.2. DUT sends GetVideoEncoderConfigurationOptionsResponse with list of options.

   6.3. For each option in GetVideoEncoderConfigurationOptionsResponse, ONVIF Client saves this option in **Option1** variable and runs the following steps 6.3.1 - 6.3.14:

      6.3.1. ONVIF Client changes the value of all parameters (including Encoding) in **Configuration1** to the available values indicated in **Option1** configuration options. For this step, if the value has a range of available values then minimum value (or first value from list if list of available values is provided by DUT) from **Option1** should be used. See details of fields mapping in Annex A.3.

6.3.2. ONVIF Client invokes SetVideoEncoderConfigurationRequest with **Configuration1** as input parameter.

6.3.3. DUT sends SetVideoEncoderConfigurationResponse message from the DUT.

6.3.4. ONVIF Client verifies SetVideoEncoderConfigurationResponse.

6.3.5. ONVIF Client invokes GetVideoEncoderConfigurationsRequest with **Configuration1** token as input parameter.

6.3.6. DUT sends GetVideoEncoderConfigurationsResponse with video encoder configuration for given token.

6.3.7. ONVIF Client compares the parameter's values of video encoder configuration from GetVideoEncoderConfigurationsResponse with the parameter's values of **Configuration1.** The parameter's values should be identical. If at list one pair of parameters is not identical then the test should FAIL.

6.3.8. ONVIF Client changes the value of all parameters (including Encoding) in **Configuration1** to the available values indicated in **Option1** configuration options. For this step, if the value has a range of available values then maximum value (or last value from list if list of available values is provided by DUT) from **Option1** should be used. See details of fields mapping in Annex A.3 VideoEncoderConfigurationOptions and VideoEncoderConfiguration mapping.

6.3.9. ONVIF Client invokes SetVideoEncoderConfigurationRequest with **Configuration1** as input parameter.

6.3.10. DUT sends SetVideoEncoderConfigurationResponse message from the DUT.

6.3.11. ONVIF Client verifies SetVideoEncoderConfigurationResponse.

6.3.12. ONVIF Client invokes GetVideoEncoderConfigurationsRequest with **Configuration1** token as input parameter.

6.3.13. DUT sends GetVideoEncoderConfigurationsResponse with video encoder configuration for given token.

6.3.14. ONVIF Client compares the parameter's values of video encoder configuration from GetVideoEncoderConfigurationsResponse with the parameter's values of **Configuration1.** The parameter's values should be identical. If at least one pair of parameters is not identical then the test should FAIL.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetVideoEncoderConfigurationsResponse message.

GetVideoEncoderConfigurationsResponse did not contain at least one Video Encoder Configuration.

Video encoder configuration parameters from GetVideoEncoderConfigurationResponse are not identical to the parameters set up in SetVideoEncoderConfigurationRequest message (see the list of parameters in Annex A.3).

DUT did not send valid GetVideoEncoderConfigurationOptionsResponse message.

DUT did not send valid SetVideoEncoderConfigurationResponse message.

DUT did not changed configuration after SetVideoEncoderConfigurationRequest.

Note: If any configuration is changed during this test, this configuration should be restored when the test is finished.

4.3 Audio Configuration

4.3.1 G.711 AUDIO ENCODER CONFIGURATION

Test Label: G.711 Audio Encoder Configuration

Test Case ID: MEDIA2-4-3-1

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoEncoderConfigurations, GetAudioEncoderConfigurationOptions, SetAudioEncoderConfiguration

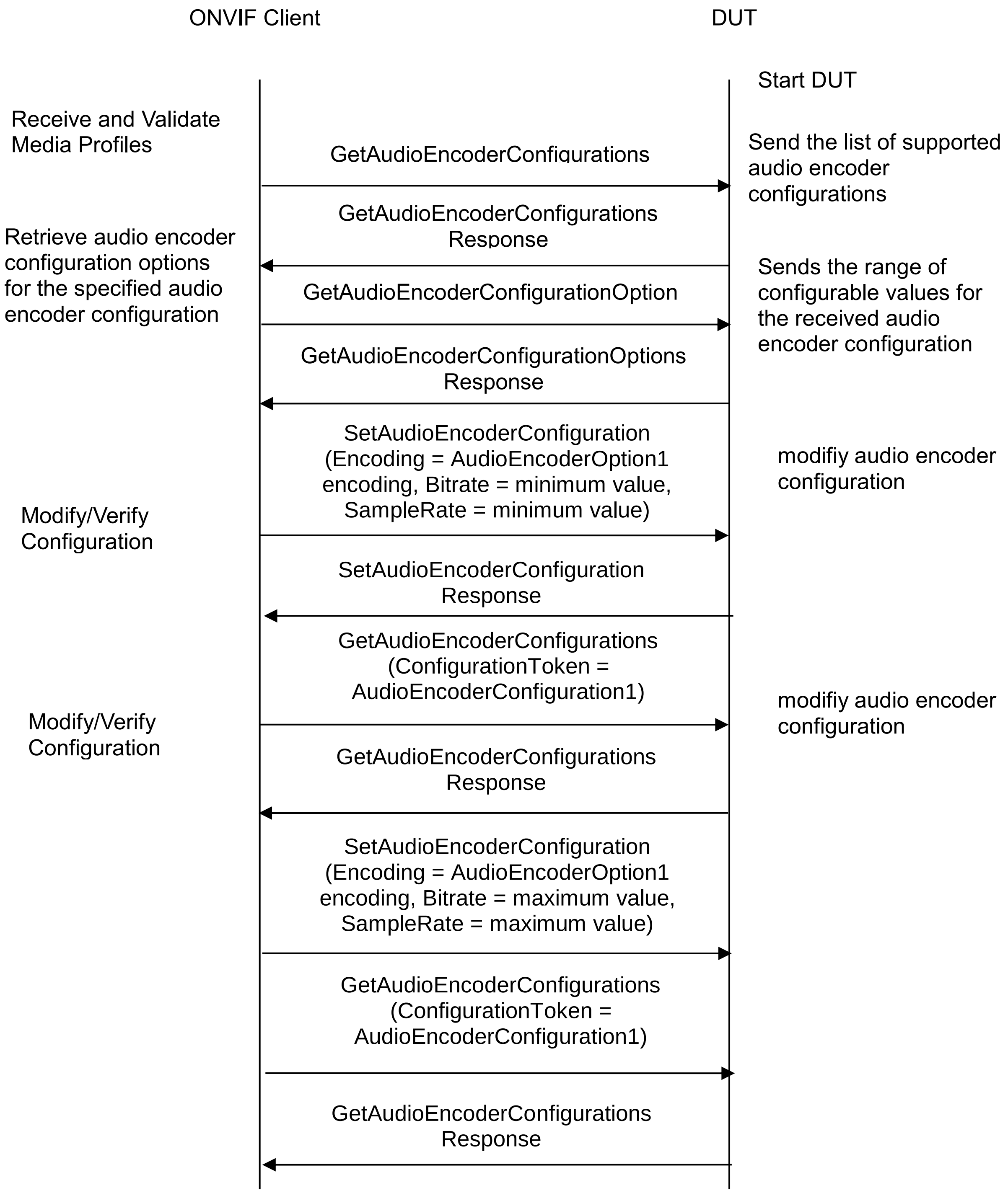
WSDL Reference: media2.wsdl

Test Purpose: To verify that DUT manages audio configuration with G.711 Encoding properly.

Pre-Requisite: Media2 Service feature is supported by DUT, Media2_G.711 feature is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
Receive and Validate Media Profiles
GetAudioEncoderConfigurations
Send the list of supported audio encoder configurations
GetAudioEncoderConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetAudioEncoderConfigurationOption
Sends the range of configurable values for the received audio encoder configuration
GetAudioEncoderConfigurationOptions Response
SetAudioEncoderConfiguration (Encoding = AudioEncoderOption1 encoding, Bitrate = minimum value, SampleRate = minimum value)
modifiy audio encoder configuration
Modify/Verify Configuration
SetAudioEncoderConfiguration Response
GetAudioEncoderConfigurations (ConfigurationToken = AudioEncoderConfiguration1)
modifiy audio encoder configuration
Modify/Verify Configuration
GetAudioEncoderConfigurations Response
SetAudioEncoderConfiguration (Encoding = AudioEncoderOption1 encoding, Bitrate = maximum value, SampleRate = maximum value)
GetAudioEncoderConfigurations (ConfigurationToken = AudioEncoderConfiguration1)
GetAudioEncoderConfigurations Response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetAudioEncoderConfigurations request with no arguments.

4. DUT sends the list of supported audio encoder configurations in GetAudioEncoderConfigurationsResponse message.

5. ONVIF Client invokes GetAudioEncoderConfigurationOptions Request (Audio Encoder Configuration token = configuration token from GetAudioEncoderConfigurationsResponse) 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. If GetAudioEncoderConfigurationOptionsResponse contains option with Encoding = “PCMU” (G.711), ONVIF Client saves this option in **AudioEncoderOption1** variable and the configuration in **AudioEncoderConfiguration1** variable and goes to the step 9.

8. If GetAudioEncoderConfigurationOptionsResponse does not contain option with Encoding = “PCMU” then ONVIF Client selects next configuration from GetAudioEncoderConfigurationsResponse and goes to the step 5.

9. ONVIF Client populates the following parameters of **AudioEncoderConfiguration1**: Encoding = **AudioEncoderOption1** encoding, Bitrate = minimum value in **AudioEncoderOption1** BitrateList, SampleRate = minimum value in **AudioEncoderOption1** SampleRateList.

10. ONVIF Client invokes SetAudioEncoderConfiguration request with Configuration = **AudioEncoderOption1** variable.

11. DUT modifies audio encoder configuration and responds with SetAudioEncoderConfigurationResponse message indicating success. ONVIF Client verifies SetAudioEncoderConfigurationResponse.

12. ONVIF Client invokes GetAudioEncoderConfigurations request with ConfigurationToken = **AudioEncoderConfiguration1** token.

13. DUT sends GetAudioEncoderConfigurationsResponse with audio encoder configuration for given token.

14. ONVIF Client verifies that GetAudioEncoderConfigurationsResponse contains only one audio encoder configuration with token = **AudioEncoderConfiguration1** token and Bitrate, SampleRate and Encoding values in this configuration equal to the values set in step 9.

15. ONVIF Client populates the following parameters of **AudioEncoderConfiguration1**: Encoding = **AudioEncoderOption1** encoding, Bitrate = maximum value in **AudioEncoderOption1** BitrateList, SampleRate = maximum value in **AudioEncoderOption1** SampleRateList.

16. ONVIF Client invokes SetAudioEncoderConfiguration request with Configuration = **AudioEncoderOption1** variable.

17. DUT modifies audio encoder configuration and responds with SetAudioEncoderConfigurationResponse message indicating success. ONVIF Client verifies SetAudioEncoderConfigurationResponse.

18. ONVIF Client invokes GetAudioEncoderConfigurations request with ConfigurationToken = **AudioEncoderConfiguration1** token.
19. DUT sends GetAudioEncoderConfigurationsResponse with audio encoder configuration for given token.
20. ONVIF Client verifies that GetAudioEncoderConfigurationsResponse contains only one audio encoder configuration with token = **AudioEncoderConfiguration1** token and Bitrate, SampleRate and Encoding values in this configuration equal to the values set in step 15

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetAudioEncoderConfigurationsResponse message.

GetAudioEncoderConfigurationsResponse did not contain at least one Audio Encoder Configuration.

Audio encoder configuration parameters from GetAudioEncoderConfigurationsResponse are not identical to the parameters set up in SetAudioEncoderConfigurationRequest message.

DUT did not send valid GetAudioEncoderConfigurationOptionsResponse message.

GetAudioEncoderConfigurationOptionsResponse did not contain at least one Audio Encoder Configuration Option.

DUT did not send valid SetAudioEncoderConfigurationResponse message.

DUT did not changed configuration after SetAudioEncoderConfigurationResponse.

4.3.2 AAC AUDIO ENCODER CONFIGURATION

Test Label: AAC Audio Encoder Configuration

Test Case ID: MEDIA2-4-3-2

ONVIF Core Specification Coverage: None

Command Under Test: GetProfiles, GetVideoEncoderConfigurations, GetAudioEncoderConfigurationOptions, SetAudioEncoderConfiguration

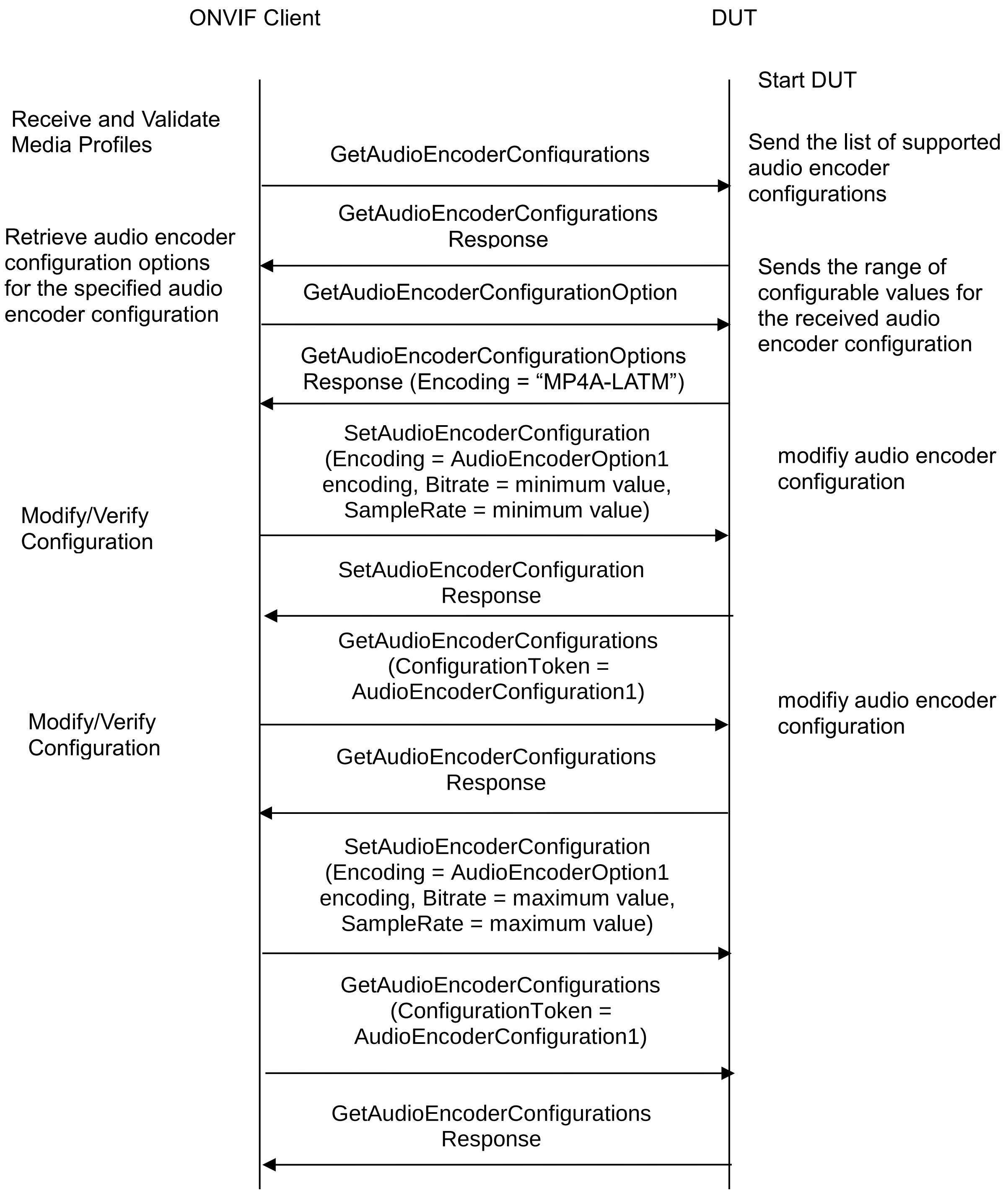
WSDL Reference: media2.wsdl

Test Purpose: To verify that DUT changes audio configuration with AAC Encoding properly.

Pre-Requisite: Media2 Service feature is supported by DUT, Media2_AAC feature is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
Receive and Validate Media Profiles
GetAudioEncoderConfigurations
Send the list of supported audio encoder configurations
GetAudioEncoderConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetAudioEncoderConfigurationOption
Sends the range of configurable values for the received audio encoder configuration
GetAudioEncoderConfigurationOptions Response (Encoding = “MP4A-LATM”)
SetAudioEncoderConfiguration (Encoding = AudioEncoderOption1 encoding, Bitrate = minimum value, SampleRate = minimum value)
modifiy audio encoder configuration
Modify/Verify Configuration
SetAudioEncoderConfiguration Response
GetAudioEncoderConfigurations (ConfigurationToken = AudioEncoderConfiguration1)
modifiy audio encoder configuration
Modify/Verify Configuration
GetAudioEncoderConfigurations Response
SetAudioEncoderConfiguration (Encoding = AudioEncoderOption1 encoding, Bitrate = maximum value, SampleRate = maximum value)
GetAudioEncoderConfigurations (ConfigurationToken = AudioEncoderConfiguration1)
GetAudioEncoderConfigurations Response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetAudioEncoderConfigurations request with no arguments.

4. DUT sends the list of supported audio encoder configurations in GetAudioEncoderConfigurationsResponse message.

5. ONVIF Client invokes GetAudioEncoderConfigurationOptions Request (Audio Encoder Configuration token = configuration token from GetAudioEncoderConfigurationsResponse) 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. If GetAudioEncoderConfigurationOptionsResponse contains option with Encoding = “MP4A-LATM” (AAC), ONVIF Client saves this option in **AudioEncoderOption1** variable and the configuration in **AudioEncoderConfiguration1** variable and goes to the step 9

8. If GetAudioEncoderConfigurationOptionsResponse does not contain option with Encoding = “MP4A-LATM” then ONVIF Client selects next configuration from GetAudioEncoderConfigurationsResponse and goes to the step 5.

9. ONVIF Client populates the following parameters of **AudioEncoderConfiguration1**: Encoding = **AudioEncoderOption1** encoding, Bitrate = minimum value in **AudioEncoderOption1** BitrateList, SampleRate = minimum value in **AudioEncoderOption1** SampleRateList.

10. ONVIF Client invokes SetAudioEncoderConfiguration request with Configuration = **AudioEncoderOption1** variable.

11. DUT modifies audio encoder configuration and responds with SetAudioEncoderConfigurationResponse message indicating success. ONVIF Client verifies SetAudioEncoderConfigurationResponse.

12. ONVIF Client invokes GetAudioEncoderConfigurations request with ConfigurationToken = **AudioEncoderConfiguration1** token.

13. DUT sends GetAudioEncoderConfigurationsResponse with audio encoder configuration for given token.

14. ONVIF Client verifies that GetAudioEncoderConfigurationsResponse contains only one audio encoder configuration with token = **AudioEncoderConfiguration1** token and Bitrate, SampleRate and Encoding values in this configuration equal to the values set in step 9.

15. ONVIF Client populates the following parameters of **AudioEncoderConfiguration1**: Encoding = **AudioEncoderOption1** encoding, Bitrate = maximum value in **AudioEncoderOption1** BitrateList, SampleRate = maximum value in **AudioEncoderOption1** SampleRateList.

16. ONVIF Client invokes SetAudioEncoderConfiguration request with Configuration = **AudioEncoderOption1** variable.

17. DUT modifies audio encoder configuration and responds with SetAudioEncoderConfigurationResponse message indicating success. ONVIF Client verifies SetAudioEncoderConfigurationResponse.

18. ONVIF Client invokes GetAudioEncoderConfigurations request with ConfigurationToken = **AudioEncoderConfiguration1** token.

19. DUT sends GetAudioEncoderConfigurationsResponse with audio encoder configuration for given token.

20. ONVIF Client verifies that GetAudioEncoderConfigurationsResponse contains only one audio encoder configuration with token = **AudioEncoderConfiguration1** token and Bitrate, SampleRate and Encoding values in this configuration equal to the values set in step 15.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

DUT did not send GetAudioEncoderConfigurationsResponse message.

GetAudioEncoderConfigurationsResponse did not contain at least one Audio Encoder Configuration.

Audio encoder configuration parameters from GetAudioEncoderConfigurationsResponse are not identical to the parameters set up in SetAudioEncoderConfigurationRequest message.

DUT did not send valid GetAudioEncoderConfigurationOptionsResponse message.

GetAudioEncoderConfigurationOptionsResponse did not contain at least one Audio Encoder Configuration Option.

DUT did not send valid SetAudioEncoderConfigurationResponse message.

DUT did not changed configuration after SetAudioEncoderConfigurationResponse.

4.4 OSD Configuration

4.4.1 CREATE OSD CONFIGURATION FOR TEXT OVERLAY

Test Label: Create OSD Configuration for Text Overlay

Test Case ID: MEDIA2-4-4-1

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoSourceConfigurations, GetOSDs, GetOSDOptions, CreateOSD, DeleteOSD

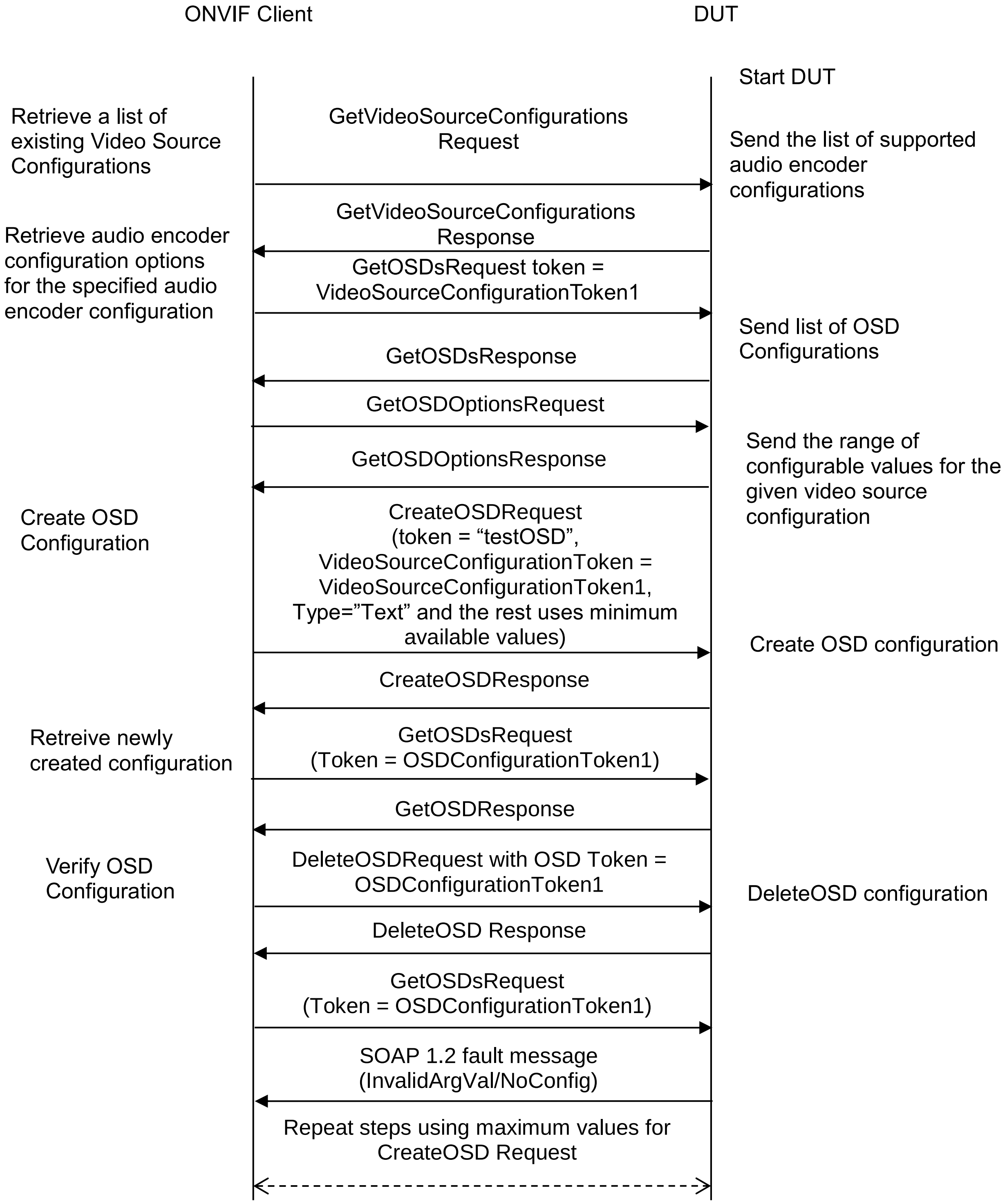
WSDL Reference: media2.wsdl

Test Purpose: To verify the DUT creates OSD Configuration.

Pre-Requisite: Media2 Service feature is supported by DUT, OSD feature is supported by DUT, The maximum number of OSD configuration is not reached.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
Retrieve a list of existing Video Source Configurations
GetVideoSourceConfigurations Request
Send the list of supported audio encoder configurations
GetVideoSourceConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetOSDsRequest token = VideoSourceConfigurationToken1
Send list of OSD Configurations
GetOSDsResponse
GetOSDOptionsRequest
Send the range of configurable values for the given video source configuration
GetOSDOptionsResponse
Create OSD Configuration
CreateOSDRequest (token = “testOSD”, VideoSourceConfigurationToken = VideoSourceConfigurationToken1, Type=”Text” and the rest uses minimum available values)
Create OSD configuration
CreateOSDResponse
Retreive newly created configuration
GetOSDsRequest (Token = OSDConfigurationToken1)
GetOSDResponse
Verify OSD Configuration
DeleteOSDRequest with OSD Token = OSDConfigurationToken1
DeleteOSD configuration
DeleteOSD Response
GetOSDsRequest (Token = OSDConfigurationToken1)
SOAP 1.2 fault message (InvalidArgVal/NoConfig)
Repeat steps using maximum values for CreateOSD Request

Test Procedure:

1 Start an ONVIF Client.

2 Start the DUT.

3 ONVIF Client invokes GetVideoSourceConfigurationsRequest message to retrieve a list of existing Video Source Configurations on the DUT.

4 Verify that the DUT returns at least one Video Source Configuration in the GetVideoSourceConfigurationsResponse message. For each Video Source Configuration from GetVideoSourceConfigurationsResponse, ONVIF Client saves the token of this configuration in **VideoSourceConfigurationToken1** variable and runs the steps 4.1 - 4.23:

4.1 ONVIF Client invokes GetOSDsRequest with configuration token = **VideoSourceConfigurationToken1**. DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client saves this list in **OSDConfigurationsList1** variable.

4.2 ONVIF Client invokes GetOSDOptionsRequest message with Configuration Token = **VideoSourceConfigurationToken1** to retrieve the OSD configuration options for the specified video source configuration.

4.3 DUT sends the range of configurable values for the given video source configuration in the GetOSDOptionsResponse message. ONVIF Client saves options from GetOSDOptionsResponse in **OSDOptions1** variable.

4.4 Onvif client checks if DUT supports OSD configurations of the type=text. Otherwise skip all steps and pass the test.

4.5 ONVIF Client invokes CreateOSDRequest with token = “testOSD” (note: this token can be ignored by DUT), VideoSourceConfigurationToken = **VideoSourceConfigurationToken1**, Type=”Text” and the rest of parameter are populated using values from **OSDOptions1**. For this step, if the value has a range of available values then minimum value (or first value from list if list of available values is provided by DUT) from **OSDOptions1** should be used. See details of fields mapping in Annex **A.4**.

4.6 DUT creates OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken1** variable.

4.7 ONVIF Client verifies that **OSDConfigurationsList1** does not contain configuration with token = **OSDConfigurationToken1**.

4.8 ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurationToken1** as input parameter to retrieve newly created OSD Configuration.

4.9 DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken1** and configuration fields equal to the fields set in step 4.5.

4.10 ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurationToken1** as input parameter.

4.11 DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

4.12 ONVIF Client invokes GetOSDsRequest with OSD Token = **OSDConfigurationToken1** as input parameter to verify that OSD Configuration has been deleted by DUT.

4.13 DUT send the SOAP 1.2 fault message (InvalidArgVal/NoConfig).

4.14 ONVIF Client verifies fault message.

4.15 ONVIF Client invokes CreateOSDRequest with token = “testOSD”, VideoSourceConfigurationToken = **VideoSourceConfigurationToken1**, Type=”Text” and the rest of parameters are populated using values from **OSDOptions1**. For this step, if the value has a range of available values then maximum value (or last value from list if list of available values is provided by DUT) from **OSDOptions1** should be used. See details of fields mapping in Annex **A.4**.

4.16 DUT created OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken2** variable.

4.17 ONVIF Client verifies that **OSDConfigurationsList1** does not contain configuration with token = **OSDConfigurationToken2**.

4.18 ONVIF Client invokes GetOSDsRequest message with configuration token = **OSDConfigurationsList1** as input parameter to retrieve the list of OSD Configurations which includes newly created OSD Configuration.

4.19 DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken2** and the fields of this configuration equal to the fields set in step 4.15.

4.20 ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurationToken2** as input parameter.

4.21 DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

4.22 ONVIF Client invokes GetOSDsRequest message with configuration token = **VideoSourceConfigurationToken1** as input parameter to retrieve the list of OSD Configurations which includes newly created OSD Configuration.

4.23 DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client verifies that response does not contain OSD Configuration with token = **OSDConfigurationToken2**.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetVideoSourceConfigurationsResponse message.

The GetVideoSourceConfigurationsResponse does not contain at least one Video Source configuration.

The DUT did not send GetOSDsResponse message.

The DUT did not send GetOSDOptionsResponse message.

The DUT did not send CreateOSDResponse message.

The DUT did not send GetOSDResponse message.

The DUT did not send SetConfigurationResponse message for DeleteOSD Request.

The DUT did not send SOAP 1.2 fault message (InvalidArgVal/NoConfig) for GetOSDsRequest with OSD Token of deleted configuration.

The GetOSDsResponse contains deleted OSD Configuration

4.4.2 CREATE OSD CONFIGURATION FOR IMAGE OVERLAY

Test Label: Create OSD Configuration for Image Overlay

Test Case ID: MEDIA2-4-4-2

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoSourceConfigurations, GetOSDs, GetOSDOptions, CreateOSD, DeleteOSD

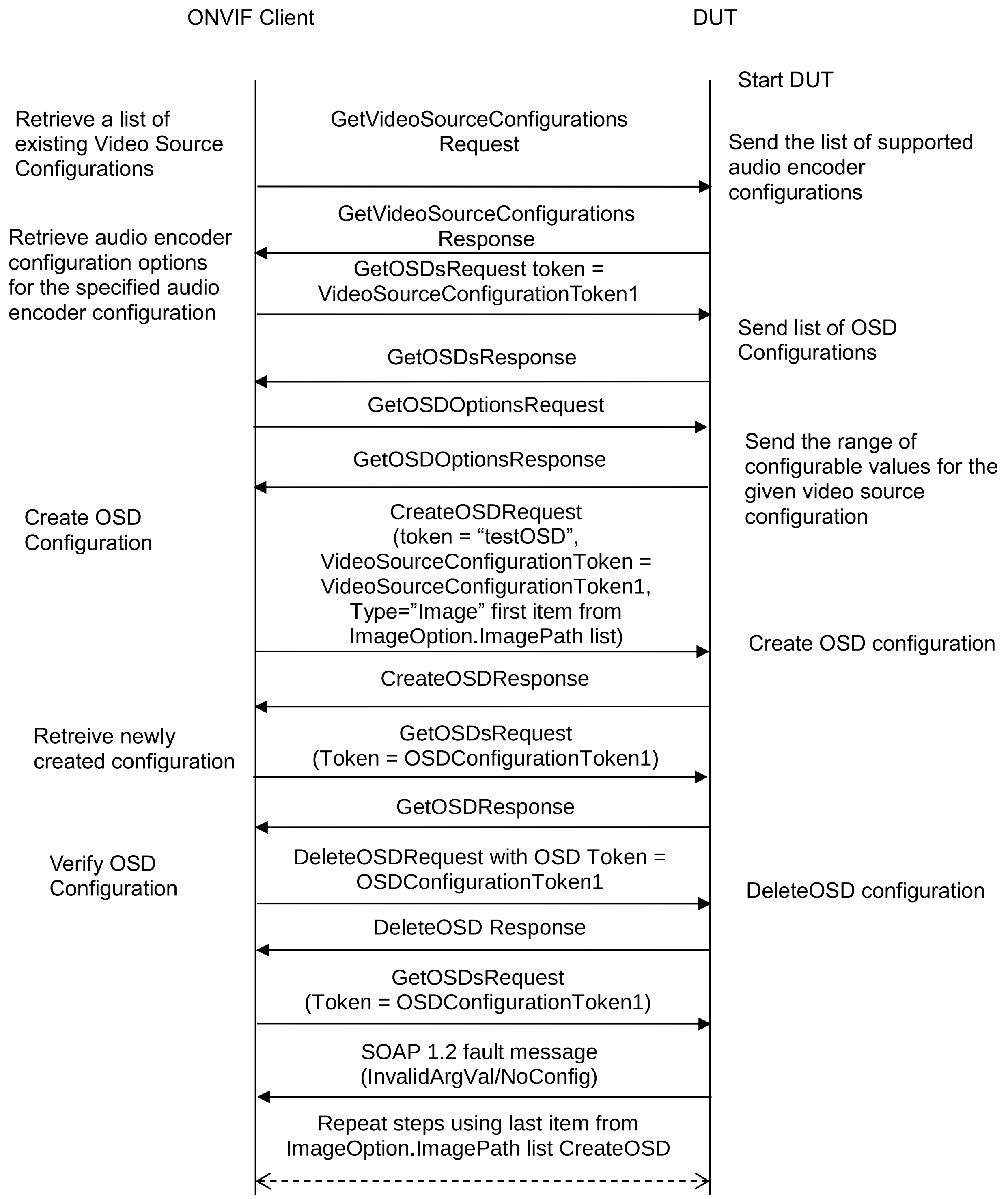
WSDL Reference: media2.wsdl

Test Purpose: To verify the DUT creates OSD Configuration.

Pre-Requisite: Media2 Service feature is supported by DUT, OSD feature is supported by DUT. The maximum number of OSD configuration is not reached. DUT returned at least one image overlay in GetOSDOptions.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
Retrieve a list of existing Video Source Configurations
GetVideoSourceConfigurations Request
Send the list of supported audio encoder configurations
GetVideoSourceConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetOSDsRequest token = VideoSourceConfigurationToken1
Send list of OSD Configurations
GetOSDsResponse
GetOSDOptionsRequest
Send the range of configurable values for the given video source configuration
GetOSDOptionsResponse
Create OSD Configuration
CreateOSDRequest (token = “testOSD”, VideoSourceConfigurationToken = VideoSourceConfigurationToken1, Type=”Image” first item from ImageOption.ImagePath list)
Create OSD configuration
CreateOSDResponse
Retreive newly created configuration
GetOSDsRequest (Token = OSDConfigurationToken1)
GetOSDResponse
Verify OSD Configuration
DeleteOSDRequest with OSD Token = OSDConfigurationToken1
DeleteOSD configuration
DeleteOSD Response
GetOSDsRequest (Token = OSDConfigurationToken1)
SOAP 1.2 fault message (InvalidArgVal/NoConfig)
Repeat steps using last item from ImageOption.ImagePath list CreateOSD

Test Procedure:

1 Start an ONVIF Client.

2 Start the DUT.

3 ONVIF Client invokes GetVideoSourceConfigurationsRequest message to retrieve a list of existing Video Source Configurations on the DUT.

4 Verify that the DUT returns at least one Video Source Configuration in the GetVideoSourceConfigurationsResponse message. For each Video Source Configuration from GetVideoSourceConfigurationsResponse, ONVIF Client saves the token of this configuration in **VideoSourceConfigurationToken1** variable and runs the steps 4.1 - 4.24:

4.1 ONVIF Client invokes GetOSDsRequest with configuration token = **VideoSourceConfigurationToken1**. DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client saves this list in **OSDConfigurationsList1** variable.

4.2 ONVIF Client invokes GetOSDOptionsRequest message with Configuration Token = **VideoSourceConfigurationToken1** to retrieve the OSD configuration options for the specified video source configuration.

4.3 DUT sends the range of configurable values for the given video source configuration in the GetOSDOptionsResponse message.

4.4 If GetOSDOptionsResponse contains at least one item in Image Option list (image overlay), then ONVIF Client saves the options from GetOSDOptionsResponse in **OSDOptions1** variable and goes to step 4.6.

4.5 If **Image Option** list is empty, then PASS the test.

4.6 ONVIF Client invokes CreateOSDRequest with token = "testOSD", VideoSourceConfigurationToken = **VideoSourceConfigurationToken1**, Type="Image", Position. Type should be set to the first item from **OSDOptions1.**PositionOption list and Image.ImgPath should be set to the first item from **OSDOptions1.**ImageOption.ImagePath list.

4.7 DUT creates OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken1** variable.

4.8 ONVIF Client verifies that **OSDConfigurationsList1** does not contain configuration with token = **OSDConfigurationToken1**.

4.9 ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurationToken1** as input parameter to retrieve newly created OSD Configuration.

4.10 DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken1** and Position.Type field equals to value has been set in 4.6.

4.11 ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurationToken1** as input parameter.

4.12 DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

4.13 ONVIF Client invokes GetOSDsRequest with OSD Token = **OSDConfigurationToken1** as input parameter to verify that OSD Configuration has been deleted by DUT.

4.14 DUT sends the SOAP 1.2 fault message (InvalidArgVal/NoConfig).

4.15 ONVIF Client verifies fault message.

4.16 ONVIF Client invokes CreateOSDRequest with token = "testOSD", VideoSourceConfigurationToken = **VideoSourceConfigurationToken1**, Type="Image", Position. Type should be set to the first item from **OSDOptions1.PositionOption** list and Image.ImgPath should be set to the last item from **OSDOptions1.**ImageOption.ImagePath list.

4.17 DUT created OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken2** variable.

4.18 ONVIF Client verifies that **OSDConfigurationsList1** does not contain configuration with token = **OSDConfigurationToken2**.

4.19 ONVIF Client invokes GetOSDsRequest message with configuration token = **OSDConfigurationToken2** as input parameter to retrieve the list of OSD Configurations which includes newly created OSD Configuration.

4.20 DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken2** and Position.Type field equals to value has been set in 4.16.

4.21 ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurationToken2** as input parameter.

4.22 DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

4.23 ONVIF Client invokes GetOSDsRequest message with configuration token = **VideoSourceConfigurationToken1** as input parameter to retrieve the list of OSD Configurations which includes newly created OSD Configuration.

4.24 DUT sends GetOSDsResponse with a list of OSD Configurations. ONVIF Client verifies that response does not contain OSD Configuration with token = **OSDConfigurationToken2**.

Test Result:

PASS –

DUT passes all assertions.

DUT did not send at least one image overlay (**Image Option** list is empty) in GetOSDOptionsResponse

FAIL –

The DUT did not send GetVideoSourceConfigurationsResponse message.

The GetVideoSourceConfigurationsResponse does not contain at least one Video Source configuration.

The DUT did not send GetOSDsResponse message.

The DUT did not send GetOSDOptionsResponse message.

The DUT did not send CreateOSDResponse message.

The DUT did not send GetOSDResponse message.

The DUT did not send SetConfigurationResponse message for DeleteOSD Request.

The DUT did not send SOAP 1.2 fault message (InvalidArgVal/NoConfig) for GetOSDsRequest with OSD Token of deleted configuration.

The GetOSDsResponse contains deleted OSD Configuration.

4.4.3 SET OSD CONFIGURATION IMAGE OVERLAY

Test Label: Set OSD Configuration Image Overlay

Test Case ID: MEDIA2-4-4-3

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoSourceConfigurations, GetOSDs, GetOSDOptions, CreateOSD, DeleteOSD, SetOSD

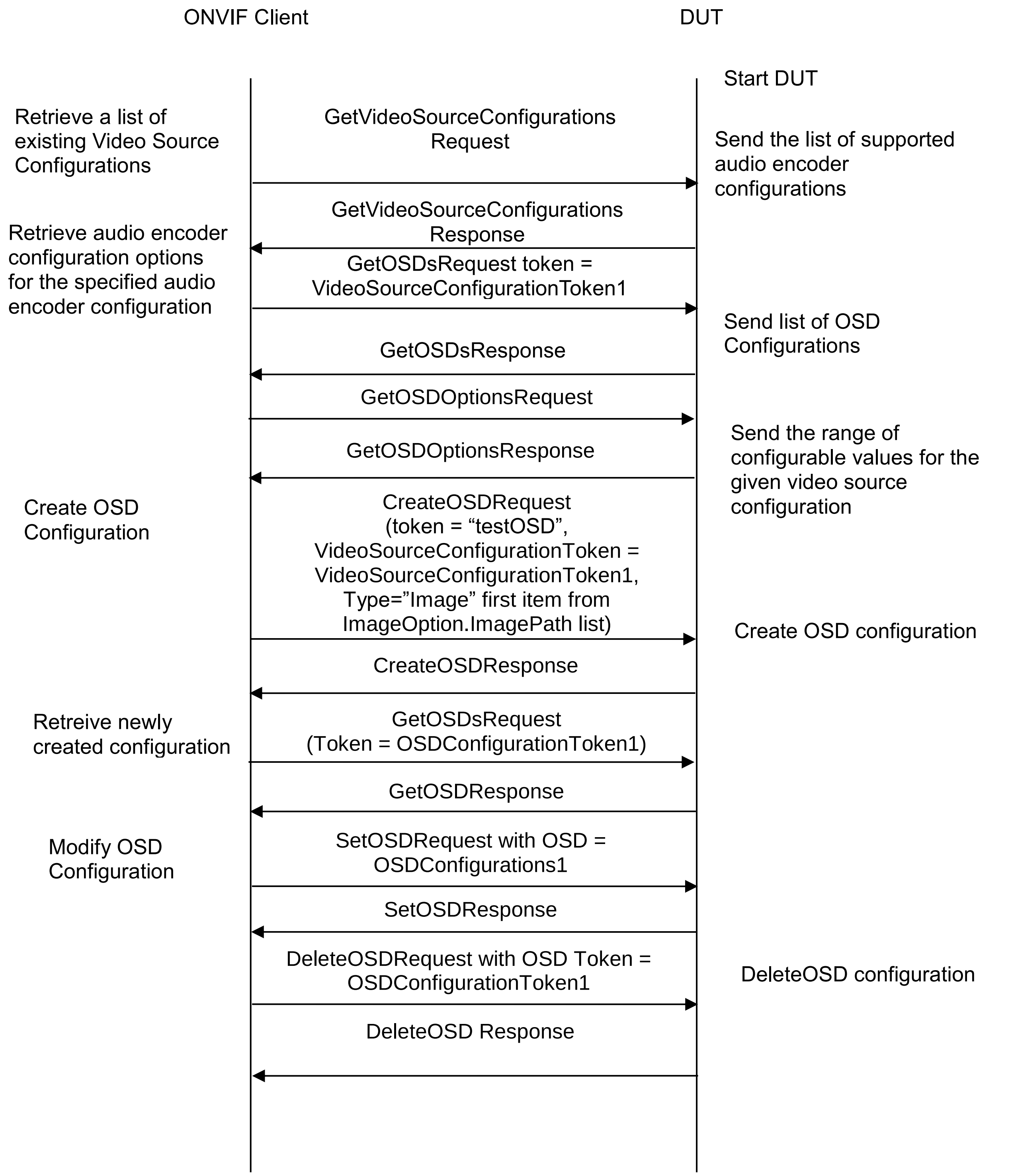
WSDL Reference: media2.wsdl

Test Purpose: To verify that DUT changes OSD Configuration for Image Overlay.

Pre-Requisite: Media2 Service feature is supported by DUT, OSD feature is supported by DUT, DUT returned at least one image overlay in GetOSDOptions.

Test Configuration: ONVIF Client and DUT

Test Sequence:


ONVIF Client
DUT
Start DUT
Retrieve a list of existing Video Source Configurations
GetVideoSourceConfigurations Request
Send the list of supported audio encoder configurations
GetVideoSourceConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetOSDsRequest token = VideoSourceConfigurationToken1
Send list of OSD Configurations
GetOSDsResponse
GetOSDOptionsRequest
GetOSDOptionsResponse
Send the range of configurable values for the given video source configuration
Create OSD Configuration
CreateOSDRequest (token = “testOSD”, VideoSourceConfigurationToken = VideoSourceConfigurationToken1, Type=”Image” first item from ImageOption.ImagePath list)
Create OSD configuration
CreateOSDResponse
Retreive newly created configuration
GetOSDsRequest (Token = OSDConfigurationToken1)
GetOSDResponse
Modify OSD Configuration
SetOSDRequest with OSD = OSDConfigurations1
SetOSDResponse
DeleteOSDRequest with OSD Token = OSDConfigurationToken1
DeleteOSD configuration
DeleteOSD Response

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client invokes GetVideoSourceConfigurationsRequest message to retrieve a list of existing Video Source Configurations on the DUT.

4. Verify that the DUT returns at least one Video Source Configuration in the GetVideoSourceConfigurationsResponse message. For each Video Source Configuration from GetVideoSourceConfigurationsResponse, ONVIF Client saves this configuration in **VideoSourceConfigurationToken1** variable and runs the following steps:

    4.1. ONVIF Client invokes GetOSDOptionsRequest message with Configuration Token = **VideoSourceConfigurationToken1** to retrieve the OSD configuration options for the specified video source configuration.

    4.2. DUT sends the range of configurable values for the given video source configuration in the GetOSDOptionsResponse message.

    4.3. If GetOSDOptionsResponse contains at least one item in **Image Option** (image overlay), then ONVIF Client saves the options from GetOSDOptionsResponse in **OSDOptions1** variable and goes to step 4.5.

    4.4. If **OSDOptions1** Image Option list is empty, then PASS the test.

    4.5. ONVIF Client invokes GetOSDsRequest with configuration token = **VideoSourceConfigurationToken1**. DUT sends GetOSDsResponse with a list of OSD Configurations.

    4.6. ONVIF Client invokes CreateOSDRequest with token = “testOSD”, VideoSourceConfigurationToken = **VideoSourceConfigurationToken1**, Type=”Image”, Position.Type should be set to the first item from **OSDOptions1.**PositionOption list and Image.ImgPath should be set to the first item from **OSDOptions1.**ImageOption.ImagePath list.

    4.7. DUT creates OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken1** variable.

    4.8. ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurationToken1** as input parameter to retrieve newly created OSD Configuration.

    4.9. DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken1** and saves this configuration in **OSDConfigurations1** variable.

    4.10. ONVIF Client changes position parameter in **OSDConfigurations1** variable. The Position.Type should be changed to the last item from **OSDOptions1.**PositionOption list.

    4.11. ONVIF Client invokes SetOSDRequest with OSD = **OSDConfigurations1** as input parameter. DUT applies the changes and sends SetConfigurationResponse.

    4.12. ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurations1** token as input parameter.

    4.13. DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurations1** token and Position.Type is set to the value which has been set in step 4.11.

    4.14. If new OSD Configuration has been created on step 4.5 then ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurations1** token as input parameter.

    4.15. DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

Test Result:

PASS –

DUT passes all assertions.

DUT did not send at least one image overlay (**Image Option** list is empty) in GetOSDOptionsResponse

FAIL –

The DUT did not send GetVideoSourceConfigurationsResponse message.

The GetVideoSourceConfigurationsResponse does not contain at least one Video Source configuration.

The DUT did not send GetOSDsResponse message.

The DUT did not send GetOSDOptionsResponse message.

The DUT did not send CreateOSDResponse message.

The DUT rejected create OSD Request.

The DUT did not send GetOSDResponse message.

The DUT did not send SetConfigurationResponse message for SetOSD Request.

The DUT did not send SetConfigurationResponse message for DeleteOSD Request.

4.4.4 SET OSD CONFIGURATION TEXT OVERLAY

Test Label: Set OSD Configuration Text Overlay

Test Case ID: MEDIA2-4-4-4

ONVIF Core Specification Coverage: None

Command Under Test: GetVideoSourceConfigurations, GetOSDs, GetOSDOptions, CreateOSD, DeleteOSD, SetOSD

WSDL Reference: media2.wsdl

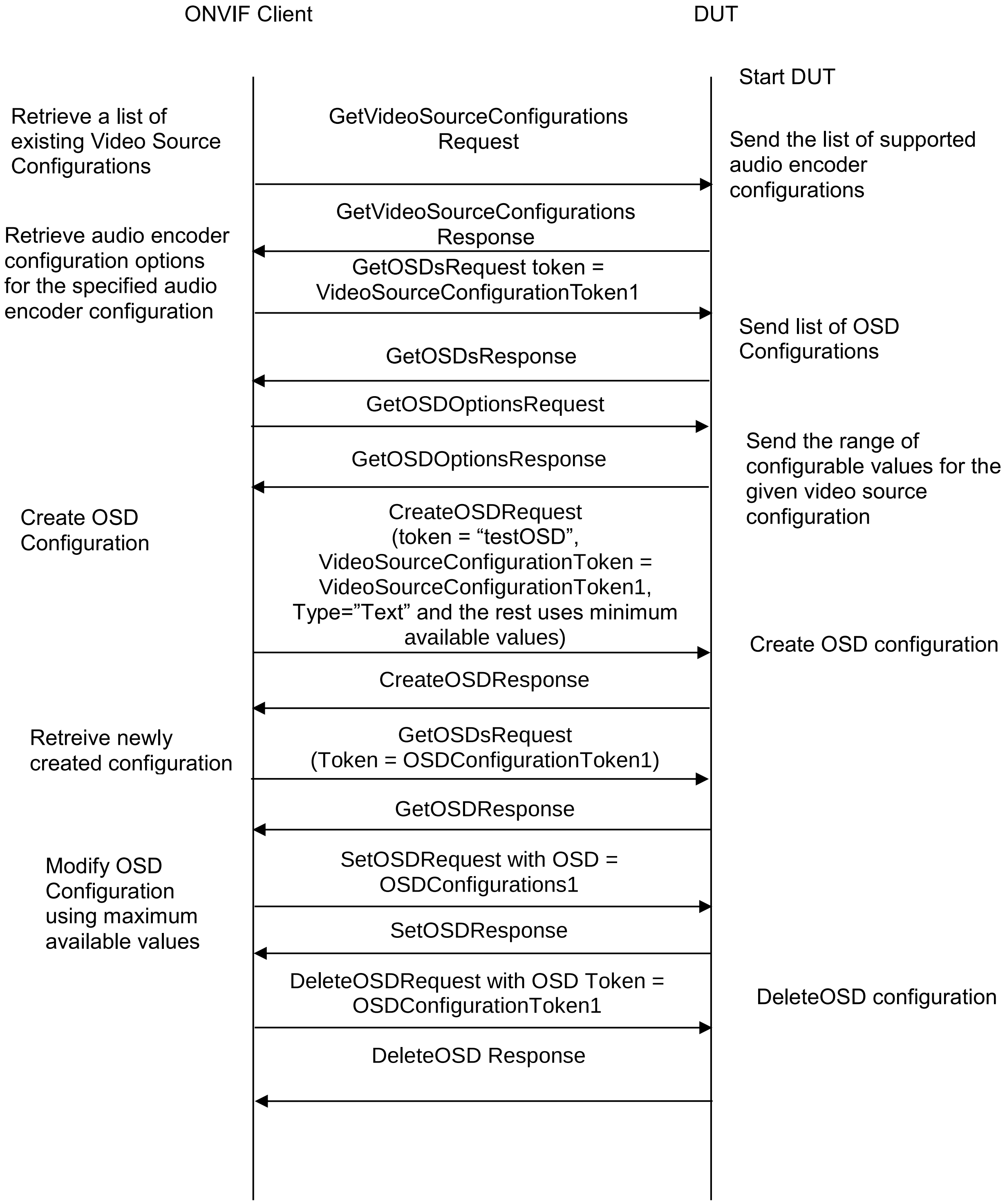
Test Purpose: To verify that DUT changes OSD Configuration for Test Overlay.

Pre-Requisite: Media2 Service feature is supported by DUT, OSD feature is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Sequence:

Driving IP-based physical security through global standardization


ONVIF Client
DUT
Start DUT
Retrieve a list of existing Video Source Configurations
GetVideoSourceConfigurations Request
Send the list of supported audio encoder configurations
GetVideoSourceConfigurations Response
Retrieve audio encoder configuration options for the specified audio encoder configuration
GetOSDsRequest token = VideoSourceConfigurationToken1
Send list of OSD Configurations
GetOSDsResponse
GetOSDOptionsRequest
Send the range of configurable values for the given video source configuration
GetOSDOptionsResponse
Create OSD Configuration
CreateOSDRequest (token = “testOSD”, VideoSourceConfigurationToken = VideoSourceConfigurationToken1, Type=”Text” and the rest uses minimum available values)
Create OSD configuration
CreateOSDResponse
Retreive newly created configuration
GetOSDsRequest (Token = OSDConfigurationToken1)
GetOSDResponse
Modify OSD Configuration using maximum available values
SetOSDRequest with OSD = OSDConfigurations1
SetOSDResponse
DeleteOSDRequest with OSD Token = OSDConfigurationToken1
DeleteOSD configuration
DeleteOSD Response

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetVideoSourceConfigurationsRequest message to retrieve a list of existing Video Source Configurations on the DUT.

4. Verify that the DUT returns at least one Video Source Configuration in the GetVideoSourceConfigurationsResponse message. For each Video Source Configuration from GetVideoSourceConfigurationsResponse, ONVIF Client saves this configuration in **VideoSourceConfigurationToken1** variable and runs the steps 4.1 - 4.13

   4.1. ONVIF Client invokes GetOSDOptionsRequest message with Configuration Token = **VideoSourceConfigurationToken1** to retrieve the OSD configuration options for the specified video source configuration.

   4.2. DUT sends the range of configurable values for the given video source configuration in the GetOSDOptionsResponse message. ONVIF Client saves options from GetOSDOptionsResponse in **OSDOptions1** variable.

   4.3. ONVIF Client invokes GetOSDsRequest with configuration token = **VideoSourceConfigurationToken1**. DUT sends GetOSDsResponse with a list of OSD Configurations.

   4.4. Onvif client checks if DUT supports OSD configurations of the type=text. Otherwise skip all steps and pass the test.

   4.5. ONVIF Client invokes CreateOSDRequest with token = “testOSD”, VideoSourceConfigurationToken = **VideoSourceConfigurationToken1,** Type=”Text” and the rest of parameters are populated using values from **OSDOptions1**. For this step, if the value has a range of available values then minimum value (or first value from list if list of available values is provided by DUT) from **OSDOptions1** should be used. See details of fields mapping in Annex **A.4**.

   4.6. DUT creates OSD Configuration and sends CreateOSDResponse message. ONVIF Client receives CreateOSDResponse with token of newly created OSD Configuration and saves this token to **OSDConfigurationToken1** variable.

   4.7. ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurationToken1** as input parameter to retrieve newly created OSD Configuration.

   4.8. DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurationToken1** and saves this configuration in **OSDConfigurations1** variable.

   4.9. ONVIF Client changes the fields value of **OSDConfigurations1** variable to the maximum available values. If the value has a range of available values, then maximum value (or last value from list if list of available values is provided by DUT) from **OSDOptions1** should be used. See details of fields mapping in Annex **A.4**.

   4.10. ONVIF Client invokes SetOSDRequest with OSD = **OSDConfigurations1** as input parameter. DUT applies the changes and sends SetConfigurationResponse.

   4.11. ONVIF Client invokes GetOSDsRequest message with OSD Token = **OSDConfigurations1** token as input parameter.

   4.12. DUT sends OSD Configuration for the given token. ONVIF Client verifies that response contains OSD Configuration with token = **OSDConfigurations1** token and configuration fields equal to the fields set in step 4.9.

   4.13. If new OSD Configuration has been created on step 4.5 then ONVIF Client invokes DeleteOSDRequest with OSD Token = **OSDConfigurations1** token as input parameter.

   4.14. DUT sends SetConfigurationResponse which indicates that DUT has deleted OSD Configuration. ONVIF Client verifies the response.

Test Result:

PASS –

DUT passes all assertions.

FAIL –

The DUT did not send GetVideoSourceConfigurationsResponse message.

The GetVideoSourceConfigurationsResponse does not contain at least one Video Source configuration.

The DUT did not send GetOSDsResponse message.

The DUT did not send GetOSDOptionsResponse message.

The DUT did not send CreateOSDResponse message.

The DUT rejected create OSD Request.

The DUT did not send GetOSDResponse message.

The DUT did not send SetConfigurationResponse message for SetOSD Request.

The DUT did not send SetConfigurationResponse message for DeleteOSD Request.

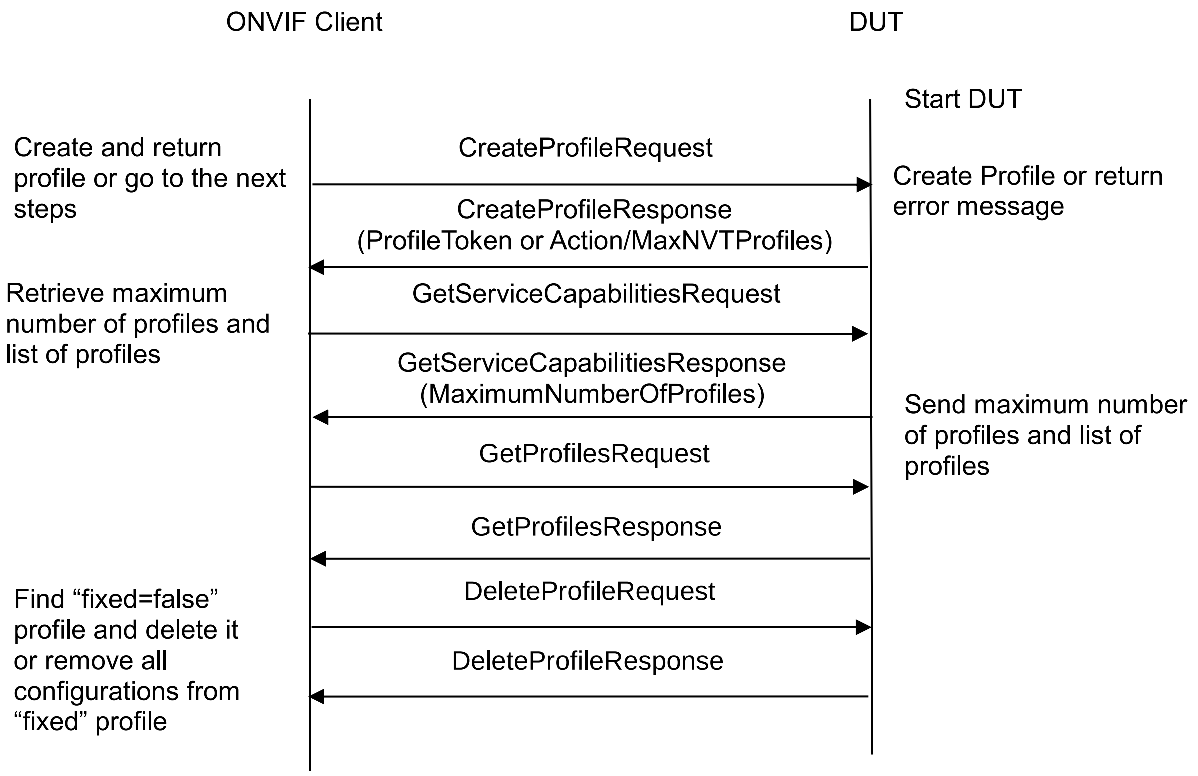
Annex A

This section describes the meaning of the following definitions. These definitions are used in the test case description.

A.1 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 (without configuration).


ONVIF Client
DUT
Start DUT
Create and return profile or go to the next steps
CreateProfileRequest
Create Profile or return error message
CreateProfileResponse (ProfileToken or Action/MaxNVTProfiles)
Retrieve maximum number of profiles and list of profiles
GetServiceCapabilitiesRequest
GetServiceCapabilitiesResponse (MaximumNumberOfProfiles)
Send maximum number of profiles and list of profiles
GetProfilesRequest
GetProfilesResponse
Find "fixed=false" profile and delete it or remove all configurations from "fixed" profile
DeleteProfileRequest
DeleteProfileResponse

1. ONVIF Client invokes CreateProfileRequest. If ONVIF Client receives CreateProfileResponse response, it gets profile token from CreateProfileResponse and returns the profile token to the test which executed this Annex and skip other tests. If DUT responses with Action/MaxNVTProfiles then go to the step 2.

2. ONVIF Client invokes GetServiceCapabilitiesRequest message to retrieve maximum number of profiles (**MaximumNumberOfProfiles**) supported by DUT.

3. ONVIF Client will invoke GetProfilesRequest message to retrieve complete profiles list. Verify the GetProfilesResponse message from the DUT.

4. If number of media profiles in GetProfilesResponse is greater or equals to **MaximumNumberOfProfiles** then go to the step 5. Otherwise, if number of media profiles in GetProfilesResponse is less than **MaximumNumberOfProfiles** then the test should FAIL.

5. If GetProfilesResponse contains profiles with fixed attribute equal to false, then ONVIF Client will invoke DeleteProfileRequest message (Profile Token = Token1, where Token1 is the first Profile Token from the GetProfilesResponse with fixed = false) to delete Profile. Verify the DeleteProfileResponse message from the DUT. Go to step 1.

6. 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.

Note: See Annex A.2 for Name and Token Parameters Length limitations.

A.2 Name Parameters

There are following limitations on maximum length of the Name parameters that sell be used during tests by ONVIF Device Test Tool to prevent faults from DUT:

- Name shall be less than or equal to 64 characters (only readable characters accepted).
- UTF-8 character set shall be used for Name.

Note: these limitations will not be used, if ONVIF Device Test Tool reuses values that were received from the DUT.

A.3 VideoEncoderConfigurationOptions and VideoEncoderConfiguration mapping

VideoEncoderConfigurationOptions field	VideoEncoderConfiguration field
Encoding	Encoding
QualityRange.Min QualityRange Max	Quality
Resolutions Available contains list of Height and Width pairs.	Height and Width
GovLengthRange Min GovLengthRange Max	GovLength
FrameRatesSupported contains list of available values.	RateControl.FrameRateLimit
BitrateRange Min BitrateRange Max	RateControl.BitrateLimit
ProfilesSupported the list of sting values listed in tt:VideoEncodingProfiles	Profile

A.4 OSDConfigurationOptions and OSDConfiguration mapping

OSDConfigurationOptions fields	OSDConfiguration field
Type contains a list of available values	Type
PositionOptions contains a list of available values	Position.Type
TextOption.Type contains a list of available values	TextString.Type
TextOption.FontSizeRange Min TextOption. FontSizeRange Max	TextString.FontSize
TextOption.DateFormat contains a list of available values	TextString.DateFormat
TextOption.TimeFormat contains a list of available values	TextString.TimeFormat
TextOption.FontColor.Color.ColorList contains a list of available color sets (X, Y, Z, Colorspace)	TextString.FontColor.Color (X, Y, Z, Colorspace)
TextOption.FontColor.Color.ColorRange.X Min TextOption.FontColor.Color.ColorRange.X Max	TextString.FontColor.Color.X
TextOption.FontColor.Color.ColorRange.Y Min TextOption.FontColor.Color.ColorRange.Y Man	TextString.FontColor.Color.Y
TextOption.FontColor.Color.ColorRange.Z Min TextOption.FontColor.Color.ColorRange.Z Max	TextString.FontColor.Color.Y

TextOption.FontColor.Color.ColorRange.Colorspace	TextString.FontColor.Color.Colorspace
TextOption.FontColor.Transparent Min TextOption.FontColor.Transparent Max	TextString.FontColor.Transparent
TextOption.BackgroundColor.Color.ColorList contains a list of available color sets (X, Y, Z, Colorspace)	TextString.BackgroundColor.Color (X, Y, Z, Colorspace)
TextOption.BackgroundColor.Color.ColorRange.X Min TextOption.BackgroundColor.Color.ColorRange.X Max	TextString.BackgroundColor.Color.X
TextOption.BackgroundColor.Color.ColorRange.Y Min TextOption.BackgroundColor.Color.ColorRange.Y Man	TextString.BackgroundColor.Color.Y
TextOption.BackgroundColor.Color.ColorRange.Z Min TextOption.BackgroundColor.Color.ColorRange.Z Max	TextString.BackgroundColor.Color.Z
TextOption.BackgroundColor.Color.ColorRange.Colorspace	TextString.BackgroundColor.Color. Colorspace
TextOption.FontColor.Transparent Min TextOption.FontColor.Transparent Max	TextString.BackgroundColor.Transparent