ONVIF^{TM}

Real Time Streaming using Media2 Device Test Specification

Version 17.01

January 2017

REVISION HISTORY

Vers.	Date	Description
16.06	Apr, 2016	Original publication
16.07	Jun, 2016	Small changes based on feedback received.
16.07	Jul 8, 2016	Added G.711 and AAC RTP-Multicast tests for IPv4 and IPv6 (4.2.8 , 4.2.9 , 4.2.15 , 4.2.16) Section 4.5 (Sart and Stop Mulicast streaming) deleted
16.07	Jul 28, 2016	Review comments implemented.
16.07	Aug 8, 2016	More comments and spelling errors.
17.01	Sep, 2016	Added the test cases for H.264, H.265, G.711, and AAC streaming over HTTPS: MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSPS/HTTP/TCP, IPv6) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSPS/HTTP/TCP, IPv6) MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)
17.01	Nov, 2016	Added the test cases for H.265: MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP) MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP) MEDIA2 SET SYNCHRONIZATION POINT – H.265 MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP, IPv6) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP, IPv6) MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv4) MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv6)
17.01	Nov, 2016	Test IDs were updated according #1253.
17.01	Jan 19, 2017	Test pecification was converted to new format.

		HTTPS test cases were updated according comments to ticket #1168: MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSPS/HTTPS/TCP, IPv6) MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSPS/HTTPS/TCP, IPv6) MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP) MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Table of Contents

1 Introduction 9

1.1 Scope 9

1.2 Real Time Streaming 10

2 Normative references 11

3 Terms and Definitions 12

3.1 Conventions 12

3.2 Definitions 12

3.3 Abbreviations 12

4 Test Overview 13

4.1 Test Setup 13

4.1.1 Network Configuration for DUT 13

4.2 Prerequisites 14

4.3 Test Policy 14

4.3.1 Media Configuration 14

5 Real Time Streaming Test Cases 16

5.1 Video Streaming 16

5.1.1 Unicast 16

5.1.1.1 MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP) 16

5.1.1.2 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP) 18

5.1.1.3 MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP) 20

5.1.1.4 MEDIA2 SET SYNCHRONIZATION POINT – H.264 22

5.1.1.5 MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP, IPv6) 25

5.1.1.6 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) 28

5.1.1.7 MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP, IPv6) 30

5.1.1.8 MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP) 33

5.1.1.9 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP) 35

5.1.1.10 MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP) 37

5.1.1.11 MEDIA2 SET SYNCHRONIZATION POINT – H.265 39

5.1.1.12 MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP, IPv6) 42

5.1.1.13 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) 45
5.1.1.14 MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP, IPv6) 47
5.1.1.15 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP) 50
5.1.1.16 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP) 53
5.1.1.17 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) 55
5.1.1.18 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) 58
5.1.2 Multicast 61
5.1.2.1 MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv4) 61
5.1.2.2 MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv6) 64
5.1.2.3 MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv4) 66
5.1.2.4 MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv6) 69
5.2 Audio Streaming 72
5.2.1 Unicast 72
5.2.1.1 MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP) 72
5.2.1.2 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP) 74
5.2.1.3 MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP) 75
5.2.1.4 MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP, IPv6) 77
5.2.1.5 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) 79
5.2.1.6 MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP, IPv6) 82
5.2.1.7 MEDIA2 STREAMING – AAC (RTP-Unicast/UDP) 84
5.2.1.8 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP) 86
5.2.1.9 MEDIA2 STREAMING – AAC (RTP/RTSP/TCP) 87
5.2.1.10 MEDIA2 STREAMING – AAC (RTP-Unicast/UDP, IPv6) 89
5.2.1.11 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) 91

5.2.1.12 MEDIA2 STREAMING – AAC (RTP/RTSP/TCP, IPv6) 94
5.2.1.13 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP) 96
5.2.1.14 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP) 98
5.2.1.15 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) 101
5.2.1.16 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) 103
5.2.2 Multicast 106
5.2.2.1 MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv4) 106
5.2.2.2 MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv6) 108
5.2.2.3 MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4) 111
5.2.2.4 MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6) 113
A Helper Procedures and Additional Notes 116
A.1 Invalid RTP Header 116
A.2 I-frame insertion time interval 116
A.3 Name and Token Parameters 116
A.4 Turn on IPv6 network interface 116
A.5 Restore Network Settings 118
A.6 Media2 Service Profile Configuration for Video Streaming 119
A.7 Media2 Service – Media Profile Configuration for Audio Streaming 120
A.8 Removing Video Encoder Configuration and Metadata Configuration from Media Profile 123
A.9 Removing Audio Encoder Configuration and Metadata Configuration from Media Profile 125
A.10 Media Streaming over RTP-Unicast/UDP 126
A.11 Media Streaming over RTP-Unicast/RTSP/HTTP/TCP 127
A.12 Media Streaming over RTP/RTSP/TCP 129
A.13 Media Streaming over RTP-Multicast 130
A.14 Media Streaming over RTP-Unicast/RTSP/HTTPS/TCP 132
A.15 Configuring HTTPS using Advanced Security 134

A.16 Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair 135
A.17 Add server certificate assignment with corresponding certification path, CA certificate and RSA key pair 137
A.18 Create an RSA key pair 139
A.19 Subject for a server certificate 140

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. In addition, 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 Real Time Streaming using Media2 Device Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases need to be executed and passed. And 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 Real Time Streaming using Media2 Device 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 and Realtime Streaming Specification, 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 Media2 Service and Realtime Streaming and some of the optional requirements.

This ONVIF Real Time Streaming using Media2 Device Test Specification covers ONVIF Media2 Service and Realtime Streaming, 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 Real Time Streaming

Real Time Streaming using Media2 covers the test cases needed for the verification of real time streaming features using Media2 Service as mentioned in [ONVIF Network Interface Specs]. Real time streaming defines different media streaming options based on RTP for video, audio and metadata streams. Media control is done using RTSP protocol.

The scope of this specification covers the following real time streaming options for H.264 and H.265 video streams, and G.711, AAC Audio streams.

- RTSP control requests
- RTP Unicast over UDP
- RTP over RTSP over TCP
- RTP over RTSP over HTTP over TCP
- RTCP

2 Normative references

- ONVIF Conformance Process Specification:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Profile Policy:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Core Specifications:

  http://www.onvif.org/Documents/Specifications.aspx

- ONVIF Base Test Specification:

  http://www.onvif.org/Portals/0/documents/testspecs/v16_07/ONVIF_Base_Test_Specification_16.07.pdf

- ONVIF Media 2 Service Specification:

  http://www.onvif.org/specs/srv/media/ONVIF-Media2-Service-Spec-v1606.pdf

- ONVIF Streaming Specification :

  http://www.onvif.org/specs/stream/ONVIF-Streaming-Spec-v1612.pdf

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

  http://www.iso.org/directives

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

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

- W3C SOAP 1.2, Part 1, Messaging Framework:

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

- W3C XML Schema Part 1: Structures Second Edition:

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

- W3C XML Schema Part 2: Datatypes Second Edition:

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

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 describes terms and definitions used in this document.

Profile	See ONVIF Profile Policy.
ONVIF Device	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
ONVIF Client	Computer appliance or software program that uses ONVIF Web Services.
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.
SOAP	SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.
Device Test Tool	ONVIF Device Test Tool that tests ONVIF Device implementation towards the ONVIF Test Specification set.
Media 2 Service	Services to determine the streaming properties of requested media streams.

3.3 Abbreviations

This section describes abbreviations used in this document.

HTTP	Hyper Text Transport Protocol.
AAC	Advanced Audio Coding.
URI	Uniform Resource Identifier.
WSDL	Web Services Description Language.
XML	eXtensible Markup Language.
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.

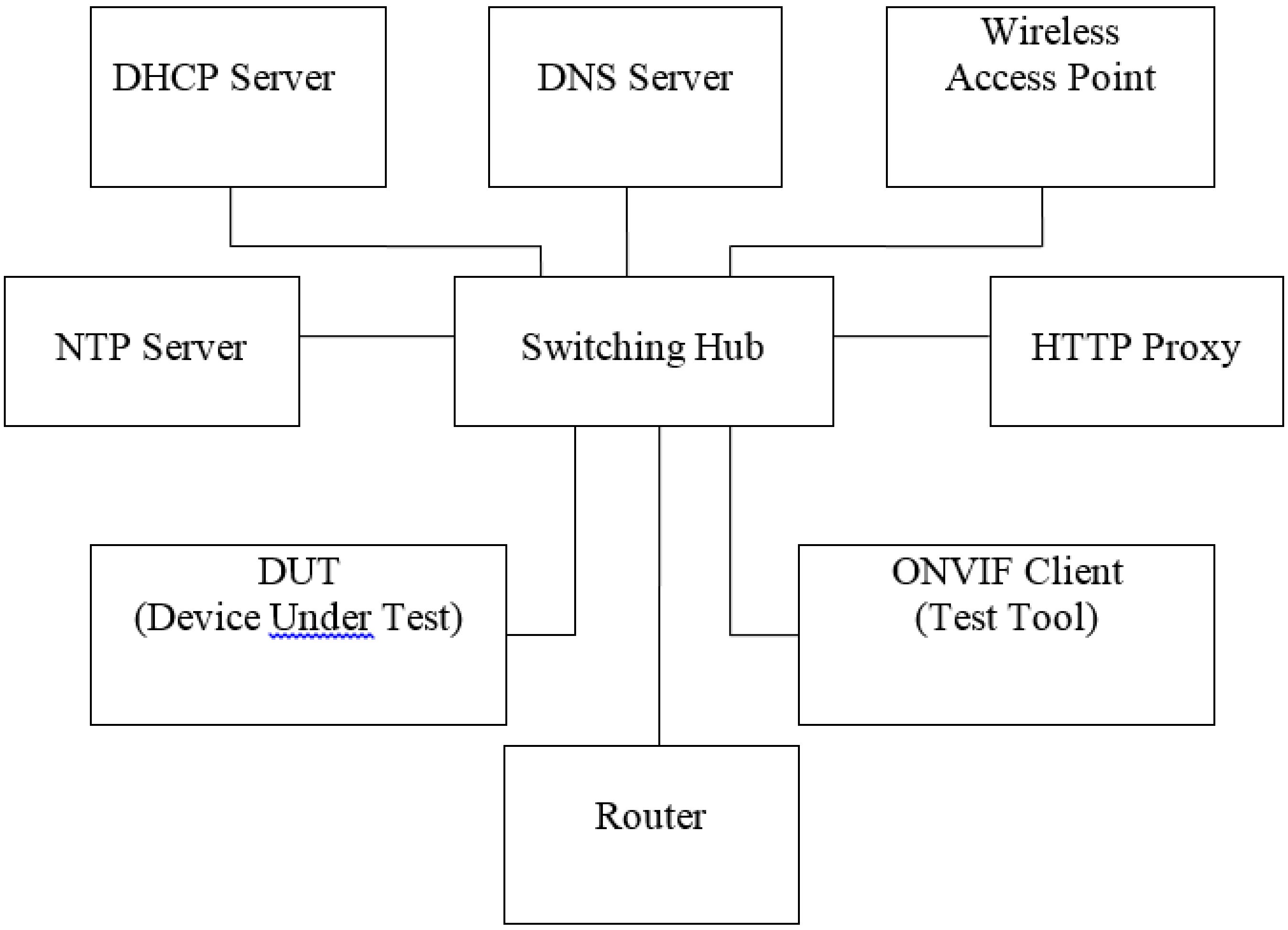
4.1 Test Setup

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 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.

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

Real time streaming test case execution would need the successful execution of some of the Media Configuration test cases. So, Media Configuration features shall be implemented successfully in order to execute the Real Time Streaming test cases.

ONVIF Client shall explicitly specify the optional transport protocols supported by DUT.

ONVIF Client and DUT time should be synchronized for media streaming.

Real time streaming testing will test only one media stream at a time.

Poor streaming test is outside the scope of the ONVIF Test Specification

Please refer to Section 5 for Real Time Streaming Test Cases.

5 Real Time Streaming Test Cases

5.1 Video Streaming

5.1.1 Unicast

5.1.1.1 MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP)

Test Label: Real Time Viewing DUT H.264 media streaming using RTP-Unicast/UDP transport.

Test Case ID: MEDIA2_RTSS-1-1-1

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP/UDP Unicast Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token

- Configuration.Name := *profile*.Configurations.VideoEncoder.Name
- Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
- Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
- Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
- Configuration.Encoding := H264
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
   - in *streamUri* - Uri for media streaming

- in video - media type
- in H.264 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.2 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP)

Test Label: Real Time Viewing DUT H.264 media2 streaming using HTTP transport.

Test Case ID: MEDIA2_RTSS-1-1-2

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on HTTP Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in H264 - required video encoding
- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
    - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
    - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
    - Configuration.Encoding := H264
    - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
    - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
    - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
    - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
    - Configuration.Quality := *vecOptions*.QualityRange.Min
5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp

- ProfileToken := *profile*.@token

7. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in *streamUri* - Uri for media streaming

   - in video - media type

   - in H.264 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.

- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.3 MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP)

Test Label: Real Time Viewing DUT H.264 media2 streaming using RTP/RTSP/TCP transport.

Test Case ID: MEDIA2_RTSS-1-1-3

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP/RTSP/TCP using RTSP tunnel.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}

- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.4 MEDIA2 SET SYNCHRONIZATION POINT – H.264

Test Label: Media2 Configuration DUT Synchronization Point – H.264

Test Case ID: MEDIA2_RTSS-1-1-4

ONVIF Core Specification Coverage: Set synchronization point.

Command Under Test: SetSynchronizationPoint

WSDL Reference: media2.wsdl

Test Purpose: To request synchronization point from DUT for H.264 media stream.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)

- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
9. The DUT responds with **200 OK** message with parameters
   - SDP information =: *sdp*
10. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) with parameters
    - Transport := RTP/AVP;unicast;client_port=*port1-port2*
11. The DUT responds with **200 OK** message with parameters
    - Transport
    - Session =: *session*
12. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
    - Session := *session*
13. The DUT responds with **200 OK** message with parameters

- Session
- RTP-Info

14. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
15. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
16. ONVIF Client invokes **SetSynchronizationPoint** request with parameters
    - ProfileToken := *profile*.@token
17. The DUT responds with **SetSynchronizationPointResponse** message.
18. If DUT does not send I-frame before the regular 'I-frame insertion time interval', FAIL the test and skip other steps.
19. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
    - Session := *session*
20. The DUT responds with **200 OK** message with parameters
    - Session

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.2 for details on 'I-frame insertion time interval'.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.5 MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP, IPv6)

Test Label: Real Time Viewing DUT H.264 media2 streaming using RTP-Unicast/UDP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-5

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP/UDP Unicast Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline

- Configuration.Encoding := H264
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.6 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Label: Real Time Viewing DUT H.264 media2 streaming using HTTP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-6

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on HTTP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in H264 - required video encoding
- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
   - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
   - Configuration.Quality := *vecOptions*.QualityRange.Min
6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspOverHttp

- ProfileToken := *profile*.@token

8. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding

10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.7 MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP, IPv6)

Test Label: Real Time Viewing DUT H.264 media2 streaming using RTP/RTSP/TCP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-7

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP/RTSP/TCP using RTSP tunnel for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height

- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.8 MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP)

Test Label: Real Time Viewing DUT H.265 media streaming using RTP-Unicast/UDP transport.

Test Case ID: MEDIA2_RTSS-1-1-8

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP/UDP Unicast Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token

- Configuration.Name := *profile*.Configurations.VideoEncoder.Name
- Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
- Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
- Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
- Configuration.Encoding := H265
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
   - in *streamUri* - Uri for media streaming

- in video - media type
- in H.265 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.9 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP)

Test Label: Real Time Viewing DUT H.265 media2 streaming using HTTP transport.

Test Case ID: MEDIA2_RTSS-1-1-9

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on HTTP Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in H265 - required video encoding
- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
   - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
   - Configuration.Quality := *vecOptions*.QualityRange.Min
5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspOverHttp

- ProfileToken := *profile*.@token

7. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.10 MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP)

Test Label: Real Time Viewing DUT H.265 media2 streaming using RTP/RTSP/TCP transport.

Test Case ID: MEDIA2_RTSS-1-1-10

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP/RTSP/TCP using RTSP tunnel.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}

- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.11 MEDIA2 SET SYNCHRONIZATION POINT – H.265

Test Label: Media2 Configuration DUT Synchronization Point – H.265

Test Case ID: MEDIA2_RTSS-1-1-11

ONVIF Core Specification Coverage: Set synchronization point.

Command Under Test: SetSynchronizationPoint

WSDL Reference: media2.wsdl

Test Purpose: To request synchronization point from DUT for H.265 media stream.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)

- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
9. The DUT responds with **200 OK** message with parameters
   - SDP information =: *sdp*
10. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) with parameters
    - Transport := RTP/AVP;unicast;client_port=*port1*-*port2*
11. The DUT responds with **200 OK** message with parameters
    - Transport
    - Session =: *session*
12. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
    - Session := *session*
13. The DUT responds with **200 OK** message with parameters

- Session
- RTP-Info

14. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.

15. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.

16. ONVIF Client invokes **SetSynchronizationPoint** request with parameters

    - ProfileToken := *profile*.@token

17. The DUT responds with **SetSynchronizationPointResponse** message.

18. If DUT does not send I-frame before the regular ‘I-frame insertion time interval’, FAIL the test and skip other steps.

19. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters

    - Session := *session*

20. The DUT responds with **200 OK** message with parameters

    - Session

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.2 for details on ‘I-frame insertion time interval’.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.12 MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP, IPv6)

Test Label: Real Time Viewing DUT H.265 media2 streaming using RTP-Unicast/UDP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-12

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP/UDP Unicast Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10

- Configuration.Encoding := H265
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.13 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Label: Real Time Viewing DUT H.265 media2 streaming using HTTP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-13

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on HTTP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in H265 - required video encoding
- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
    - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
    - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
    - Configuration.Encoding := H265
    - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
    - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
    - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
    - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
    - Configuration.Quality := *vecOptions*.QualityRange.Min
6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp

- ProfileToken := *profile*.@token

8. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding

10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.14 MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP, IPv6)

Test Label: Real Time Viewing DUT H.265 media2 streaming using RTP/RTSP/TCP transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-14

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP/RTSP/TCP using RTSP tunnel for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width

- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.15 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP)

Test Label: Real Time Viewing DUT H.264 media2 streaming using HTTPS transport.

Test Case ID: MEDIA2_RTSS-1-1-15

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on HTTPS Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** request.
4. The DUT responds with **GetNetworkProtocolsResponse** with parameters
    - NetworkProtocols list =: *networkProtocolsList*

5. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 8.

6. If the DUT does not support TLS Server, FAIL the test and skip other steps.

7. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.

8. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile

9. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters

   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
   - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast

- Configuration.Quality := *vecOptions*.QualityRange.Min

10. The DUT responds with **SetVideoEncoderConfigurationResponse** message.

11. ONVIF Client invokes **GetStreamUri** request with parameters

    - Protocol := RtspOverHttp

    - ProfileToken := *profile*.@token

12. The DUT responds with **GetStreamUriResponse** message with parameters

    - Uri =: *streamUri*

13. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters

    - in *streamUri* - Uri for media streaming

    - in video - media type

    - in H.264 - expected media stream encoding

14. ONVIF Client restores settings of Video Encoder Configuration with @token = *profile*.Configurations.VideoEncoder.@token.

15. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 8.

16. ONVIF Client restores HTTPS settings wich was changed at step 7.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.

- DUT did not send **GetStreamUriResponse** message.

- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.16 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP)

Test Label: Real Time Viewing DUT H.265 media2 streaming using HTTPS transport.

Test Case ID: MEDIA2_RTSS-1-1-16

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on HTTPS Transport.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** request.
4. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 8.
6. If the DUT does not support TLS Server, FAIL the test and skip other steps.
7. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.
8. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding

- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

9. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
    - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
    - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
    - Configuration.Encoding := H265
    - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
    - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
    - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
    - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
    - Configuration.Quality := *vecOptions*.QualityRange.Min
10. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
11. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token

12. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
13. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in video - media type
    - in H.265 - expected media stream encoding
14. ONVIF Client restores settings of Video Encoder Configuration with @token = *profile*.Configurations.VideoEncoder.@token.
15. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 8.
16. ONVIF Client restores HTTPS settings wich was changed at step 7.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.17 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Label: Real Time Viewing DUT H.264 media2 streaming using HTTPS transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-17

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on HTTPS Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client invokes **GetNetworkProtocols** request.
5. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
6. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 9.
7. If the DUT does not support TLS Server, FAIL the test and skip other steps.
8. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.
9. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
10. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name

- Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
- Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
- Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
- Configuration.Encoding := H264
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

11. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
12. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
13. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
14. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *streamUri* - Uri for media streaming

- in video - media type
- in H.264 - expected media stream encoding

15. ONVIF Client restores settings of Video Encoder Configuration with @token = *profile*.Configurations.VideoEncoder.@token.
16. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 9.
17. ONVIF Client restores HTTPS settings wich was changed at step 8.
18. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.1.18 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Label: Real Time Viewing DUT H.265 media2 streaming using HTTPS transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-1-18

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on HTTPS Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client invokes **GetNetworkProtocols** request.
5. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
6. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 9.
7. If the DUT does not support TLS Server, FAIL the test and skip other steps.
8. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.
9. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
10. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount

- Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
- Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
- Configuration.Encoding := H265
- Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
- Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
- Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
- Configuration.Quality := *vecOptions*.QualityRange.Min

11. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
12. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
13. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
14. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in video - media type

- in H.265 - expected media stream encoding

15. ONVIF Client restores settings of Video Encoder Configuration with @token = *profile*.Configurations.VideoEncoder.@token.
16. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 9.
17. ONVIF Client restores HTTPS settings wich was changed at step 8.
18. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.2 Multicast

5.1.2.1 MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv4)

Test Label: Real Time Viewing DUT H.264 Media2 Streaming Using RTP-Multicast Transport for IPv4.

Test Case ID: MEDIA2_RTSS-1-2-1

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP-Multicast/UDP Transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
   - Configuration.Encoding := H264
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)

- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast.Address.Type := IPv4
- Configuration.Multicast.Address.IPv4Address := multicast IPv4 address
- Configuration.Multicast.Address.IPv6Address skipped
- Configuration.Multicast.Port := port for multicast streaming
- Configuration.Multicast.TTL := 1
- Configuration.Multicast.AutoStart := false
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding
   - in IPv4 - IP version for multicast streaming

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.2.2 MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv6)

Test Label: Real Time Viewing DUT H.264 Media2 Streaming Using RTP-Multicast Transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-2-2

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.264 media streaming based on RTP-Multicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT. IPv6 is supported by DUT. A media profile with H.264 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H264 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters

    - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
    - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
    - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
    - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline
    - Configuration.Encoding := H264
    - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
    - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
    - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
    - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
    - Configuration.Multicast.Address.Type := IPv6
    - Configuration.Multicast.Address.IPv4Address skipped
    - Configuration.Multicast.Address.IPv6Address := multicast IPv6 address
    - Configuration.Multicast.Port := port for multicast streaming
    - Configuration.Multicast.TTL := 1
    - Configuration.Multicast.AutoStart := false
    - Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.

7. ONVIF Client invokes **GetStreamUri** request with parameters

- Protocol := RtspMulticast
- ProfileToken := *profile*.@token

8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.264 - expected media stream encoding
   - in IPv6 - IP version for multicast streaming
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.2.3 MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv4)

Test Label: Real Time Viewing DUT H.265 Media2 Streaming Using RTP-Multicast Transport for IPv4.

Test Case ID: MEDIA2_RTSS-1-2-3

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP-Multicast/UDP Transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in H265 - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)

- Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Multicast.Address.Type := IPv4
- Configuration.Multicast.Address.IPv4Address := multicast IPv4 address
- Configuration.Multicast.Address.IPv6Address skipped
- Configuration.Multicast.Port := port for multicast streaming
- Configuration.Multicast.TTL := 1
- Configuration.Multicast.AutoStart := false
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding
   - in IPv4 - IP version for multicast streaming

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.1.2.4 MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv6)

Test Label: Real Time Viewing DUT H.265 Media2 Streaming Using RTP-Multicast Transport for IPv6.

Test Case ID: MEDIA2_RTSS-1-2-4

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify H.265 media streaming based on RTP-Multicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. H.265 encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT. IPv6 is supported by DUT. A media profile with H.265 video encoder configuration is configured on the Device.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

- in H265 - required video encoding
- out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
- out *vecOptions* - Video Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is Main/Main10
   - Configuration.Encoding := H265
   - Configuration.Resolution.Width := *vecOptions*.ResolutionsAvailable[0].Width
   - Configuration.Resolution.Height := *vecOptions*.ResolutionsAvailable[0].Height
   - Configuration.RateControl.@ConstantBitRate := *profile*.Configurations.VideoEncoder.RateControl.@ConstantBitRate (or skipped if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.FrameRateLimit := *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit (or 0 if *profile*.Configurations.VideoEncoder.RateControl skipped)
   - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
   - Configuration.Multicast.Address.Type := IPv6
   - Configuration.Multicast.Address.IPv4Address skipped
   - Configuration.Multicast.Address.IPv6Address := multicast IPv6 address
   - Configuration.Multicast.Port := port for multicast streaming

- Configuration.Multicast.TTL := 1
- Configuration.Multicast.AutoStart := false
- Configuration.Quality := *vecOptions*.QualityRange.Min

6. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in video - media type
   - in H.265 - expected media stream encoding
   - in IPv6 - IP version for multicast streaming
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2 Audio Streaming

5.2.1 Unicast

5.2.1.1 MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP)

Test Label: Real Time Viewing DUT G.711 Media2 Streaming Using RTP-Unicast/UDP Transport.

Test Case ID: MEDIA2_RTSS-2-1-1

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP-Unicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name

- Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
- Configuration.Encoding := PCMU
- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspUnicast
    - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in G.711 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.2 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP)

Test Label: Real Time Viewing DUT G.711 media2 streaming using HTTP transport.

Test Case ID: MEDIA2_RTSS-2-1-2

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G7.11 media streaming based on HTTP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := PCMU
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast

- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspOverHttp
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in G.711 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.3 MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP)

Test Label: Real Time Viewing DUT G.711 media2 streaming using RTP/RTSP/TCP transport.

Test Case ID: MEDIA2_RTSS-2-1-3

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP/RTSP/TCP using RTSP tunnel.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in PCMU - required audio encoding

   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding

   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token

   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name

   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount

   - Configuration.Encoding := PCMU

   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast

   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list

   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

6. ONVIF Client invokes **GetStreamUri** request with parameters

    - Protocol := RTSP

    - ProfileToken := *profile*.@token

7. The DUT responds with **GetStreamUriResponse** message with parameters

    - Uri =: *streamUri*

8. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters

    - in *streamUri* - Uri for media streaming

    - in audio - media type

    - in G.711 - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.

- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.4 MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP, IPv6)

Test Label: Real Time Viewing DUT G.711 media2 streaming using RTP-Unicast/UDP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-4

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP/UDP Unicast Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
    - in PCMU - required audio encoding
    - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
    - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
    - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
    - Configuration.Encoding := PCMU
    - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
    - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
    - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

7. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

8. ONVIF Client invokes **GetStreamUri** request with parameters

   - Protocol := RtspUnicast

   - ProfileToken := *profile*.@token

9. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

10. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters

    - in *streamUri* - Uri for media streaming

    - in audio - media type

    - in G.711 - expected media stream encoding

11. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.

- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.5 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Label: Real Time Viewing DUT G.711 media2 streaming using HTTP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-5

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on HTTP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := PCMU

- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in G.711 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.6 MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP, IPv6)

Test Label: Real Time Viewing DUT G.711 media2 streaming using RTP/RTSP/TCP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-6

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP/RTSP/TCP using RTSP tunnel for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token

- Configuration.Name := *profile*.Configurations.AudioEncoder.Name
- Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
- Configuration.Encoding := PCMU
- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in G.711 - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.7 MEDIA2 STREAMING – AAC (RTP-Unicast/UDP)

Test Label: Real Time Viewing DUT AAC Media2 Streaming Using RTP-Unicast/UDP Transport.

Test Case ID: MEDIA2_RTSS-2-1-7

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP-Unicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name

- Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
- Configuration.Encoding := MP4A-LATM
- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspUnicast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in AAC - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.8 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP)

Test Label: Real Time Viewing DUT AAC media2 streaming using HTTP transport.

Test Case ID: MEDIA2_RTSS-2-1-8

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G7.11 media streaming based on HTTP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast

- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in AAC - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.9 MEDIA2 STREAMING – AAC (RTP/RTSP/TCP)

Test Label: Real Time Viewing DUT AAC media2 streaming using RTP/RTSP/TCP transport.

Test Case ID: MEDIA2_RTSS-2-1-9

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP/RTSP/TCP using RTSP tunnel.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

6. ONVIF Client invokes **GetStreamUri** request with parameters

   - Protocol := RTSP
   - ProfileToken := *profile*.@token

7. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

8. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in AAC - expected media stream encoding

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.10 MEDIA2 STREAMING – AAC (RTP-Unicast/UDP, IPv6)

Test Label: Real Time Viewing DUT AAC media2 streaming using RTP-Unicast/UDP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-10

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP/UDP Unicast Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

7. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

8. ONVIF Client invokes **GetStreamUri** request with parameters

   - Protocol := RtspUnicast

   - ProfileToken := *profile*.@token

9. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

10. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in Annex A.10 with the following input and output parameters

    - in *streamUri* - Uri for media streaming

    - in audio - media type

    - in AAC - expected media stream encoding

11. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters

    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.

- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.11 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Label: Real Time Viewing DUT AAC media2 streaming using HTTP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-11

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on HTTP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters

   - out *initialNetworkSettings* - initial Network settings

4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM

    - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
    - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
    - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in Annex A.11 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in AAC - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.12 MEDIA2 STREAMING – AAC (RTP/RTSP/TCP, IPv6)

Test Label: Real Time Viewing DUT AAC media2 streaming using RTP/RTSP/TCP transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-12

ONVIF Core Specification Coverage: RTP/RTSP/TCP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP/RTSP/TCP using RTSP tunnel for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

- Configuration.@token := *profile*.Configurations.AudioEncoder.@token
- Configuration.Name := *profile*.Configurations.AudioEncoder.Name
- Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
- Configuration.Encoding := MP4A-LATM
- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RTSP
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP/RTSP/TCP by following the procedure mentioned in Annex A.12 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in AAC - expected media stream encoding
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.13 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP)

Test Label: Real Time Viewing DUT G.711 media2 streaming using HTTPS transport.

Test Case ID: MEDIA2_RTSS-2-1-13

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G7.11 media streaming based on HTTPS Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT.TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** request.
4. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 8.

6. If the DUT does not support TLS Server, FAIL the test and skip other steps.

7. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.

8. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in PCMU - required audio encoding

   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding

   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

9. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token

   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name

   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount

   - Configuration.Encoding := PCMU

   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast

   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list

   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

10. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

11. ONVIF Client invokes **GetStreamUri** request with parameters

   - Protocol := RtspOverHttp

   - ProfileToken := *profile*.@token

12. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

13. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters

- in *streamUri* - Uri for media streaming
- in audio - media type
- in G.711 - expected media stream encoding

14. ONVIF Client restores settings of Audio Encoder Configuration with @token = *profile*.Configurations.AudioEncoder.@token.
15. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 8.
16. ONVIF Client restores HTTPS settings wich was changed at step 7.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.14 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP)

Test Label: Real Time Viewing DUT AAC media2 streaming using HTTPS transport.

Test Case ID: MEDIA2_RTSS-2-1-14

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G7.11 media streaming based on HTTPS Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetNetworkProtocols** request.
4. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
5. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 8.
6. If the DUT does not support TLS Server, FAIL the test and skip other steps.
7. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.
8. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
9. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast

- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

10. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
11. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
12. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
13. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in AAC - expected media stream encoding
14. ONVIF Client restores settings of Audio Encoder Configuration with @token = *profile*.Configurations.AudioEncoder.@token.
15. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 8.
16. ONVIF Client restores HTTPS settings wich was changed at step 7.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.15 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Label: Real Time Viewing DUT G.711 media2 streaming using HTTPS transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-15

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on HTTPS Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client invokes **GetNetworkProtocols** request.
5. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
6. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 9.

7. If the DUT does not support TLS Server, FAIL the test and skip other steps.

8. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.

9. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters

   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

10. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

    - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
    - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
    - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
    - Configuration.Encoding := PCMU
    - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
    - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
    - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

11. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

12. ONVIF Client invokes **GetStreamUri** request with parameters

    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token

13. The DUT responds with **GetStreamUriResponse** message with parameters

    - Uri =: *streamUri*

14. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters

- in *streamUri* - Uri for media streaming
- in audio - media type
- in G.711 - expected media stream encoding

15. ONVIF Client restores settings of Audio Encoder Configuration with @token = *profile*.Configurations.AudioEncoder.@token.
16. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 9.
17. ONVIF Client restores HTTPS settings wich was changed at step 8.
18. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.1.16 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Label: Real Time Viewing DUT AAC media2 streaming using HTTPS transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-1-16

ONVIF Core Specification Coverage: RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on HTTPS Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT.TLS1.0, TLS1.1, TLS1.2, or TLS Server is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Advanced Security Service is received from the DUT, if TLS Server is not supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client invokes **GetNetworkProtocols** request.
5. The DUT responds with **GetNetworkProtocolsResponse** with parameters
   - NetworkProtocols list =: *networkProtocolsList*
6. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step 9.
7. If the DUT does not support TLS Server, FAIL the test and skip other steps.
8. ONVIF Client configures HTTPS by following the procedure mentioned in Annex A.15.
9. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - in MP4A-LATM - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
10. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
    - Configuration.@token := *profile*.Configurations.AudioEncoder.@token

- Configuration.Name := *profile*.Configurations.AudioEncoder.Name
- Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
- Configuration.Encoding := MP4A-LATM
- Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

11. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
12. ONVIF Client invokes **GetStreamUri** request with parameters
    - Protocol := RtspOverHttp
    - ProfileToken := *profile*.@token
13. The DUT responds with **GetStreamUriResponse** message with parameters
    - Uri =: *streamUri*
14. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in Annex A.14 with the following input and output parameters
    - in *streamUri* - Uri for media streaming
    - in audio - media type
    - in AAC - expected media stream encoding
15. ONVIF Client restores settings of Audio Encoder Configuration with @token = *profile*.Configurations.AudioEncoder.@token.
16. ONVIF Client restores Media Profile with @token = *profile*.@token if it was changed at step 9.
17. ONVIF Client restores HTTPS settings wich was changed at step 8.
18. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters

- in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.
- DUT did not send **GetNetworkProtocolsResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.2 Multicast

5.2.2.1 MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv4)

Test Label: Real Time Viewing DUT G.711 Media2 Streaming Using RTP-Multicast Transport for IPv4.

Test Case ID: MEDIA2_RTSS-2-2-1

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP-Multicast/UDP Transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := PCMU
   - Configuration.Multicast.Address.Type := IPv4
   - Configuration.Multicast.Address.IPv4Address := multicast IPv4 address
   - Configuration.Multicast.Address.IPv6Address skipped
   - Configuration.Multicast.Port := port for multicast streaming
   - Configuration.Multicast.TTL := 1
   - Configuration.Multicast.AutoStart := false
   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list
5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast

- ProfileToken := *profile*.@token

7. The DUT responds with **GetStreamUriResponse** message with parameters

   - Uri =: *streamUri*

8. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in G.711 - expected media stream encoding
   - in IPv4 - IP version for multicast streaming

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.2.2 MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv6)

Test Label: Real Time Viewing DUT G.711 Media2 Streaming Using RTP-Multicast Transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-2-2

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP-Multicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. RTP-Multicast transport protocol is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
   - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
   - in PCMU - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := PCMU
   - Configuration.Multicast.Address.Type := IPv6
   - Configuration.Multicast.Address.IPv4Address skipped
   - Configuration.Multicast.Address.IPv6Address := multicast IPv6 address
   - Configuration.Multicast.Port := port for multicast streaming
   - Configuration.Multicast.TTL := 1
   - Configuration.Multicast.AutoStart := false

- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in G.711 - expected media stream encoding
   - in IPv6 - IP version for multicast streaming
10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
    - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.2.3 MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4)

Test Label: Real Time Viewing DUT AAC Media2 Streaming Using RTP-Multicast Transport for IPv4.

Test Case ID: MEDIA2_RTSS-2-2-3

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP-Multicast/UDP Transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. RTP-Multicast transport protocol is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.

3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters

   - in MP4A-LATM - required audio encoding

   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding

   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters

   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token

   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name

   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount

   - Configuration.Encoding := MP4A-LATM

- Configuration.Multicast.Address.Type := IPv4
- Configuration.Multicast.Address.IPv4Address := multicast IPv4 address
- Configuration.Multicast.Address.IPv6Address skipped
- Configuration.Multicast.Port := port for multicast streaming
- Configuration.Multicast.TTL := 1
- Configuration.Multicast.AutoStart := false
- Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
- Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
7. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
8. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters
   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in AAC - expected media stream encoding
   - in IPv4 - IP version for multicast streaming

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

5.2.2.4 MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6)

Test Label: Real Time Viewing DUT AAC Media2 Streaming Using RTP-Multicast Transport for IPv6.

Test Case ID: MEDIA2_RTSS-2-2-4

ONVIF Core Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP.

Command Under Test: None

WSDL Reference: None

Test Purpose: To verify AAC media streaming based on RTP-Multicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. RTP-Multicast transport protocol is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures IPv6 address to use it for the next test steps by following the procedure mentioned in Annex A.4 with the following input and output parameters
    - out *initialNetworkSettings* - initial Network settings
4. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - in MP4A-LATM - required audio encoding

- out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
- out *aecOptions* - Audio Encoder Configuration Options for the Media Profile

5. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := MP4A-LATM
   - Configuration.Multicast.Address.Type := IPv6
   - Configuration.Multicast.Address.IPv4Address skipped
   - Configuration.Multicast.Address.IPv6Address := multicast IPv6 address
   - Configuration.Multicast.Port := port for multicast streaming
   - Configuration.Multicast.TTL := 1
   - Configuration.Multicast.AutoStart := false
   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list
6. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
7. ONVIF Client invokes **GetStreamUri** request with parameters
   - Protocol := RtspMulticast
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetStreamUriResponse** message with parameters
   - Uri =: *streamUri*
9. ONVIF Client tries to start and decode media streaming over RTP-Multicast by following the procedure mentioned in Annex A.13 with the following input and output parameters

   - in *streamUri* - Uri for media streaming
   - in audio - media type
   - in AAC - expected media stream encoding
   - in IPv6 - IP version for multicast streaming

10. ONVIF Client restores network settings by following the procedure mentioned in Annex A.5 with the following input and output parameters
   - in *initialNetworkSettings* - initial Network settings

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **SetAudioEncoderConfigurationResponse** message.
- DUT did not send **GetStreamUriResponse** message.

Note: See Annex A.3 for Name and Token Parameters Length limitations.

Annex A Helper Procedures and Additional Notes

A.1 Invalid RTP Header

A RTP header, which is not formed according to the header field format defined in the RFC 3550 Section 5.1, is considered an invalid RTP header.

A.2 I-frame insertion time interval

‘I-frame insertion time interval’ is the time interval between two consecutive I-frames sent by DUT.

ONVIF Client calculates this value by using the ‘GovLength’ parameter in the Video encoder configuration. ONVIF Client has to configure ‘GovLength’ to a large enough value so that there will be a sufficient time difference between two I-frames.

For SetSynchronizationPoint test cases in the “Real Time Streaming” section, ONVIF Client follows this procedure to verify that I-frame is inserted as a result of SetSynchronizationPoint request.

ONVIF Client waits for an I-frame before invoking SetSynchronizationPoint command.

After receiving I-frame, ONVIF Client starts a timer with time out period less than ‘I-frame insertion time interval’ and immediately invokes SetSynchronizationPoint command.

ONVIF Client waits for the I-frame and verifies that it receives I-frame before the timeout period.

A.3 Name and Token Parameters

There are the following limitations on maximum length of the Name and Token parameters that shall 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).
- 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.4 Turn on IPv6 network interface

Name: HelperTurnOnIPv6

Procedure Purpose: Helper procedure to turn on IPv6 network interface.

Pre-requisite: IPv6 is supported by DUT.

Input: None

Returns: Initial Network settings (*initialNetworkSettings*).

Procedure:

1. ONVIF Client will invoke GetNetworkInterfacesRequest message to retrieve the original settings of the DUT.
2. ONVIF Client verifies GetNetworkInterfacesResponse message.
3. Set *initialNetworkSettings* := available network interface.
4. If GetNetworkInterfacesResponse message contains NetworkInterfaces.IPv6 and NetworkInterfaces.IPv6.Enabled=true, then ONVIF Client checks NetworkInterfaces.IPv6.Config.DHCP. Otherwise, go to step 11.
5. If NetworkInterfaces.IPv6.Config.DHCP=Off, then ONVIF Client checks NetworkInterfaces.IPv6.Config.Manual element. Otherwise, go to step 8.
6. If NetworkInterfaces.IPv6.Config.Manual element is present and not empty, then ONVIF Client skips other steps and run test using NetworkInterfaces.IPv6.Config.Manual value as device IP. Otherwise, ONVIF Client checks NetworkInterfaces.IPv6.Config.LinkLocal element.
7. If NetworkInterfaces.IPv6.Config.LinkLocal element is present and not empty, then ONVIF Client skips other steps and runs test using NetworkInterfaces.IPv6.Config.LinkLocal value as device IP. Otherwise, ONVIF Client skip other steps and failed test.
8. ONVIF Client will invoke SetNetworkInterfacesRequest message to turn off DHCP IPv6 (InterfaceToken = available network interface, NetworkInterfaces.IPv6.Config.DHCP=Off).
9. ONVIF Client gets current network interfaces via GetNetworkInterfacesRequest message.
10. ONVIF Client verifies GetNetworkInterfacesResponse message and checks that set settings were applied. Repeat steps 6-7.
11. If GetNetworkInterfacesResponse message does not contain NetworkInterfaces.IPv6 or NetworkInterfaces.IPv6.Enabled=false, then ONVIF Client will invoke SetNetworkInterfacesRequest message (InterfaceToken = available network interface, NetworkInterfaces.IPv6. Enabled=true) to turn on IPv6 configuration.
12. The DUT will return SetNetworkInterfacesResponse message.
13. If Reboot is required by DUT, invoke SystemReboot command.

14. ONVIF Client waits for HELLO message from the default network interface.

15. ONVIF Client gets current network interfaces via GetNetworkInterfacesRequest message.

16. ONVIF Client verifies GetNetworkInterfacesResponse message and checks that set settings were applaied. Execute steps 5-7.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemReboot** message.

A.5 Restore Network Settings

Name: HelperRestoreNetworkSettings

Procedure Purpose: Helper procedure to restore the original default settings.

Pre-requisite: None

Input: Initial Network settings to restore (*initialNetworkSettings*).

Returns: None

Procedure:

1. Restore the initial network settings by invoking SetNetworkInterfaces (Default settings) command.

2. If Reboot is required by DUT, invoke SystemReboot command.

3. If SystemReboot is invoked, wait for HELLO message from the default network interface.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetNetworkInterfacesResponse** message.
- DUT did not send **SetNetworkInterfacesResponse** message.
- DUT did not send **SystemReboot** message.

A.6 Media2 Service Profile Configuration for Video Streaming

Name: HelperFindMediaProfileForVideoStreaming

Procedure Purpose: Helper procedure to configure Media Profile to contain Video Source Configuration and Video Encoder Configuration with the required video encoding.

Pre-requisite: Media2 Service is received from the DUT.

Input: Required video encoding (*requiredVideoEncoding*)

Returns: Media Profile (*profile*) containing Video Source Configuration and Video Encoder Configuration with the required video encoding. Video Encoder Configuration Options for the Media Profile (*vecOptions*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := VideoEncoder
2. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
3. For each Media Profile *profile1* in *profileList* repeat the following steps:

   3.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
   - ConfigurationToken skipped
   - ProfileToken := *profile1*.@token

   3.2. The DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
   - Options list =: *vecOptionsList*

   3.3. If *vecOptionsList* list contains item with Encoding = *requiredVideoEncoding*:

3.3.1. Set *vecOptions* := item at *vecOptionsList* list with Encoding = *requiredVideoEncoding*.

3.3.2. Set *profile* := *profile1*.

3.3.3. Skip other steps in procedure.

4. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.7 Media2 Service – Media Profile Configuration for Audio Streaming

Name: HelperConfigureMediaProfileForAudioStreaming

Procedure Purpose: Helper procedure to configure Media Profile to contain Audio Source Configuration and Audio Encoder Configuration with the required audio encoding.

Pre-requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT.

Input: Required audio encoding (*requiredAudioEncoding*)

Returns: Media Profile (*profile*) containing Audio Source Configuration and Audio Encoder Configuration with the required audio encoding. Audio Encoder Configuration Options for the Media Profile (*aecOptions*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := AudioSource
   - Type[1] := AudioEncoder

2. The DUT responds with **GetProfilesResponse** message with parameters

   - Profiles list =: *profileList*

3. For each Media Profile *profile1* in *profileList* with both Configuration.AudioSource and Configuration.AudioEncoder repeat the following steps:

   3.1. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** request with parameters

   - ConfigurationToken := *profile1*.Configuration.AudioEncoder.@token

   - ProfileToken := *profile1*.@token

   3.2. DUT responds with **GetAudioEncoderConfigurationOptionsResponse** message with parameters

   - Options list =: *optionsList*

   3.3. If *optionsList* list contains item with Encoding = *requiredAudioEncoding*:

   3.3.1.Set *profile* := *profile1*.

   3.3.2.Set *aecOptions* := item with Encoding = *requiredAudioEncoding* from *optionsList* list.

   3.3.3.Skip other steps in procedure.

4. For each Media Profile *profile1* in *profileList* repeat the following steps:

   4.1. ONVIF Client invokes **GetAudioSourceConfigurations** request with parameters

   - ConfigurationToken skipped

   - ProfileToken := *profile1*.@token

   4.2. The DUT responds with **GetAudioSourceConfigurationsResponse** with parameters

   - Configurations list =: *audioSourceConfList*

   4.3. For each Audio Source Configuration *audioSourceConfiguration1* in *audioSourceConfList* repeat the following steps:

   4.3.1. ONVIF Client invokes **AddConfiguration** request with parameters

   - ProfileToken := *profile1*.@token

   - Name skipped

- Configuration[0].Type := AudioSource
- Configuration[0].Token := *audioSourceConfiguration1*.@token

4.3.2. The DUT responds with **AddConfigurationResponse** message.

4.3.3. ONVIF Client invokes **GetAudioEncoderConfigurations** request with parameters

- ConfigurationToken skipped
- ProfileToken := *profile1*.@token

4.3.4. The DUT responds with **GetAudioEncoderConfigurationsResponse** with parameters

- Configurations list =: *audioEncoderConfList*

4.3.5. For each Audio Encoder Configuration *audioEncoderConfiguration1* in *audioEncoderConfList* repeat the following steps:

4.3.5.1. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *audioEncoderConfiguration1*.@token
- ProfileToken := *profile1*.@token

4.3.5.2. DUT responds with **GetAudioEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

4.3.5.3. If *optionsList* list contains item with Encoding = *requiredAudioEncoding*:

4.3.5.3.1. ONVIF Client invokes **AddConfiguration** request with parameters

- ProfileToken := *profile1*.@token
- Name skipped
- Configuration[0].Type := AudioEncoder

- Configuration[0].Token := *audioEncoderConfiguration1*.@token

4.3.5.3.2. The DUT responds with **AddConfigurationResponse** message.

4.3.5.3.3. Set *profile* := *profile1*.

4.3.5.3.4. Set *aecOptions* := item with Encoding = *requiredAudioEncoding* from *optionsList* list.

4.3.5.3.5. Skip other steps in procedure.

5. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetAudioEncoderConfigurationOptionsResponse** message.
- DUT did not send **GetAudioSourceConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetAudioEncoderConfigurationsResponse** message.

A.8 Removing Video Encoder Configuration and Metadata Configuration from Media Profile

Name: HelperRemoveVideoEncoderConfigAndMetadataConfigFromMediaProfile

Procedure Purpose: Helper Procedure to guarantee that Media Profile does not contain Video Encoder Configuration and Metadata Configuration.

Pre-requisite: Media2 Service is received from the DUT.

Input: Media Profile (*profile*)

Returns: Updated Media Profile (*profile*) without Video Encoder Configuration and Metadata Configuration.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token := *profile*.@token
   - Type[0] := VideoEncoder
   - Type[1] := Metadata
2. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
3. If *profileList*[0] contains Configuration.VideoEncoder or Configuration.Metadata:

   3.1. ONVIF Client invokes **RemoveConfiguration** request with parameters
   - ProfileToken := *profile1*.@token
   - If *profileList*[0] contains Configuration.VideoEncoder:
     - Configuration[0].Type := VideoEncoder
     - Configuration[0].Token skipped
   - If *profileList*[0] contains Configuration.Metadata:
     - Configuration[1].Type := Metadata
     - Configuration[1].Token skipped

   3.2. The DUT responds with **RemoveConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.9 Removing Audio Encoder Configuration and Metadata Configuration from Media Profile

Name: HelperRemoveAudioEncoderConfigAndMetadataConfigFromMediaProfile

Procedure Purpose: Helper Procedure to guarantee that Media Profile does not contain Audio Encoder Configuration and Metadata Configuration.

Pre-requisite: Media2 Service is received from the DUT.

Input: Media Profile (*profile*)

Returns: Updated Media Profile (*profile*) without Audio Encoder Configuration and Metadata Configuration.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token := *profile*.@token
   - Type[0] := AudioEncoder
   - Type[1] := Metadata
2. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
3. If *profileList*[0] contains Configuration.AudioEncoder or Configuration.Metadata:

   3.1. ONVIF Client invokes **RemoveConfiguration** request with parameters
   - ProfileToken := *profile1*.@token
   - If *profileList*[0] contains Configuration.AudioEncoder:
     - Configuration[0].Type := AudioEncoder
     - Configuration[0].Token skipped
   - If *profileList*[0] contains Configuration.Metadata:
     - Configuration[1].Type := Metadata
     - Configuration[1].Token skipped

   3.2. The DUT responds with **RemoveConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.10 Media Streaming over RTP-Unicast/UDP

Name: HelperStreamingRTPUnicastUDP

Procedure Purpose: Helper procedure to verify media streaming over RTP-Unicast/UDP.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*). Media type (*mediaType*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
   - SDP information =: *sdp*
3. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) with parameters
   - Transport := RTP/AVP;unicast;client_port=*port1-port2*
4. The DUT responds with **200 OK** message with parameters
   - Transport
   - Session =: *session*
5. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*

6. The DUT responds with **200 OK** message with parameters
   - Session
   - RTP-Info
7. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
8. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
9. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*
10. The DUT responds with **200 OK** message with parameters
    - Session

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response for **RTSP DESCRIBE**, **RTSP SETUP**, **RTSP PLAY** and **RTSP TEARDOWN** requests.
- RTSP Session is terminated by DUT during media streaming.

Note: See Annex A.1 for invalid RTP header definition.

A.11 Media Streaming over RTP-Unicast/RTSP/HTTP/TCP

Name: HelperStreamingRTPUnicastRTSPHTTPTCP

Procedure Purpose: Helper procedure to verify media streaming over RTP-Unicast/RTSP/HTTP/TCP.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*). Media type (*mediaType*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **HTTP GET** request to *streamUri* address to establish DUT to ONVIF Client connection for RTP data transfer (*connection1*).
2. ONVIF Client invokes **HTTP POST** request to *streamUri* address to establish ONVIF Client to DUT connection for RTSP control requests (*connection2*).
3. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address converted to rtsp address on *connection2*.
4. The DUT responds with **200 OK** message with parameters on *connection1*
   - SDP information =: *sdp*
5. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) on *connection2* with parameters
   - Transport := RTP/AVP/TCP;unicast;client_port=*port1-port2*
6. The DUT responds with **200 OK** message on *connection1* with parameters
   - Transport
   - Session =: *session*
7. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) on *connection2* with parameters
   - Session := *session*
8. The DUT responds with **200 OK** message on *connection1* with parameters
   - Session
   - RTP-Info
9. If DUT does not send *encoding* RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.
10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
11. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) on *connection2* with parameters
    - Session := *session*
12. ONVIF Client closes *connection2*.

13. The DUT responds with **HTTP 200 OK** message on *connection1* and closes *connection1*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response for **RTSP DESCRIBE**, **RTSP SETUP** and **RTSP PLAY** requests.
- RTSP Session is terminated by DUT during media streaming.

Note: See Annex A.1 for invalid RTP header definition.

A.12 Media Streaming over RTP/RTSP/TCP

Name: HelperStreamingRTPRTSPTCP

Procedure Purpose: Helper procedure to verify media streaming over RTP/RTSP/TCP.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*). Media type (*mediaType*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
   - SDP information =: *sdp*
3. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) with parameters
   - Transport := RTP/AVP/TCP;unicast;interleaved=0-1
4. The DUT responds with **200 OK** message with parameters
   - Transport
   - Session =: *session*

5. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*
6. The DUT responds with **200 OK** message with parameters
   - Session
   - RTP-Info
7. If DUT does not send *encoding* RTP media stream to ONVIF Client over RTSP control connection, FAIL the test and skip other steps.
8. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
9. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*
10. The DUT responds with **200 OK** message with parameters
    - Session

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response for **RTSP DESCRIBE**, **RTSP SETUP**, **RTSP PLAY** and **RTSP TEARDOWN** requests.
- RTSP Session is terminated by DUT during media streaming.

Note: See Annex A.1 for invalid RTP header definition.

A.13 Media Streaming over RTP-Multicast

Name: HelperStreamingRTPMulticast

Procedure Purpose: Helper procedure to verify media streaming over RTP-Multicast.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*). Media type (*mediaType*). Expected media stream encoding (*encoding*). IP version (*ipVersion*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
   - SDP information =: *sdp*
3. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) with parameters
   - Transport := RTP/AVP;multicast;client_port=*port1*-*port2*
4. The DUT responds with **200 OK** message with parameters
   - Transport
   - Session =: *session*
5. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*
6. The DUT responds with **200 OK** message with parameters
   - Session
   - RTP-Info
7. If DUT does not send *encoding* RTP *ipVersion* multicast media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
8. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
9. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) with parameters
   - Session := *session*
10. The DUT responds with **200 OK** message with parameters
    - Session

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response for **RTSP DESCRIBE**, **RTSP SETUP**, **RTSP PLAY** and **RTSP TEARDOWN** requests.
- RTSP Session is terminated by DUT during media streaming.

Note: See Annex A.1 for invalid RTP header definition.

A.14 Media Streaming over RTP-Unicast/RTSP/HTTPS/TCP

Name: HelperStreamingRTPUnicastRTSPHTTPSTCP

Procedure Purpose: Helper procedure to verify media streaming over RTP-Unicast/RTSP/HTTPS/TCP.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*). Media type (*mediaType*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **HTTPS GET** request to *streamUri* address to establish DUT to ONVIF Client secured connection for RTP data transfer (*connection1*).
2. ONVIF Client invokes **HTTPS POST** request to *streamUri* address to establish ONVIF Client to DUT secured connection for RTSP control requests (*connection2*).
3. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address converted to rtsp address on *connection2*.
4. The DUT responds with **200 OK** message with parameters on *connection1*
    - SDP information =: *sdp*
5. ONVIF Client invokes **RTSP SETUP** request to uri address which corresponds to *mediaType* media type (see [RFC2326] for details) on *connection2* with parameters

- Transport := RTP/AVP/TCP;unicast;client_port=*port1-port2*

6. The DUT responds with **200 OK** message on *connection1* with parameters
   - Transport
   - Session =: *session*
7. ONVIF Client invokes **RTSP PLAY** request to uri address which corresponds to agregate control (see [RFC2326] for details) on *connection2* with parameters
   - Session := *session*
8. The DUT responds with **200 OK** message on *connection1* with parameters
   - Session
   - RTP-Info
9. If DUT does not send *encoding* RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.
10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
11. ONVIF Client invokes **RTSP TEARDOWN** request to uri address which corresponds to agregate control (see [RFC2326] for details) on *connection2* with parameters
    - Session := *session*
12. ONVIF Client closes *connection2*.
13. The DUT responds with **HTTP 200 OK** message on *connection1* and closes *connection1*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response for **RTSP DESCRIBE**, **RTSP SETUP** and **RTSP PLAY** requests.
- RTSP Session is terminated by DUT during media streaming.

Note: See Annex A.1 for invalid RTP header definition.

A.15 Configuring HTTPS using Advanced Security

Name: HelperConfigureHTTPS

Procedure Purpose: Helper Procedure to configure HTTPS using Advanced Security.

Pre-requisite: Advanced Security Service is received from the DUT. TLS Server is supported by the DUT. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment. Current time of the DUT shall be at least Jan 01, 1970.

Input: None

Returns: None

Procedure:

1. If Create self-signed certificate is supported by the DUT:

   1.1. ONVIF Client Client adds server certification assignment and creates related certification path, the self-signed certificate and the RSA key pair by following the procedure mentioned in Annex A.16.

   1.2. Go to the step 3.

2. ONVIF Client creates a certification path based on CA-signed certificate and related RSA key pair and a corresponding CA certificate and related RSA key pair by following the procedure mentioned in Annex A.17.

3. ONVIF Client invokes **SetNetworkProtocols** request with parameters

   - NetworkProtocols[0].Name := HTTPS

   - NetworkProtocols[0].Enabled := true

   - NetworkProtocols[0].Port := 443

   - NetworkProtocols[0].Extension skipped

4. The DUT responds with **SetNetworkProtocolsResponse** message.

5. ONVIF Client waits until *timeout1* timeout expires.

6. ONVIF Client checks that HTTPS protocol Port 443 is open. If HTTPS protocol port HTTPS is not open, FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

A.16 Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair

Name: HelperAddServerCertAssign_SSCertificate

Procedure Purpose: Helper Procedure to configure HTTPS using Advanced Security.

Pre-requisite: Advanced Security Service is received from the DUT. TLS Server is supported by the DUT. Create self-signed certificate is supported by the DUT. RSA key pair generation is supported by the DUT. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

Input: None

Returns: The identifiers of the new certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*).

Procedure:

1. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters
   - out *keyID* - RSA key pair
2. ONVIF Client invokes **CreateSelfSignedCertificate** with parameters
   - X509Version skipped
   - KeyID := *keyID*

- Subject := subject (see Annex A.19)
- Alias skipped
- notValidBefore skipped
- notValidAfter skipped
- SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
- SignatureAlgorithm.parameters skipped
- SignatureAlgorithm.anyParameters skipped
- Extension skipped

3. The DUT responds with a **CreateSelfSignedCertificateResponse** message with parameters
   - CertificateID =: *certID*
4. ONVIF Client invokes **CreateCertificationPath** with parameters
   - CertficateIDs.CertificateID[0] := *certID*
   - Alias := "ONVIF_Test"
5. The DUT responds with a **CreateCertificationPathResponse** message with parameters
   - CertificationPathID =: *certPathID*
6. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
   - CertificationPathID := *certPathID*
7. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreateSelfSignedCertificateResponse** message.
- DUT did not send **CreateCertificationPathResponse** message.

- DUT did not send **AddServerCertificateAssignmentResponse** message.

A.17 Add server certificate assignment with corresponding certification path, CA certificate and RSA key pair

Name: HelperAddServerCertAssign_CACertificate

Procedure Purpose: Helper Procedure to configure HTTPS using Advanced Security.

Pre-requisite: Advanced Security Service is received from the DUT. TLS Server is supported by the DUT. Create PCKS#10 supported by the DUT. RSA key pair generation is supported by the DUT. The DUT shall have enough free storage capacity for one additional RSA key pair. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional server certificate assignment.

Input: None

Returns: The identifiers of the new certification path (*certPathID*), certificate (*certID*) and RSA key pair (*keyID*).

Procedure:

1. ONVIF Client creates an RSA key pair by following the procedure mentioned in Annex A.18 with the following input and output parameters
   - out *keyID* - RSA key pair
2. ONVIF Client invokes **CreatePKCS10CSR** with parameter
   - Subject := subject (see Annex A.19)
   - KeyID := *keyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := 1.2.840.113549.1.1.5 (OID of SHA-1 with RSA Encryption algorithm)
3. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *pkcs10*
4. Create an [RFC5280] compliant X.509 certificate (*cert*) from the PKCS#10 request (*pkcs10*) with the following properties:

- version:= v3
- signature := sha1-WithRSAEncryption
- subject := subject from the PKCS#10 request (*pkcs10*)
- subject public key := subject public key in the PKCS#10 request (*pkcs10*)
- validity := not before 19700101000000Z and not after 99991231235959Z
- certificate signature is generated with the private key (*privateKey*) in the CA certificate (*CAcert*)
- certificate extensions := the X.509v3 extensions from the PKCS#10 request (*pkcs10*)

5. ONVIF Client invokes **UploadCertificate** with parameters
    - Certificate := *cert*
    - Alias := "ONVIF_Test1"
    - PrivateKeyRequired := true
6. The DUT responds with a **UploadCertificateResponse** message with parameters
    - CertificateID =: *certID*
    - KeyID =: *keyID*
7. ONVIF Client invokes **CreateCertificationPath** with parameters
    - CertficateIDs.CertificateID[0] := *certID*
    - Alias := "ONVIF_Test2"
8. The DUT responds with a **CreateCertificationPathResponse** message with parameters
    - CertificationPathID =: *certPathID*
9. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
    - CertificationPathID := *certPathID*
10. The DUT responds with an **AddServerCertificateAssignmentResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreatePKCS10CSRResponse** message.
- DUT did not send **UploadCertificateResponse** message.
- DUT did not send **CreateCertificationPathResponse** message.
- DUT did not send **AddServerCertificateAssignmentResponse** message.

A.18 Create an RSA key pair

Name: HelperCreateRSAKeyPair

Procedure Purpose: Helper procedure to create an RSA key pair.

Pre-requisite: Advanced Security Service is received from the DUT. RSA key pair generation is supported by the DUT. The DUT shall have enough free storage capacity for one additional RSA key pair.

Input: None

Returns: The identifier of the new and RSA key pair (*keyID*).

Procedure:

1. ONVIF Client invokes **GetServiceCapabilities** request.
2. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*
3. Set *keyLength* := the smallest supported key length at *cap*.RSAKeyLengths.
4. ONVIF Client invokes **CreateRSAKeyPair** with parameter
    - KeyLength := *length*
5. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
    - KeyID =: *keyID*
    - EstimatedCreationTime =: *duration*
6. Until *duration* + *timeout1* expires repeat the following steps:

6.1. ONVIF Client waits for time *duration*.

6.2. ONVIF Client invokes **GetKeyStatus** with parameters

- KeyID := *keyID*

6.3. The DUT responds with **GetKeyStatusResponse** message with parameters

- KeyStatus =: *keyStatus*

6.4. If *keyStatus* is equal to "ok", *keyID*, skip other steps of the procedure.

6.5. If *keyStatus* is equal to "corrupt", FAIL the test and skip other steps.

7. If *timeout1* expires for step 6 and the last *keyStatus* is other than "ok", FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetKeyStatusResponse** message.
- DUT did not send **CreateRSAKeyPairResponse** message.
- DUT did not send **GetServiceCapabilitiesResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

A.19 Subject for a server certificate

Use the following subject for test cases:

- Subject.Country := "US"
- Subject.CommonName := DUT IP-address