

ONVIF[®]

Real Time Streaming using Media2 Device Test Specification

Version 21.06

June 2021

© 2021 ONVIF, Inc. All rights reserved.

Recipients of this document may copy, distribute, publish, or display this document so long as this copyright notice, license and disclaimer are retained with all copies of the document. No license is granted to modify this document.

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

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

REVISION HISTORY

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 (Start and Stop Multicast 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 – AAC (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 specification was converted to new format.

		<p>HTTPS test cases were updated according comments to ticket #1168:</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSPS/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSPS/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p>
17.06	Feb 06, 2017	<p>Multicast test cases were updated according to ticket #1297:</p> <p>MEDIA2 STREAMING - H.264 (RTSPMulticast, IPv4)</p> <p>MEDIA2 STREAMING - H.264 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING - H.265 (RTSPMulticast, IPv4)</p> <p>MEDIA2 STREAMING - H.265 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6)</p>
17.06	Feb 09, 2017	<p>All Video Streaming test cases were updated according to ticket #1306.</p> <p>Annex A.6 Media2 Service Profile Configuration for Video Streaming was updated according to ticket #1306.</p>
17.06	Feb 20, 2017	<p>Audio backchannel test cases were added according to ticket #1164:</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP-Multicast/UDP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP-Multicast/UDP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv6)</p>

		<p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv6)</p> <p>BACKCHANNEL – G.711 (RTP-Multicast/UDP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Multicast/UDP, IPv6)</p> <p>Scope/Real Time Streaming section was updated to include audio backchannel streaming.</p>
17.06	Mar 10, 2017	<p>Metadata Streaming test cases were added according to ticket #1158:</p> <p>METADATA STREAMING (RTP-Unicast/UDP)</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>METADATA STREAMING (RTP/RTSP/TCP)</p> <p>METADATA STREAMING - SET SYNCHRONIZATION POINT</p> <p>METADATA STREAMING (RTP-Unicast/UDP, IPv6)</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>METADATA STREAMING (RTP/RTSP/TCP, IPv6)</p> <p>METADATA STREAMING (RTP-Multicast/UDP)</p> <p>METADATA STREAMING (RTP-Multicast/UDP, IPv6)</p> <p>Scope/Real Time Streaming section was updated to include metadata streaming.</p>
17.06	Apr 24, 2017	<p>Annex A.17 added and Annex A.39 updated to add more description.</p>
17.06	Apr 27, 2017	<p>Typo fixes according #1168.</p>
17.06	May 26, 2017	<p>The following test cases were added according to #1322:</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/WebSockets)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/WebSockets)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/WebSockets)</p>
17.06	Jun 5, 2017	<p>The following test cases were added according to #1322:</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/WebSockets, IPv6)</p>

		<p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/WebSockets)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets)</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets, IPv6)</p>
17.06	Jun 20, 2017	<p>The following test cases and Annexes were changed according to #1315:</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP)</p> <p>MEDIA2 SET SYNCHRONIZATION POINT – H.264</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP, IPv6)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP)</p> <p>MEDIA2 SET SYNCHRONIZATION POINT – H.265</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP)</p>

		<p>MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/UDP)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>MEDIA2 STREAMING – AAC (RTP/RTSP/TCP)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/UDP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP/RTSP/TCP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4)</p> <p>MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6)</p> <p>Annex A.40 Device Configuration for Video Streaming</p> <p>Annex A.41 Device Configuration for Audio Streaming</p> <p>Annex A.42 Backchannel Streaming over WebSocket</p> <p>Annex A.43 Configuring HTTPS if Required was added according to #1315</p> <p>Annex A.44 Removing Configurations from Media Profile was added according to #1315</p>
18.06	Jun 21, 2018	Reformatting document using new template
17.12	Jul 10, 2017	<p>The following Annexes were changed to check stream uri to #1346:</p> <p>Annex A.20 Media2 Service – Media Profile Configuration for Audio Backchannel Streaming</p> <p>Annex A.31 Media2 Service – Media Profile Configuration for Metadata Streaming</p> <p>Annex A.40 Get Stream Uri</p>
17.12	Aug 1, 2017	The following test cases and Annexes were added according to #1348:

		<p>MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/UDP)</p> <p>MEDIA2_RTSS-5-1-1 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP)</p> <p>Annex A.45 Removing Configurations from Media Profile</p> <p>Annex A.46 Device Configuration for Audio Streaming using Media Profile that contains only Audio Configurations</p> <p>Annex A.47 Device Configuration for Video and Audio Streaming</p> <p>Annex A.48 Media2 Service – Adding AudioSource and AudioEncoder configurations to Media Profile</p> <p>The following Annexes were changed according to #1348:</p> <p>Annex A.10 Media Streaming over RTP-Unicast/UDP</p>
17.12	Aug 31, 2017	<p>The following Annex was added according to #1285:</p> <p>Annex A.49 RTSP Authentication Check</p>
17.12	Oct 04, 2017	<p>The following test cases were removed according to #1474:</p> <p>BACKCHANNEL – G.711 (RTP-Multicast/UDP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP-Multicast/UDP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Multicast/UDP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP-Multicast/UDP, IPv6)</p>
17.12	Oct 13, 2017	<p>Pre-Requisite of the following test cases and Annex were updated according to #1475:</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>Annex A.44 Configuring HTTPS if Required</p>
17.12	Oct 19, 2017	<p>The following test cases and Annexes were added according to #1348:</p> <p>MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>MEDIA2 AUDIO STREAMING – G.711 (RTP/RTSP/TCP)</p>

MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/UDP, IPv6)
 MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/
TCP, IPv6)
 MEDIA2 AUDIO STREAMING – G.711 (RTP/RTSP/TCP, IPv6)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/UDP)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/RTSP/HTTP/
TCP)
 MEDIA2 AUDIO STREAMING – AAC (RTP/RTSP/TCP)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/UDP, IPv6)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/RTSP/HTTP/
TCP, IPv6)
 MEDIA2 AUDIO STREAMING – AAC (RTP/RTSP/TCP, IPv6)
 MEDIA2 AUDIO STREAMING – G.711 (RTP-Multicast, IPv4)
 MEDIA2 AUDIO STREAMING – G.711 (RTP-Multicast, IPv6)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Multicast, IPv4)
 MEDIA2 AUDIO STREAMING – AAC (RTP-Multicast, IPv6)
 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/
TCP)
 MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP)
 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP, IPv6)
 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/
TCP, IPv6)
 MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP, IPv6)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/
TCP)
 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP, IPv6)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/
TCP, IPv6)
 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP, IPv6)
 MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast, IPv4)
 MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast, IPv6)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast, IPv4)
 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast, IPv6)
 The following test cases and Annexes were updated according to
#1348:

		<p>MEDIA2_RTSS-5-1-1 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP)</p> <p>Annex A.10 Media Streaming over RTP-Unicast/UDP</p> <p>Annex A.11 Media Streaming over RTP-Unicast/RTSP/HTTP/TCP</p> <p>Annex A.12 Media Streaming over RTP/RTSP/TCP</p> <p>Annex A.13 Media Streaming over RTP-Multicast</p> <p>Annex A.47 Device Configuration for Video and Audio Streaming</p>
17.12	Oct 30, 2017	<p>The following test cases and Annexes were added according to #1501:</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>Annex A.50 Audio Backchannel over RTP-Unicast/RTSP/HTTPS/TCP</p>
17.12	Oct 31, 2017	<p>The following test cases and Annexes were added according to #1497:</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>Annex A.51 Metadata Streaming over RTP-Unicast/RTSP/HTTPS/TCP</p>
17.12	Oct 31, 2017	<p>The following test cases and Annexes were added according to #1498:</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/WebSockets)</p> <p>METADATA STREAMING (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>Annex A.52 Metadata Streaming over WebSocket</p>
17.12	Nov 29, 2017	<p>The following test cases and Annexes were added according to #1409:</p> <p>MEDIA2_RTSS-1-1-23 VIDEO ENCODER INSTANCES</p> <p>MEDIA2_RTSS-1-1-24 VIDEO ENCODER INSTANCES - H.264</p> <p>MEDIA2_RTSS-1-1-25 VIDEO ENCODER INSTANCES - H.265</p> <p>Annex A.53 Remove all non-fixed Media Profiles and remove all configurations from fixed Media Profiles was removed</p> <p>Annex A.55 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration</p> <p>Annex A.54 Get Video Source Configurations List</p> <p>Annex A.56 Concurrent Video Streaming over RTP-Unicast/UDP</p> <p>Annex A.57 Concurrent Video Streaming over RTP-Unicast/UDP</p>

		Annex A.58 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration for Specified Encoding
17.12	Nov 30, 2017	Annex A.41 Device Configuration for Video Streaming was updated according to #1536
18.06	Jan 10, 2018	The following were updated according to #1549: Annex A.37 Removing Video Encoder Configuration and Audio Encoder Configuration from Media Profile was removed Annex A.31 Media2 Service – Media Profile Configuration for Metadata Streaming was updated with (step 11 added). MEDIA2_RTSS-4-2-1 METADATA STREAMING (RTP-Multicast/UDP) was updated (step 4 removed). MEDIA2_RTSS-4-2-2 METADATA STREAMING (RTP-Multicast/UDP, IPv6) was updated (step 5 removed).
18.06	Jan 11, 2018	The following were updated according to #1555: Annex A.39 Get Stream Uri
18.06	Jan 24, 2018	The following were updated according to #1568: MEDIA2_RTSS-2-1-22 MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP) (removed) MEDIA2_RTSS-2-1-23 MEDIA2 AUDIO STREAMING – G.711 (RTP/RTSP/TCP) (removed) MEDIA2_RTSS-2-1-24 MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/UDP, IPv6) (removed) MEDIA2_RTSS-2-1-25 MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (removed) MEDIA2_RTSS-2-1-26 MEDIA2 AUDIO STREAMING – G.711 (RTP/RTSP/TCP, IPv6) (removed) MEDIA2_RTSS-2-1-27 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/UDP) (removed) MEDIA2_RTSS-2-1-28 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP) (removed) MEDIA2_RTSS-2-1-29 MEDIA2 AUDIO STREAMING – AAC (RTP/RTSP/TCP) (removed) MEDIA2_RTSS-2-1-30 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/UDP, IPv6) (removed) MEDIA2_RTSS-2-1-31 MEDIA2 AUDIO STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (removed) MEDIA2_RTSS-2-1-32 MEDIA2 AUDIO STREAMING – AAC (RTP/RTSP/TCP, IPv6) (removed) MEDIA2_RTSS-2-2-5 MEDIA2 AUDIO STREAMING – G.711 (RTP-Multicast, IPv4) (removed) MEDIA2_RTSS-2-2-6 MEDIA2 AUDIO STREAMING – G.711 (RTP-Multicast, IPv6) (removed)

		<p>MEDIA2_RTSS-2-2-7 MEDIA2 AUDIO STREAMING – AAC (RTP-Multicast, IPv4) (removed)</p> <p>MEDIA2_RTSS-2-2-8 MEDIA2 AUDIO STREAMING – AAC (RTP-Multicast, IPv6) (removed)</p>
18.06	Jan 24, 2018	Annex A.6 Media2 Service Profile Configuration for Video Streaming was updated according to #1554.
18.06	Feb 06, 2018	<p>The following were updated according to #1550:</p> <p>Annex A.46 Device Configuration for Video and Audio Streaming (Resolution and FrameRateLimit settings were updated for step 3)</p> <p>Annex A.40 Device Configuration for Video Streaming (Resolution and FrameRateLimit settings were updated for step 3)</p>
18.06	Mar 15, 2018	<p>timeout1 variable was renamed to operationDelay variable</p> <p>The following were updated according to #1562:</p> <p>Annex A.15 Configuring HTTPS using Advanced Security (steps 1-3 were added)</p>
18.06	Mar 15, 2018	<p>The following were updated according to #1586:</p> <p>Annex A.18 Create an RSA key pair (steps 6.1 and 7 were updated)</p>
18.06	Apr 10, 2018	Annex A.48 RTSP Authentication Check was updated according to #1590.
18.06	Apr 17, 2018	<p>The following were updated according to #1615:</p> <p>Annex A.38 Provide CA certificate (step 1 added, step 3 updated)</p> <p>Annex A.58 Determine RSA key length (added)</p>
18.06	Apr 18, 2018	<p>The following were updated according to #1595:</p> <p>Annex A.30 Media Streaming over WebSocket (step 1 added)</p> <p>Annex A.59 Get WebSocket URI (added)</p>
18.06	May 15, 2018	<p>The following were updated according to #1593:</p> <p>Annex A.16 Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair (step 8 added, note added)</p> <p>Annex A.17 Add server certificate assignment with corresponding certification path, CA certificate and RSA key pair (step 12 added, note added)</p>
18.06	Jun 21, 2018	Reformatting document using new template
18.06	Nov 15, 2018	<p>The following were updated according to #1653:</p> <p>Pre-Requisites for all test cases ("Advanced Security Service is received from the DUT, if TLS Server is supported by DUT." was replaced with "Security Configuration Service is received from the DUT, if TLS Server is supported by DUT.")</p> <p>For all test cases Pre-Requisites were updated ("Advanced Security Service is received from the DUT." was replaced with "Security Configuration Service is received from the DUT.")</p>

		Annex A.15 Configuring HTTPS using Advanced Security ("Advanced Security Service" was replaced with "Security Configuration Service" in many places)
18.12	Dec 21, 2018	Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737
18.12	Dec 24, 2018	<p>The following annexes were updated to be consistency with implementation according to #1697:</p> <p>A.22 Audio Backchannel over RTP-Unicast/RTSP/HTTP/TCP (step 9: connection2 was replaced with connection1)</p> <p>A.49 Audio Backchannel over RTP-Unicast/RTSP/HTTPS/TCP (step 9: connection2 was replaced with connection1)</p> <p>The following were added according to #1697:</p> <p>A.60 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTP/TCP</p> <p>A.61 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTPS/TCP</p> <p>BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p>
19.06	Apr 11, 2019	<p>The following annexes were updated according to #1764:</p> <p>A.7 Media2 Service – Media Profile Configuration for Audio Streaming (step 3.3 and step 4.3.5.3 added)</p> <p>A.41 Device Configuration for Audio Streaming (Returns parameters changed, step 3 added)</p> <p>A.20 Media2 Service – Media Profile Configuration for Audio Backchannel Streaming (Returns parameters changed, step 3.7 added, step 4.3.9.3 added)</p> <p>A.47 Media2 Service – Adding AudioSource and AudioEncoder configurations to Media Profile (step 5.5.3 added)</p> <p>A.46 Device Configuration for Video and Audio Streaming (Returns parameters changed, step 6 added)</p>

MEDIA2_RTSS-2-1-7	MEDIA2 STREAMING – AAC (RTP-Unicast/UDP) (step 3 and step 4 changed)
MEDIA2_RTSS-2-1-8	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP) (step 3 and step 4 changed)
MEDIA2_RTSS-2-1-9	MEDIA2 STREAMING – AAC (RTP/RTSP/TCP) (step 3 and step 4 changed)
MEDIA2_RTSS-2-1-10	MEDIA2 STREAMING – AAC (RTP-Unicast/UDP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-2-1-11	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-2-1-12	MEDIA2 STREAMING – AAC (RTP/RTSP/TCP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-2-1-14	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP) (step 4 and step 5 changed)
MEDIA2_RTSS-2-1-16	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 and step 6 changed)
MEDIA2_RTSS-2-1-18	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets) (step 3 and step 4 changed)
MEDIA2_RTSS-2-1-20	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-2-2-3	MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4) (step 3 and step 4 changed)
MEDIA2_RTSS-2-2-4	MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-3-1-8	BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4) (step 3 and step 4 changed)
MEDIA2_RTSS-3-1-7	BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv4) (step 3 and step 4 changed)
MEDIA2_RTSS-3-1-8	BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4) (step 3 and step 4 changed)
MEDIA2_RTSS-3-1-9	BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv4) (step 3 and step 4 changed)
MEDIA2_RTSS-3-1-10	BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-3-1-11	BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-3-1-12	BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-3-1-14	BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets) (step 3 and step 4 changed)
MEDIA2_RTSS-3-1-16	BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets, IPv6) (step 4 and step 5 changed)
MEDIA2_RTSS-3-1-18	BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4) (step 4 and step 5 changed)

		<p>MEDIA2_RTSS-3-1-20 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 and step 6 changed)</p> <p>MEDIA2_RTSS-3-1-23 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4) (step 3 and step 4 changed)</p> <p>MEDIA2_RTSS-3-1-24 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 and step 5 changed)</p> <p>MEDIA2_RTSS-3-1-26 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4) (step 4 and step 5 changed)</p> <p>MEDIA2_RTSS-3-1-28 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 and step 6 changed)</p> <p>MEDIA2_RTSS-5-1-7 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP) (step 3 and step 6 changed, step 5 added)</p> <p>MEDIA2_RTSS-5-1-8 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/TCP) (step 3 and step 6 changed, step 5 added)</p> <p>MEDIA2_RTSS-5-1-9 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP) (step 3 and step 6 changed, step 5 added)</p> <p>MEDIA2_RTSS-5-1-10 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP, IPv6) (step 4 and step 7 changed, step 6 added)</p> <p>MEDIA2_RTSS-5-1-11 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 and step 7 changed, step 6 added)</p> <p>MEDIA2_RTSS-5-1-12 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP, IPv6) (step 4 and step 7 changed, step 6 added)</p> <p>MEDIA2_RTSS-5-2-3 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP) (step 3 and step 6 changed, step 5 added)</p> <p>MEDIA2_RTSS-5-2-4 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP, IPv6) (step 4 and step 7 changed, step 6 added)</p>
19.06	Jun 07, 2019	<p>Note added in the following annexes according to #1869:</p> <ul style="list-style-type: none"> A.10 Media Streaming over RTP-Unicast/UDP A.11 Media Streaming over RTP-Unicast/RTSP/HTTP/TCP A.12 Media Streaming over RTP/RTSP/TCP A.13 Media Streaming over RTP-Multicast A.14 Media Streaming over RTP-Unicast/RTSP/HTTPS/TCP A.21 Audio Backchannel streaming over RTP-Unicast/UDP A.22 Audio Backchannel over RTP-Unicast/RTSP/HTTP/TCP A.23 Audio Backchannel Streaming over RTP/RTSP/TCP A.24 Audio Backchannel Streaming over RTP-Multicast

		<p>A.30 Media Streaming over WebSocket</p> <p>A.42 Backchannel Streaming over WebSocket</p> <p>A.49 Audio Backchannel over RTP-Unicast/RTSP/HTTPS/TCP</p> <p>A.60 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTP/TCP</p> <p>A.61 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTPS/TCP</p>
19.12	Aug 30, 2019	<p>The following annexes were updated according to #1783:</p> <p>Normative references: link to RFC 2326 added, link to IETF RFC 4566 added</p> <p>A.39 Get Stream Uri (Input section updated, step 4 and step 5 added)</p> <p>A.62 Check of IP address type in response to RTSP DESCRIBE (added)</p> <p>MEDIA2_RTSS-3-1-4 BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-5 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-6 BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-10 BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-11 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-12 BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-15 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/WebSockets, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-16 BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-19 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 updated)</p> <p>MEDIA2_RTSS-3-1-20 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 updated)</p> <p>MEDIA2_RTSS-3-1-22 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-24 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 updated)</p> <p>MEDIA2_RTSS-3-1-27 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 updated)</p> <p>MEDIA2_RTSS-3-1-28 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 updated)</p>

MEDIA2_RTSS-4-1-5 METADATA STREAMING (RTP-Unicast/UDP, IPv6) (step 4 updated)

MEDIA2_RTSS-4-1-6 METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP, IPv6) (step 4 updated)

MEDIA2_RTSS-4-1-7 METADATA STREAMING (RTP/RTSP/TCP, IPv6) (step 4 updated)

MEDIA2_RTSS-4-1-9 METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) (step 5 updated)

MEDIA2_RTSS-4-1-11 METADATA STREAMING (RTP-Unicast/RTSP/WebSockets, IPv6) (step 4 updated)

MEDIA2_RTSS-4-2-2 METADATA STREAMING (RTP-Multicast/UDP, IPv6) (step 4 updated)

MEDIA2_RTSS-1-1-4 MEDIA2 SET SYNCHRONIZATION POINT – H.264 (step 6 added)

MEDIA2_RTSS-1-1-11 MEDIA2 SET SYNCHRONIZATION POINT – H.265 (step 6 added)

MEDIA2_RTSS-4-1-4 METADATA STREAMING - SET SYNCHRONIZATION POINT (step 6 added)

A.10 Media Streaming over RTP-Unicast/UDP (step 5 added)

A.11 Media Streaming over RTP-Unicast/RTSP/HTTP/TCP (step 7 added)

A.12 Media Streaming over RTP/RTSP/TCP (step 5 added)

A.13 Media Streaming over RTP-Multicast (step 5 added)

A.14 Media Streaming over RTP-Unicast/RTSP/HTTPS/TCP (step 6 added)

A.21 Audio Backchannel streaming over RTP-Unicast/UDP (step 4 added)

A.22 Audio Backchannel over RTP-Unicast/RTSP/HTTP/TCP (step 6 added)

A.23 Audio Backchannel Streaming over RTP/RTSP/TCP (step 4 added)

A.24 Audio Backchannel Streaming over RTP-Multicast (step 4 added)

A.30 Media Streaming over WebSocket (step 6 added)

A.34 Metadata Streaming over RTP-Unicast/UDP (step 4 added)

A.35 Metadata Streaming over RTP-Unicast/RTSP/HTTP/TCP (step 6 added)

A.36 Metadata Streaming over RTP/RTSP/TCP (step 4 added)

A.37 Metadata Streaming over RTP-Unicast/UDP (step 4 added)

A.42 Backchannel Streaming over WebSocket (step 8 added)

		<p>A.49 Audio Backchannel over RTP-Unicast/RTSP/HTTPS/TCP (step 6 added)</p> <p>A.50 Metadata Streaming over RTP-Unicast/RTSP/HTTPS/TCP (step 6 added)</p> <p>A.51 Metadata Streaming over WebSocket (step 8 added)</p> <p>A.55 Concurrent Video Streaming over RTP-Unicast/UDP (step 1.4 added)</p> <p>A.60 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTP/TCP (step 6 added)</p> <p>A.61 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTPS/TCP (step 6 added)</p> <p>A.40 Device Configuration for Video Streaming (step 5 updated)</p> <p>A.41 Device Configuration for Audio Streaming (step 5 updated)</p> <p>A.45 Device Configuration for Audio Streaming using Media Profile that contains only Audio Configurations (step 6 updated)</p> <p>A.46 Device Configuration for Video and Audio Streaming (step 10 updated)</p> <p>A.20 Media2 Service – Media Profile Configuration for Audio Backchannel Streaming (Input updated, step 7 updated)</p> <p>A.31 Media2 Service – Media Profile Configuration for Metadata Streaming (Input updated, step 8 and step 12 updated)</p>
19.12	Oct 08, 2019	<p>Supporting of Metadata feature was added into Pre-Requisite of the following test cases according to #1894:</p> <p>MEDIA2_RTSS-4-1-1 METADATA STREAMING (RTP-Unicast/UDP)</p> <p>MEDIA2_RTSS-4-1-2 METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP)</p> <p>MEDIA2_RTSS-4-1-3 METADATA STREAMING (RTP/RTSP/TCP)</p> <p>MEDIA2_RTSS-4-1-4 METADATA STREAMING - SET SYNCHRONIZATION POINT</p> <p>MEDIA2_RTSS-4-1-5 METADATA STREAMING (RTP-Unicast/UDP, IPv6)</p> <p>MEDIA2_RTSS-4-1-6 METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2_RTSS-4-1-7 METADATA STREAMING (RTP/RTSP/TCP, IPv6)</p> <p>MEDIA2_RTSS-4-1-8 METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP)</p> <p>MEDIA2_RTSS-4-1-9 METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2_RTSS-4-1-10 METADATA STREAMING (RTP-Unicast/RTSP/WebSockets)</p>

		<p>MEDIA2_RTSS-4-1-11 METADATA STREAMING (RTP-Unicast/RTSP/WebSockets, IPv6)</p> <p>MEDIA2_RTSS-4-2-1 METADATA STREAMING (RTP-Multicast/UDP)</p> <p>MEDIA2_RTSS-4-2-2 METADATA STREAMING (RTP-Multicast/UDP, IPv6)</p>
19.12	Nov 21, 2019	<p>The following annexes were updated according to #1971:</p> <p>A.54 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration (Profile token parameter added into GetVideoEncoderConfigurationOptions request)</p> <p>A.57 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration for Specified Encoding (Profile token parameter added into GetVideoEncoderConfigurationOptions request)</p>
20.12	Jul 31, 2020	<p>The following annexes were changed according to #2072:</p> <p>A.46 Device Configuration for Video and Audio Streaming</p> <p>A.47 Media2 Service – Adding AudioSource and AudioEncoder configurations to Media Profile</p>
20.12	Sep 23, 2020	<p>The following were added according to #2082:</p> <p>MEDIA2_RTSS-6-1-1 MEDIA2 STREAMING – G.711 BACKCHANNEL AND H.26X VIDEO AND G.711/AAC AUDIO (RTP-Unicast/UDP)</p> <p>A.65 Device Configuration for Audio Backchannel and Video and Audio Streaming</p> <p>A.66 Media2 Service – Adding VideoSource, VideoEncoder, AudioSource and AudioEncoder configurations to Media Profile</p> <p>A.67 Media2 Service – Adding AudioSource and AudioEncoder to Media Profile</p> <p>A.68 Audio Backchannel and Media Streaming over RTP-Unicast/UDP</p>
20.12	Sep 23, 2020	<p>Annex A.47 was renamed in the scope of #2082:</p> <p>Old name: Media2 Service – Adding AudioSource and AudioEncoder configurations to Media Profile</p> <p>New name: Media2 Service – Adding AudioSource and AudioEncoder with Specified Audio Encoder Value to Media Profile</p>
20.12	Nov 17, 2020	<p>Annex A.48 RTSP Authentication Check was updated according to #2073</p>
21.06	Apr 09, 2021	<p>The following annexes were updated according to #2124:</p> <p>A.23 Turn on IPv6 network interface</p> <p>A.24 Restore Network Settings</p>
21.06	May 26, 2021	<p>The following test cases and annexes were removed according to #2215:</p>

	<p>MEDIA2_RTSS-3-1-2 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>MEDIA2_RTSS-3-1-5 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2_RTSS-3-1-8 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4)</p> <p>MEDIA2_RTSS-3-1-11 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)</p> <p>MEDIA2_RTSS-3-1-17 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>MEDIA2_RTSS-3-1-18 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)</p> <p>MEDIA2_RTSS-3-1-19 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>MEDIA2_RTSS-3-1-20 BACKCHANNEL – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)</p> <p>A.22 Audio Backchannel over RTP-Unicast/RTSP/HTTP/TCP</p> <p>A.49 Audio Backchannel over RTP-Unicast/RTSP/HTTPS/TCP</p>
--	---

Table of Contents

1	Introduction	30
1.1	Scope	30
1.2	Real Time Streaming	31
2	Normative references	32
3	Terms and Definitions	34
3.1	Conventions	34
3.2	Definitions	34
3.3	Abbreviations	34
4	Test Overview	35
4.1	Test Setup	35
4.1.1	Network Configuration for DUT	35
4.2	Prerequisites	36
4.3	Test Policy	36
4.3.1	Real Time Streaming	36
5	Real Time Streaming Test Cases	38
5.1	Video Streaming	38
5.1.1	Unicast	38
5.1.1.1	MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP)	38
5.1.1.2	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP)	39
5.1.1.3	MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP)	40
5.1.1.4	MEDIA2 SET SYNCHRONIZATION POINT – H.264	42
5.1.1.5	MEDIA2 STREAMING – H.264 (RTP-Unicast/UDP, IPv6)	44
5.1.1.6	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)	46
5.1.1.7	MEDIA2 STREAMING – H.264 (RTP/RTSP/TCP, IPv6)	47
5.1.1.8	MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP)	49
5.1.1.9	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP)	50
5.1.1.10	MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP)	51
5.1.1.11	MEDIA2 SET SYNCHRONIZATION POINT – H.265	52
5.1.1.12	MEDIA2 STREAMING – H.265 (RTP-Unicast/UDP, IPv6)	55

5.1.1.13	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)	56
5.1.1.14	MEDIA2 STREAMING – H.265 (RTP/RTSP/TCP, IPv6)	58
5.1.1.15	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP)	59
5.1.1.16	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP)	61
5.1.1.17	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)	62
5.1.1.18	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)	64
5.1.1.19	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/WebSockets)	66
5.1.1.20	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/WebSockets)	67
5.1.1.21	MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/WebSockets, IPv6)	68
5.1.1.22	MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/WebSockets, IPv6)	70
5.1.1.23	VIDEO ENCODER INSTANCES	71
5.1.1.24	VIDEO ENCODER INSTANCES - H.264	74
5.1.1.25	VIDEO ENCODER INSTANCES - H.265	76
5.1.2	Multicast	79
5.1.2.1	MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv4)	79
5.1.2.2	MEDIA2 STREAMING – H.264 (RTP-Multicast, IPv6)	80
5.1.2.3	MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv4)	82
5.1.2.4	MEDIA2 STREAMING – H.265 (RTP-Multicast, IPv6)	83
5.2	Audio Streaming	85
5.2.1	Unicast	85
5.2.1.1	MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP)	85
5.2.1.2	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP)	86

5.2.1.3	MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP)	87
5.2.1.4	MEDIA2 STREAMING – G.711 (RTP-Unicast/UDP, IPv6)	89
5.2.1.5	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)	90
5.2.1.6	MEDIA2 STREAMING – G.711 (RTP/RTSP/TCP, IPv6)	92
5.2.1.7	MEDIA2 STREAMING – AAC (RTP-Unicast/UDP)	93
5.2.1.8	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP)	94
5.2.1.9	MEDIA2 STREAMING – AAC (RTP/RTSP/TCP)	96
5.2.1.10	MEDIA2 STREAMING – AAC (RTP-Unicast/UDP, IPv6)	97
5.2.1.11	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)	98
5.2.1.12	MEDIA2 STREAMING – AAC (RTP/RTSP/TCP, IPv6)	100
5.2.1.13	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/ TCP)	101
5.2.1.14	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP) ..	103
5.2.1.15	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)	104
5.2.1.16	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)	106
5.2.1.17	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/ WebSockets)	108
5.2.1.18	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets) ..	109
5.2.1.19	MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/WebSockets, IPv6)	110
5.2.1.20	MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets, IPv6)	112
5.2.1.21	MEDIA2 AUDIO STREAMING – G.711 (RTP-Unicast/UDP)	113
5.2.2	Multicast	115
5.2.2.1	MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv4)	115
5.2.2.2	MEDIA2 STREAMING – G.711 (RTP-Multicast, IPv6)	116
5.2.2.3	MEDIA2 STREAMING – AAC (RTP-Multicast, IPv4)	117

5.2.2.4	MEDIA2 STREAMING – AAC (RTP-Multicast, IPv6)	119
5.3	Audio Backchannel	120
5.3.1	Unicast	120
5.3.1.1	BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv4)	120
5.3.1.2	BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv4)	122
5.3.1.3	BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv6)	123
5.3.1.4	BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv6)	125
5.3.1.5	BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv4)	126
5.3.1.6	BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv4)	127
5.3.1.7	BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv6)	129
5.3.1.8	BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv6)	130
5.3.1.9	BACKCHANNEL – G.711 (RTP-Unicast/RTSP/WebSockets)	132
5.3.1.10	BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets)	133
5.3.1.11	BACKCHANNEL – G.711 (RTP-Unicast/RTSP/WebSockets, IPv6)	135
5.3.1.12	BACKCHANNEL – AAC (RTP-Unicast/RTSP/WebSockets, IPv6) .	136
5.3.1.13	BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/ RTSP/HTTP/TCP, IPv4)	138
5.3.1.14	BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/ RTSP/HTTP/TCP, IPv6)	139
5.3.1.15	BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/ RTSP/HTTP/TCP, IPv4)	141
5.3.1.16	BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/ RTSP/HTTP/TCP, IPv6)	142
5.3.1.17	BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/ RTSP/HTTPS/TCP, IPv4)	144
5.3.1.18	BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/ RTSP/HTTPS/TCP, IPv4)	146
5.3.1.19	BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/ RTSP/HTTPS/TCP, IPv6)	147

5.3.1.20	BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/ RTSP/HTTPS/TCP, IPv6)	149
5.4	Metadata Streaming	151
5.4.1	Unicast	151
5.4.1.1	METADATA STREAMING (RTP-Unicast/UDP)	151
5.4.1.2	METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP)	152
5.4.1.3	METADATA STREAMING (RTP/RTSP/TCP)	154
5.4.1.4	METADATA STREAMING - SET SYNCHRONIZATION POINT	155
5.4.1.5	METADATA STREAMING (RTP-Unicast/UDP, IPv6)	157
5.4.1.6	METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP, IPv6) ...	159
5.4.1.7	METADATA STREAMING (RTP/RTSP/TCP, IPv6)	160
5.4.1.8	METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP)	162
5.4.1.9	METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP, IPv6) .	163
5.4.1.10	METADATA STREAMING (RTP-Unicast/RTSP/WebSockets)	165
5.4.1.11	METADATA STREAMING (RTP-Unicast/RTSP/WebSockets, IPv6)	166
5.4.2	Multicast	168
5.4.2.1	METADATA STREAMING (RTP-Multicast/UDP)	168
5.4.2.2	METADATA STREAMING (RTP-Multicast/UDP, IPv6)	169
5.5	Audio & Video Streaming	171
5.5.1	Unicast	171
5.5.1.1	MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP)	171
5.5.1.2	MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/ TCP)	172
5.5.1.3	MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP)	174
5.5.1.4	MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP, IPv6)	175
5.5.1.5	MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/ TCP, IPv6)	177
5.5.1.6	MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP, IPv6)	179
5.5.1.7	MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP)	180

5.5.1.8	MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/ TCP)	182
5.5.1.9	MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP)	183
5.5.1.10	MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP, IPv6)	185
5.5.1.11	MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/ TCP, IPv6)	186
5.5.1.12	MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP, IPv6)	188
5.5.2	Multicast	190
5.5.2.1	MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast/UDP)	190
5.5.2.2	MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast/UDP, IPv6) ...	191
5.5.2.3	MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP)	193
5.5.2.4	MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP, IPv6)	194
5.6	Audio Backchannel & Video & Audio Streaming	196
5.6.1	Unicast	196
5.6.1.1	MEDIA2 STREAMING – G.711 BACKCHANNEL AND H.26X VIDEO AND G.711/AAC AUDIO (RTP-Unicast/UDP)	196
A	Helper Procedures and Additional Notes	199
A.1	Invalid RTP Header	199
A.2	I-frame insertion time interval	199
A.3	Name and Token Parameters	199
A.4	Turn on IPv6 network interface	199
A.5	Restore Network Settings	201
A.6	Media2 Service Profile Configuration for Video Streaming	202
A.7	Media2 Service – Media Profile Configuration for Audio Streaming	207
A.8	Removing Video Encoder Configuration and Metadata Configuration from Media Profile	211
A.9	Removing Audio Encoder Configuration and Metadata Configuration from Media Profile	212
A.10	Media Streaming over RTP-Unicast/UDP	213
A.11	Media Streaming over RTP-Unicast/RTSP/HTTP/TCP	216
A.12	Media Streaming over RTP/RTSP/TCP	218

A.13	Media Streaming over RTP-Multicast	221
A.14	Media Streaming over RTP-Unicast/RTSP/HTTPS/TCP	223
A.15	Configuring HTTPS using Security Configuration Service	225
A.16	Add server certificate assignment with corresponding certification path, self-signed certificate and RSA key pair	227
A.17	Add server certificate assignment with corresponding certification path, CA certificate and RSA key pair	228
A.18	Create an RSA key pair	231
A.19	Subject for a server certificate	232
A.20	Media2 Service – Media Profile Configuration for Audio Backchannel Streaming	232
A.21	Audio Backchannel streaming over RTP-Unicast/UDP	238
A.22	Audio Backchannel Streaming over RTP/RTSP/TCP	240
A.23	Audio Backchannel Streaming over RTP-Multicast	242
A.24	Removing Video Encoder Configuration, Audio Encoder Configuration and Metadata Configuration from Media Profile	244
A.25	Get Media2 Service Capabilities	246
A.26	Web Socket Handshake	246
A.27	Sec-WebSocket-Key value generation	248
A.28	Basic TLS handshake	249
A.29	Media Streaming over WebSocket	251
A.30	Media2 Service – Media Profile Configuration for Metadata Streaming	253
A.31	Media2 Service – Add PTZ Configuration to Media Profile	258
A.32	Media2 Service – Add Analytics Configuration to Media Profile	259
A.33	Metadata Streaming over RTP-Unicast/UDP	261
A.34	Metadata Streaming over RTP-Unicast/RTSP/HTTP/TCP	263
A.35	Metadata Streaming over RTP/RTSP/TCP	266
A.36	Metadata Streaming over RTP-Unicast/UDP	269
A.37	Provide CA certificate	272
A.38	Get Stream Uri	273
A.39	Device Configuration for Video Streaming	274

A.40	Device Configuration for Audio Streaming	277
A.41	Backchannel Streaming over WebSocket	279
A.42	Configuring HTTPS if Required	282
A.43	Removing Configurations from Media Profile	282
A.44	Device Configuration for Audio Streaming using Media Profile that contains only Audio Configurations	285
A.45	Device Configuration for Video and Audio Streaming	287
A.46	Media2 Service – Adding AudioSource and AudioEncoder with Specified Audio Encoder Value to Media Profile	293
A.47	RTSP Authentication Check	296
A.48	Metadata Streaming over RTP-Unicast/RTSP/HTTPS/TCP	297
A.49	Metadata Streaming over WebSocket	300
A.50	Remove all non-fixed Media Profiles and remove all configurations from fixed Media Profiles	303
A.51	Get Video Source Configurations List	304
A.52	Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration	304
A.53	Concurrent Video Streaming over RTP-Unicast/UDP	308
A.54	Concurrent Video Streaming over RTP-Unicast/UDP	310
A.55	Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration for Specified Encoding	310
A.56	Determine RSA key length	314
A.57	Get WebSocket URI	314
A.58	Audio Backchannel by POST over RTP-Unicast/RTSP/HTTP/TCP	315
A.59	Audio Backchannel by POST over RTP-Unicast/RTSP/HTTPS/TCP	317
A.60	Check of IP address type in response to RTSP DESCRIBE	319
A.61	Set Video Encoder Configuration for Streaming	321
A.62	Set Audio Encoder Configuration for Streaming	323
A.63	Device Configuration for Audio Backchannel and Video and Audio Streaming	324
A.64	Media2 Service – Adding VideoSource, VideoEncoder, AudioSource and AudioEncoder configurations to Media Profile	329

A.65 Media2 Service – Adding AudioSource and AudioEncoder to Media Profile 333

A.66 Audio Backchannel and Media Streaming over RTP-Unicast/UDP 334

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 Real-time 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 Real-time Streaming and some of the optional requirements.

This ONVIF Real Time Streaming using Media2 Device Test Specification covers ONVIF Media2 Service and Real-time 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, Audio backchannel streams, and Metadata 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] ONVIF Conformance Process Specification:
<https://www.onvif.org/profiles/conformance/>
- [ONVIF Profile Policy] ONVIF Profile Policy:
<https://www.onvif.org/profiles/>
- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:
<https://www.onvif.org/profiles/specifications/>
- [ONVIF Core Specs] ONVIF Core Specification:
<https://www.onvif.org/profiles/specifications/>
- [ONVIF Media2 Spec] ONVIF Media 2 Service Specification:
<https://www.onvif.org/profiles/specifications/>
- [ONVIF Streaming Spec] ONVIF Streaming Specification:
<https://www.onvif.org/profiles/specifications/>
- [ONVIF Base Test] ONVIF Base Device Test Specification:
<https://www.onvif.org/profiles/conformance/device-test/>
- [ISO/IEC Directives, Part 2] ISO/IEC Directives, Part 2, Annex H:
<http://www.iso.org/directives>
- [ISO 16484-5] ISO 16484-5:2014-09 Annex P:
<https://www.iso.org/obp/ui/#!iso:std:63753:en>
- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:
<http://www.w3.org/TR/soap12-part1/>
- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:
<http://www.w3.org/TR/xmlschema-1/>
- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:
<http://www.w3.org/TR/xmlschema-2/>

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

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

- IETF RFC 4566, SDP: Session Description Protocol

<http://www.ietf.org/rfc/rfc4566.txt>

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

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

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

This section 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.

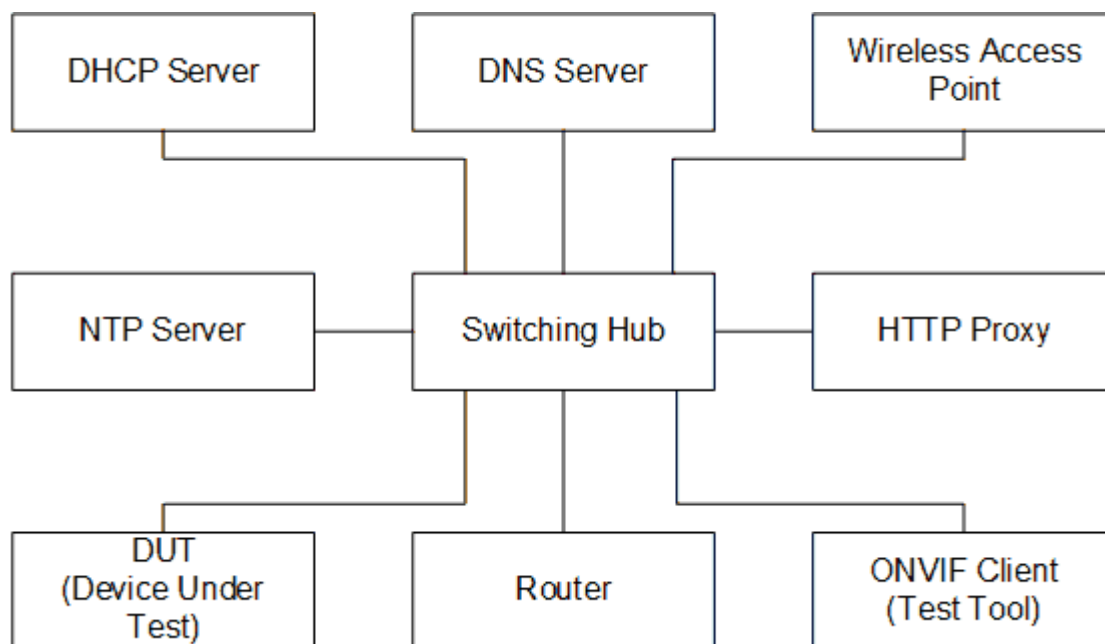
4.1 Test Setup

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 4.1).

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

Figure 4.1. Test Configuration for DUT



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

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

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

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

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

DHCP Server: provides IPv4 Address to the connected devices.

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

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

Router: provides router advertisements for IPv6 configuration.

4.2 Prerequisites

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

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

4.3 Test Policy

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

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

4.3.1 Real Time Streaming

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 Case ID: MEDIA2_RTSS-1-1-1

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

Feature Under Test: Streaming over RTP-Unicast/UDP, H.264

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.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-2

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, H.264

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-1-3

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

Feature Under Test: Streaming over RTP/RTSP/TCP, H.264

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.1.1.4 MEDIA2 SET SYNCHRONIZATION POINT – H.264

Test Case ID: MEDIA2_RTSS-1-1-4

Specification Coverage: Set synchronization point.

Feature Under Test: SetSynchronizationPoint, H.264

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
5. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE

- in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. 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*
 8. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 10. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 11. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
 12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 13. ONVIF Client invokes **SetSynchronizationPoint** request with parameters
 - ProfileToken := *profile.@token*
 14. The DUT responds with **SetSynchronizationPointResponse** message.
 15. If DUT does not send I-frame before the regular 'I-frame insertion time interval', FAIL the test and skip other steps.
 16. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 17. The DUT responds with **200 OK** message with parameters
 - Session

18. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **200 OK** message for **RTSP DESCRIBE**.
- DUT did not send **200 OK** message for **RTSP SETUP**.
- DUT did not send **200 OK** message for **RTSP PLAY**.
- DUT did not send **200 OK** message for **RTSP TEARDOWN**.
- DUT did not send **SetSynchronizationPointResponse** 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 Case ID: MEDIA2_RTSS-1-1-5

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

Feature Under Test: Streaming over RTP-Unicast/UDP, H.264, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-6

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, H.264, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtpOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
 7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-7

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

Feature Under Test: Streaming over RTP/RTSP/TCP, H.264, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-1-8

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

Feature Under Test: Streaming over RTP-Unicast/UDP, H.265

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-9

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, H.265

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding
 - in RtpOverHttp - Transport Protocol
 - in IPv4 - IP version

- out *streamUri* - Uri for media streaming
4. 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
 5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-1-10

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

Feature Under Test: Streaming over RTP/RTSP/TCP, H.265

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.1.1.11 MEDIA2 SET SYNCHRONIZATION POINT – H.265

Test Case ID: MEDIA2_RTSS-1-1-11

Specification Coverage: Set synchronization point.

Feature Under Test: SetSynchronizationPoint, H.265

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
5. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. 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*

8. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
11. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **SetSynchronizationPoint** request with parameters
 - ProfileToken := *profile.@token*
14. The DUT responds with **SetSynchronizationPointResponse** message.
15. If DUT does not send I-frame before the regular 'I-frame insertion time interval', FAIL the test and skip other steps.
16. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
17. The DUT responds with **200 OK** message with parameters
 - Session
18. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **200 OK** message for **RTSP DESCRIBE**.
- DUT did not send **200 OK** message for **RTSP SETUP**.
- DUT did not send **200 OK** message for **RTSP PLAY**.
- DUT did not send **200 OK** message for **RTSP TEARDOWN**.
- DUT did not send **SetSynchronizationPointResponse** 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 Case ID: MEDIA2_RTSS-1-1-12

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

Feature Under Test: Streaming over RTP-Unicast/UDP, H.265, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters

- in H265 - required video encoding
 - in RtspUnicast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-13

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, H.265, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-14

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

Feature Under Test: Streaming over RTP/RTSP/TCP, H.265, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters

- in H265 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-15

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, H.264

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-16

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, H.265

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters

- in H265 - required video encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-17

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, H.264, IPv6

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtpOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
6. 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
7. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 5.

8. ONVIF Client restores HTTPS settings which was changed at step 4.
9. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-1-1-18

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, H.265, IPv6

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
6. 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
7. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step [5](#).
8. ONVIF Client restores HTTPS settings which was changed at step [4](#).
9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.1.1.19 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/ WebSockets)

Test Case ID: MEDIA2_RTSS-1-1-19

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket, H.264

WSDL Reference: None

Test Purpose: To verify H.264 media streaming over WebSocket.

Pre-Requisite: Media2 Service is received from the DUT. H.264 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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in video - media type

- in H.264 - expected media stream encoding
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.1.1.20 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/WebSockets)

Test Case ID: MEDIA2_RTSS-1-1-20

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket, H.265

WSDL Reference: None

Test Purpose: To verify H.265 media streaming over WebSocket.

Pre-Requisite: Media2 Service is received from the DUT. H.265 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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding

- in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in video - media type
 - in H.265 - expected media stream encoding
 5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.1.1.21 MEDIA2 STREAMING – H.264 (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-1-1-21

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket, H.264, IPv6

WSDL Reference: None

Test Purpose: To verify H.264 media streaming over WebSocket 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.

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in video - media type
 - in H.264 - expected media stream encoding
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

5.1.1.22 MEDIA2 STREAMING – H.265 (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-1-1-22

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket, H.265, IPv6

WSDL Reference: None

Test Purpose: To verify H.265 media streaming over WebSocket 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.

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H265 - required video encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming

- in video - media type
 - in H.264 - expected media stream encoding
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
 7. 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 does not pass all assertions.

5.1.1.23 VIDEO ENCODER INSTANCES

Test Case ID: MEDIA2_RTSS-1-1-23

Specification coverage: Get video encoder instance information (Media2), Media Profile Management (Profile T), RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP

Feature under test: GetVideoEncoderInstances, Streaming over RTP-Unicast/UDP

WSDL Reference: media2.wsdl

Test Purpose: To verify that for each video source configuration DUT supports creation of as many Media Profiles and concurrent video streams as the number of instances, which is returned by GetVideoEncoderInstances for that video source configuration token.

Pre-Requisite: Media2 Service is received from the DUT. Real-time streaming is supported by DUT. Profile T is supported by DUT as indicated by receiving the GetScopesResponse.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := All
4. DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList1*
5. ONVIF Client retrieves Media2 Service Capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap1* - Media2 Service Capabilities
6. ONVIF Client retrieves Video Source Configurations list by following the procedure mentioned in [Annex A.51](#) with the following input and output parameters
 - out *videoSourceConfList1* - Video Source Configurations list
7. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:
 - 7.1. ONVIF Client invokes **GetVideoEncoderInstances** request with parameters
 - ConfigurationToken := *videoSourceConfig1.token*
 - 7.2. DUT responds with **GetVideoEncoderInstancesResponse** message with parameters
 - Info = *info1*
 - 7.3. Set *infoList1[videoSourceConfig1.@token]* := *info1*.
8. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken - skipped
 - ProfileToken - skipped
9. The DUT responds with all video encoder configurations in **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList1*
10. Set *numberOfProfilesToBeCreated1* := sum of all Total values from *infoList1* list.
11. Set *numberOfFixedProfiles1* := number of items at *profileList1* list with @fixed = true.

12. Set *numberOfVEC1* := number of items at *videoEncoderConfList1* list.
13. If *numberOfProfilesToBeCreated1* > *cap1.ProfileCapabilities.MaximumNumberOfProfiles - numberOfFixedProfiles1*, FAIL the test and skip other steps.
14. If *numberOfProfilesToBeCreated1* > *numberOfVEC1*, FAIL the test and skip other steps.
15. ONVIF Client removes all non-fixed Media Profiles and removes all configurations from fixed Media Profiles by following the procedure mentioned in [Annex A.50](#) with the following input and output parameters
 - in *profileList1* - Media Profiles List
16. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:
 - 16.1. ONVIF Client tries to create new Media Profiles to get number of profiles equal to *info.Total* by following the procedure mentioned in [Annex A.52](#) with the following input and output parameters
 - in *videoSourceConfig1* - Video Source Configuration
 - in *infoList1[videoSourceConfig1.@token]* - information about guaranteed Encoder instances for Video Source Configuration
 - out *configuredProfilesList1* - list of configured Media Profiles for Video Source Configuration
 - 16.2. If number of Media Profiles with *@token = videoSourceConfig1.@token* in *configuredProfilesList1* < *infoList1[videoSourceConfig1.@token].Total*, then FAIL the test and go to step [18](#).
17. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP for each configured media profile by following the procedure mentioned in [Annex A.53](#) with the following input and output parameters
 - in *configuredProfilesList1* - Media Profiles list
18. ONVIF Client restores Media Profiles list if it was changed at steps [15](#), [16.1](#).

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

5.1.1.24 VIDEO ENCODER INSTANCES - H.264

Test Case ID: MEDIA2_RTSS-1-1-24

Specification coverage: Get video encoder instance information (Media2), Media Profile Management (Profile T), RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP

Feature under test: GetVideoEncoderInstances, Streaming over RTP-Unicast/UDP, H.264

WSDL Reference: media2.wsdl

Test Purpose: To verify that for each video source configuration DUT supports creation of as many Media Profiles and concurrent H.264 video streams as the number of instances for H.264, which is returned by GetVideoEncoderInstances for that video source configuration token.

Pre-Requisite: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. Profile T is supported by DUT as indicated by receiving the GetScopesResponse.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := All
4. DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList1*
5. ONVIF Client retrieves Media2 Service Capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap1* - Media2 Service Capabilities
6. ONVIF Client retrieves Video Source Configurations list by following the procedure mentioned in [Annex A.51](#) with the following input and output parameters

- out *videoSourceConfList1* - Video Source Configurations list
7. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:
 - 7.1. ONVIF Client invokes **GetVideoEncoderInstances** request with parameters
 - ConfigurationToken := *videoSourceConfig1.token*
 - 7.2. DUT responds with **GetVideoEncoderInstancesResponse** message with parameters
 - Info = *info1*
 - 7.3. Set *infoList1[videoSourceConfig1.@token]* := *info1*.
 8. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken - skipped
 - ProfileToken - skipped
 9. The DUT responds with all video encoder configurations in **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList1*
 10. Set *numberOfProfilesToBeCreated1* := sum of limitations for H.264 encoding for each item in the *infoList1* list which calculates by following the procedure mentioned in [Annex A.54](#).
 11. Set *numberOfFixedProfiles1* := number of items at *profileList1* list with @fixed = true.
 12. Set *numberOfVEC1* := number of items at *videoEncoderConfList1* list.
 13. If *numberOfProfilesToBeCreated1* > *cap1.ProfileCapabilities.MaximumNumberOfProfiles* - *numberOfFixedProfiles1*, FAIL the test and skip other steps.
 14. If *numberOfProfilesToBeCreated1* > *numberOfVEC1*, FAIL the test and skip other steps.
 15. ONVIF Client removes all non-fixed Media Profiles and removes all configurations from fixed Media Profiles by following the procedure mentioned in [Annex A.50](#) with the following input and output parameters
 - in *profileList1* - Media Profiles List
 16. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:

16.1. ONVIF Client tries to create new Media Profiles to get number of profiles equal to limitation for specified encoder by following the procedure mentioned in [Annex A.55](#) with the following input and output parameters

- in *videoSourceConfig1* - Video Source Configuration
- in *infoList1[videoSourceConfig1.@token]* - information about guaranteed Encoder instances for Video Source Configuration
- in H264 - encoding
- out *configuredProfilesList1* - list of configured Media Profiles for Video Source Configuration

16.2. If number of Media Profiles with @token = *videoSourceConfig1.@token* in *configuredProfilesList1* < limitation for H.264 encoding from *infoList1[videoSourceConfig1.@token]* which calculates by following the procedure mentioned in [Annex A.54](#), then FAIL the test and go to step 18.

17. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP for each configured media profile by following the procedure mentioned in [Annex A.53](#) with the following input and output parameters

- in *configuredProfilesList1* - Media Profiles list

18. ONVIF Client restores Media Profiles list if it was changed at steps 15, 16.1.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.1.25 VIDEO ENCODER INSTANCES - H.265

Test Case ID: MEDIA2_RTSS-1-1-25

Specification coverage: Get video encoder instance information (Media2), Media Profile Management (Profile T), RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP

Feature under test: GetVideoEncoderInstances, Streaming over RTP-Unicast/UDP, H.265

WSDL Reference: media2.wsdl

Test Purpose: To verify that for each video source configuration DUT supports creation of as many Media Profiles and concurrent H.265 video streams as the number of instances for H.264, which is returned by GetVideoEncoderInstances for that video source configuration token.

Pre-Requirement: Media2 Service is received from the DUT. H.264 encoding is supported by DUT. Real-time streaming is supported by DUT. Profile T is supported by DUT as indicated by receiving the GetScopesResponse.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := All
4. DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList1*
5. ONVIF Client retrieves Media2 Service Capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap1* - Media2 Service Capabilities
6. ONVIF Client retrieves Video Source Configurations list by following the procedure mentioned in [Annex A.51](#) with the following input and output parameters
 - out *videoSourceConfList1* - Video Source Configurations list
7. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:
 - 7.1. ONVIF Client invokes **GetVideoEncoderInstances** request with parameters
 - ConfigurationToken := *videoSourceConfig1.token*
 - 7.2. DUT responds with **GetVideoEncoderInstancesResponse** message with parameters

- Info = *info1*
- 7.3. Set *infoList1[videoSourceConfig1.@token]* := *info1*.
8. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
- ConfigurationToken - skipped
 - ProfileToken - skipped
9. The DUT responds with all video encoder configurations in **GetVideoEncoderConfigurationsResponse** with parameters
- Configurations list =: *videoEncoderConfList1*
10. Set *numberOfProfilesToBeCreated1* := sum of limitations for H.265 encoding for each item in the *infoList1* list which calculates by following the procedure mentioned in [Annex A.54](#).
11. Set *numberOfFixedProfiles1* := number of items at *profileList1* list with @fixed = true.
12. Set *numberOfVEC1* := number of items at *videoEncoderConfList1* list.
13. If *numberOfProfilesToBeCreated1* > *cap1.ProfileCapabilities.MaximumNumberOfProfiles - numberOfFixedProfiles1*, FAIL the test and skip other steps.
14. If *numberOfProfilesToBeCreated1* > *numberOfVEC1*, FAIL the test and skip other steps.
15. ONVIF Client removes all non-fixed Media Profiles and removes all configurations from fixed Media Profiles by following the procedure mentioned in [Annex A.50](#) with the following input and output parameters
- in *profileList1* - Media Profiles List
16. For each Video Source Configuration *videoSourceConfig1* from *videoSourceConfList1* repeat the following steps:
- 16.1. ONVIF Client tries to create new Media Profiles to get number of profiles equal to limitation for specified encoder by following the procedure mentioned in [Annex A.55](#) with the following input and output parameters
- in *videoSourceConfig1* - Video Source Configuration
 - in *infoList1[videoSourceConfig1.@token]* - information about guaranteed Encoder instances for Video Source Configuration
 - in H265 - encoding

- out *configuredProfilesList1* - list of configured Media Profiles for Video Source Configuration

16.2. If number of Media Profiles with @token = *videoSourceConfig1.@token* in *configuredProfilesList1* < limitation for H.265 encoding from *infoList1[videoSourceConfig1.@token]* which calculates by following the procedure mentioned in [Annex A.54](#), then FAIL the test and go to step 18.

17. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP for each configured media profile by following the procedure mentioned in [Annex A.53](#) with the following input and output parameters

- in *configuredProfilesList1* - Media Profiles list

18. ONVIF Client restores Media Profiles list if it was changed at steps 15, 16.1.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

5.1.2 Multicast

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

Test Case ID: MEDIA2_RTSS-1-2-1

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

Feature Under Test: Streaming over RTP-Multicast, H.264

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-2-2

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

Feature Under Test: Streaming over RTP-Multicast, H.264, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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

6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-2-3

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

Feature Under Test: Streaming over RTP-Multicast, H.265

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters

- in H265 - required video encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
 5. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-1-2-4

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

Feature Under Test: Streaming over RTP-Multicast, H.265, IPv6

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 configures a media profile and retrieves a stream uri for video streaming by following the procedure mentioned in [Annex A.39](#) with the following input and output parameters
 - in H264 - required video encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Video Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-1

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

Feature Under Test: Streaming over RTP-Unicast/UDP, G.711

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version

- out *streamUri* - Uri for media streaming
4. 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
 5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step [3](#).

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-2

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, G.711

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-1-3

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

Feature Under Test: Streaming over RTP/RTSP/TCP, G.711

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-4

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

Feature Under Test: Streaming over RTP-Unicast/UDP, G.711, IPv6

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtpUnicast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
 7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-5

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, G.711, IPv6

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-6

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

Feature Under Test: Streaming over RTP/RTSP/TCP, G.711, IPv6

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
 7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-1-7

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

Feature Under Test: Streaming over RTP-Unicast/UDP, AAC

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters

- in AAC - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
4. 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 *aacEncoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-8

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, AAC

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
4. 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 *aacEncoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-1-9

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

Feature Under Test: Streaming over RTP/RTSP/TCP, AAC

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
4. 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 *aacEncoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-1-10

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

Feature Under Test: Streaming over RTP-Unicast/UDP, AAC, IPv6

WSDL Reference: None

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

Pre-Requirement: 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspUnicast - Transport Protocol

- in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
5. 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 *aacEncoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-11

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, AAC, IPv6

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtpOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
5. 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 *aacEncoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-1-12

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

Feature 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters

- in AAC - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
5. 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 *aacEncoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-13

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, G.711

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-14

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, AAC

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters

- in AAC - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
5. 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 *aacEncoding* - expected media stream encoding
 6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
 7. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-15

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTPS/TCP, G.711, IPv6

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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
6. 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

7. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 5.
8. ONVIF Client restores HTTPS settings which was changed at step 4.
9. 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 does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-1-16

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

Feature 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. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
6. 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 *aacEncoding* - expected media stream encoding
7. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step [5](#).
8. ONVIF Client restores HTTPS settings which was changed at step [4](#).
9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.2.1.17 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/WebSockets)

Test Case ID: MEDIA2_RTSS-2-1-17

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket

WSDL Reference: None

Test Purpose: To verify G.711 media streaming over WebSocket.

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. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming

4. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in audio - media type
 - in G.711 - expected media stream encoding
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.2.1.18 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/WebSockets)

Test Case ID: MEDIA2_RTSS-2-1-18

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket

WSDL Reference: None

Test Purpose: To verify AAC media streaming over WebSocket.

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. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
4. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in audio - media type
 - in *aacEncoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.2.1.19 MEDIA2 STREAMING – G.711 (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-2-1-19

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket for IPv6.

WSDL Reference: None

Test Purpose: To verify G.711 media streaming over WebSocket for IPv6.

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

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in audio - media type
 - in G.711 - expected media stream encoding
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

5.2.1.20 MEDIA2 STREAMING – AAC (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-2-1-20

Specification Coverage: Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Streaming over WebSocket for IPv6.

WSDL Reference: None

Test Purpose: To verify AAC media streaming over WebSocket for IPv6.

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

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding

- in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
5. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.29](#) with the following input and output parameters
- in *streamUri* - Uri for media streaming
 - in audio - media type
 - in *aacEncoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

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

Test Case ID: MEDIA2_RTSS-2-1-21

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

Feature Under Test: Streaming over RTP-Unicast/UDP, G.711

WSDL Reference: None

Test Purpose: To verify G.711 media streaming based on RTP-Unicast/UDP Transport for case if there is only Audio Source Configuration and Audio encoder Configuration in Media Profile.

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.44](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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 Case ID: MEDIA2_RTSS-2-2-1

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

Feature Under Test: Streaming over RTP-Multicast, G.711

WSDL Reference: None

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

Pre-Requirement: 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. 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
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-2-2

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

Feature Under Test: Streaming over RTP-Multicast, G.711, IPv6

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. 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
6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-2-3

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

Feature Under Test: Streaming over RTP-Multicast, AAC

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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *aacEncoding* - AAC audio encoding that is set in profile
4. 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 *aacEncoding* - expected media stream encoding
 - in IPv4 - IP version for multicast streaming
5. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step [3](#).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

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

Test Case ID: MEDIA2_RTSS-2-2-4

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

Feature Under Test: Streaming over RTP-Multicast, AAC, IPv6

WSDL Reference: None

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

Pre-Requirement: 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 configures a media profile and retrieves a stream uri for audio streaming by following the procedure mentioned in [Annex A.40](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming

- out *aacEncoding* - AAC audio encoding that is set in profile
5. 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 *aacEncoding* - expected media stream encoding
 - in IPv6 - IP version for multicast streaming
 6. ONVIF Client restores settings of Audio Encoder Configuration and Media Profile changed at step 4.
 7. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.3 Audio Backchannel

5.3.1 Unicast

5.3.1.1 BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-1

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP-Unicast/UDP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/UDP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in `PCMU` - required audio decoding
 - in `RtspUnicast` - transport protocol
 - out `profile` - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out `streamUri` - Uri for media streaming
4. ONVIF Client tries to start audio backchannel streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.21](#) with the following input and output parameters
 - in `streamUri` - Uri for media streaming
 - in `G.711` - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 3.
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.2 BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-3

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP/RTSP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP/RTSP/TCP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder is supported by DUT. RTP/RTSP/TCP 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RTSP - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
4. ONVIF Client tries to start audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.22](#) with the following input and output parameters

- in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 3.
 6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.3 BACKCHANNEL – G.711 (RTP-Unicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-4

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP-Unicast/UDP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/UDP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RtspUnicast - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start audio backchannel streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.21](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 does not pass all assertions.

5.3.1.4 BACKCHANNEL – G.711 (RTP/RTSP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-6

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP/RTSP/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP/RTSP/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder is supported by DUT. RTP/RTSP/TCP 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 configures a Media Profile which supports a required audio decoding and send primacy with not only [www.onvif.org/ver20/HalfDuplex/Server](#) value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RTSP - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming

5. ONVIF Client tries to start audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.22](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 does not pass all assertions.

5.3.1.5 BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-7**Specification coverage:** Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).**Feature under test:** Audio Backchannel AAC, RTP-Unicast/UDP, IPv4**WSDL Reference:** None**Test Purpose:** To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/UDP transport for IPv4.**Pre-Requisite:** Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RtpUnicast - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
4. ONVIF Client tries to start audio backchannel streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.21](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step [3](#).
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step [3](#).

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.6 BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-9

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel AAC, RTP/RTSP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP/RTSP/TCP transport for IPv4.

Pre-Requirement: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder is supported by DUT. RTP/RTSP/TCP 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RTSP - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
4. ONVIF Client tries to start audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.22](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 3.

6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.7 BACKCHANNEL – AAC (RTP-Unicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-10

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel AAC, RTP-Unicast/UDP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/UDP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters

- in AAC - required audio decoding
 - in RtspUnicast - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
5. ONVIF Client tries to start audio backchannel streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.21](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
 6. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 4.
 7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
 8. 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 does not pass all assertions.

5.3.1.8 BACKCHANNEL – AAC (RTP/RTSP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-12

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel AAC, RTP/RTSP/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP/RTSP/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder is supported by DUT. RTP/RTSP/TCP 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RTSP - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
5. ONVIF Client tries to start audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.22](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding

6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 does not pass all assertions.

5.3.1.9 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/ WebSockets)

Test Case ID: MEDIA2_RTSS-3-1-13

Specification coverage: Back Channel Connection (Streaming), Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature under test: Audio Backchannel G.711 over WebSocket, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming over WebSocket for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder is supported by DUT. RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RTSP - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
4. ONVIF Client tries to start an audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.41](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 3.
6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.10 BACKCHANNEL – AAC (RTP-Unicast/RTSP/ WebSockets)

Test Case ID: MEDIA2_RTSS-3-1-14

Specification coverage: Back Channel Connection (Streaming), Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature under test: Audio Backchannel AAC over WebSocket, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming over WebSocket for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder is supported by DUT. RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RTSP - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
4. ONVIF Client tries to start an audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.41](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 3.
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.11 BACKCHANNEL – G.711 (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-15

Specification coverage: Back Channel Connection (Streaming), Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature under test: Audio Backchannel G.711 over WebSocket, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming over WebSocket for IPv6.

Pre-Requirement: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder is supported by DUT. RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding

- in RTSP - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start an audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.41](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
 6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
 7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
 8. 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 does not pass all assertions.

5.3.1.12 BACKCHANNEL – AAC (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-16

Specification coverage: Back Channel Connection (Streaming), Capabilities (ONVIF Media2 Service Specification), WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature under test: Audio Backchannel AAC over WebSocket, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming over WebSocket for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder is supported by DUT. RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RTSP - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
5. ONVIF Client tries to start an audio backchannel streaming over RTP/RTSP/TCP by following the procedure mentioned in [Annex A.41](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding

6. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
8. 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 does not pass all assertions.

5.3.1.13 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-21

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP-Unicast/RTSP/HTTP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/RTSP/HTTP/TCP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RtspOverHttp - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
4. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.58](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 3.
6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.14 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-22

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel G.711, RTP-Unicast/RTSP/HTTP/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/RTSP/HTTP/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. G.711 decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RtpOverHttp - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
5. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.58](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - expected media stream encoding

6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 does not pass all assertions.

5.3.1.15 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-23

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel AAC, RTP-Unicast/RTSP/HTTP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/RTSP/HTTP/TCP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RtspOverHttp - transport protocol
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
4. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.58](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
5. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step [3](#).
6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step [3](#).

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.16 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-24

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming).

Feature under test: Audio Backchannel AAC, RTP-Unicast/RTSP/HTTP/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/RTSP/HTTP/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by DUT. AAC decoder 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 configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RtspOverHttp - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
5. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.58](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming

- in *aacDecoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 4.
 7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
 8. 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 does not pass all assertions.

5.3.1.17 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-25

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming), RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Feature under test: Audio Backchannel G.711, Streaming over RTP-Unicast/RTSP/HTTPS/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/RTSP/HTTPS/TCP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by the DUT. G.711 decoder is supported by the DUT. Real-time streaming is supported by the DUT. RTP/RTSP/HTTPS/TCP feature is supported by the DUT. HTTPS is configured on the DUT, if TLS Server is not supported by the DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in `PCMU` - required audio decoding
 - in `RtspOverHttp` - transport protocol
 - out `profile` - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out `streamUri` - Uri for media streaming
5. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.59](#) with the following input and output parameters
 - in `streamUri` - Uri for media streaming
 - in `G.711` - expected media stream encoding
6. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
8. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.18 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv4)

Test Case ID: MEDIA2_RTSS-3-1-26

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming), RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Feature under test: Audio Backchannel AAC, Streaming over RTP-Unicast/RTSP/HTTPS/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/RTSP/HTTPS/TCP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by the DUT. AAC decoder is supported by the DUT. Real-time streaming is supported by the DUT. RTP/RTSP/HTTPS/TCP feature is supported by the DUT. HTTPS is configured on the DUT, if TLS Server is not supported by the DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only www.onvif.org/ver20/HalfDuplex/Server value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RtspOverHttp - transport protocol

- out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
5. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.59](#) with the following input and output parameters
- in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
6. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

5.3.1.19 BACKCHANNEL STREAMING BY POST – G.711 (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-27

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming), RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Feature under test: Audio Backchannel G.711, Streaming over RTP-Unicast/RTSP/HTTPS/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for G.711 audio streaming using RTP-Unicast/RTSP/HTTPS/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by the DUT. G.711 decoder is supported by the DUT. Real-time streaming is supported by the DUT. RTP/RTSP/HTTPS/TCP feature is supported by the DUT. HTTPS is configured on the DUT, if TLS Server is not supported by the DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in PCMU - required audio decoding
 - in RtspOverHttp - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
6. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.59](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming

- in G.711 - expected media stream encoding
7. ONVIF Client restores settings of Audio Decoder Configuration with @token = *profile.Configurations.AudioDecoder.@token* if it was changed at step 5.
 8. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 5.
 9. ONVIF Client restores HTTPS settings which was changed at step 4.
 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 does not pass all assertions.

5.3.1.20 BACKCHANNEL STREAMING BY POST – AAC (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-3-1-28

Specification coverage: Back Channel Connection (Streaming), RTSP Require- Tag (Streaming), Connection setup for a bi-directional connection (Streaming), RTP/RTSP/HTTP/TCP, RTP, RTCP, Stream control, RTSP, RTSP over HTTP, RTSP over HTTPS.

Feature under test: Audio Backchannel AAC, Streaming over RTP-Unicast/RTSP/HTTPS/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify DUT Backchannel for AAC audio streaming using RTP-Unicast/RTSP/HTTPS/TCP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio Backchannel is supported by the DUT. AAC decoder is supported by the DUT. Real-time streaming is supported by the DUT. RTP/RTSP/HTTPS/TCP feature is supported by the DUT. HTTPS is configured on the DUT, if TLS Server

is not supported by the DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client configures a Media Profile which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and gets stream URI for required transport protocol by following the procedure mentioned in [Annex A.20](#) with the following input and output parameters
 - in AAC - required audio decoding
 - in RtspOverHttp - transport protocol
 - in IPv6 - IP Type
 - out *profile* - Media Profile with Audio Output Configuration and Audio Decoder Configuration with the required audio decoding
 - out *streamUri* - Uri for media streaming
 - out *aacDecoding* - AAC audio decoding that is set in profile
6. ONVIF Client tries to start audio backchannel streaming by POST over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.59](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *aacDecoding* - expected media stream encoding
7. ONVIF Client restores settings of Audio Decoder Configuration with `@token = profile.Configurations.AudioDecoder.@token` if it was changed at step 5.

8. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 5.
9. ONVIF Client restores HTTPS settings which was changed at step 4.
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 does not pass all assertions.

5.4 Metadata Streaming

5.4.1 Unicast

5.4.1.1 METADATA STREAMING (RTP-Unicast/UDP)

Test Case ID: MEDIA2_RTSS-4-1-1

Specification Coverage: RTP data transfer via UDP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Unicast/UDP, IPv4

WSDL Reference: None

Test Purpose: To verify metadata streaming based on RTP/UDP Unicast Transport.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspUnicast* - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.33](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
5. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 3.
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.2 METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP)

Test Case ID: MEDIA2_RTSS-4-1-2

Specification Coverage: RTP/RTSP/HTTP/TCP, RTSP over HTTP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Unicast/RTSP/HTTP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify metadata streaming based on HTTP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspOverHttp* - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.34](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
5. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 3.
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.3 METADATA STREAMING (RTP/RTSP/TCP)

Test Case ID: MEDIA2_RTSS-4-1-3

Specification Coverage: RTP/RTSP/TCP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP/RTSP/TCP

WSDL Reference: None

Test Purpose: To verify metadata streaming based on RTP/RTSP/TCP Unicast Transport.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP/RTSP/TCP 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in RTSP - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client tries to start and decode media streaming over RTSP by following the procedure mentioned in [Annex A.35](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
5. ONVIF Client restores settings of Metadata Configuration with @token = *profile.Configurations.Metadata.@token* if it was changed at step [3](#).
6. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step [3](#).

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.4 METADATA STREAMING - SET SYNCHRONIZATION POINT

Test Case ID: MEDIA2_RTSS-4-1-4

Specification Coverage: RTP for Metadata stream, Synchronization Points.

Feature Under Test: Synchronization Points for Metadata Streaming

WSDL Reference: None

Test Purpose: To request synchronization point from DUT for metadata streaming.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspUnicast* - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.

5. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtmpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port1-port2*
8. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
11. If DUT does not send Metadata RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **SetSynchronizationPoint** request with parameters
 - ProfileToken := *profile.@token*
14. The DUT responds with **SetSynchronizationPointResponse** message.

15. If DUT does not close previous XML document and does not start new XML document, FAIL the test and skip other steps.
16. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
17. The DUT responds with **200 OK** message with parameters
 - Session
18. ONVIF Client restores settings of Metadata Configuration with @token = *profile.Configurations.Metadata.@token* if it was changed at step 3.
19. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **SetSynchronizationPointResponse** message.
- 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.3](#) for Name and Token Parameters Length limitations.

5.4.1.5 METADATA STREAMING (RTP-Unicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-4-1-5

Specification Coverage: RTP data transfer via UDP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Unicast/UDP, IPv6

WSDL Reference: None

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

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspUnicast* - Transport protocol
 - in *IPv6* - IP type
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.33](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
8. 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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.6 METADATA STREAMING (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-4-1-6

Specification Coverage: RTP/RTSP/HTTP/TCP, RTSP over HTTP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Unicast/RTSP/HTTP/TCP, IPv6

WSDL Reference: None

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

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspOverHttp* - Transport protocol
 - in *IPv6* - IP type

- out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTP/TCP by following the procedure mentioned in [Annex A.34](#) with the following input and output parameters
- in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
8. 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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.7 METADATA STREAMING (RTP/RTSP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-4-1-7

Specification Coverage: RTP/RTSP/TCP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP/RTSP/TCP, IPv6

WSDL Reference: None

Test Purpose: To verify metadata streaming based on RTP/RTSP/TCP Unicast Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP/RTSP/TCP 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in RTSP - Transport protocol
 - in IPv6 - IP type
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over RTSP by following the procedure mentioned in [Annex A.35](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with @token = *profile.Configurations.Metadata.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.8 METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP)

Test Case ID: MEDIA2_RTSS-4-1-8

Specification Coverage: RTP/RTSP/HTTPS/TCP, RTSP over HTTP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream, RTSP over HTTPS

Feature Under Test: Metadata Streaming, Streaming over RTP-Unicast/RTSP/HTTP/TCP, IPv4

WSDL Reference: None

Test Purpose: To verify metadata streaming based on HTTPS Transport.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 HTTPS if required by following the procedure mentioned in [Annex A.42](#).
4. ONVIF Client selects a Media Profile with Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters

- in *RtspOverHttp* - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.48](#) with the following input and output parameters
- in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.
8. ONVIF Client restores HTTPS settings which was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.9 METADATA STREAMING (RTP-Unicast/RTSP/HTTPS/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-4-1-9

Specification Coverage: RTP/RTSP/HTTPS/TCP, RTSP over HTTP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream, RTSP over HTTPS

Feature Under Test: Metadata Streaming, Streaming over RTP-Unicast/RTSP/HTTP/TCP, IPv6

WSDL Reference: None

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

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server 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 configures HTTPS if required by following the procedure mentioned in [Annex A.42](#).
5. ONVIF Client selects a Media Profile with Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspOverHttp* - Transport protocol
 - in *IPv6* - IP type
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
6. ONVIF Client tries to start and decode media streaming over RTP-Unicast/RTSP/HTTPS/TCP by following the procedure mentioned in [Annex A.48](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
7. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 5.

8. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 5.
9. ONVIF Client restores HTTPS settings which was changed at step 4.
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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.10 METADATA STREAMING (RTP-Unicast/RTSP/ WebSockets)

Test Case ID: MEDIA2_RTSS-4-1-10

Specification Coverage: WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Metadata Streaming, Streaming over WebSocket, IPv4

WSDL Reference: None

Test Purpose: To verify metadata streaming over WebSocket.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client selects a Media Profile with Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in RTSP - Transport protocol
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.49](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
5. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 3.
6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.1.11 METADATA STREAMING (RTP-Unicast/RTSP/ WebSockets, IPv6)

Test Case ID: MEDIA2_RTSS-4-1-11

Specification Coverage: WebSocket transport for RTP/RTSP/TCP (ONVIF Streaming Specification).

Feature Under Test: Metadata Streaming, Streaming over WebSocket, IPv6

WSDL Reference: None

Test Purpose: To verify metadata streaming over WebSocket for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. IPv6 is supported by DUT. WebSocket is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in RTSP - Transport protocol
 - in IPv6 - IP type
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over WebSocket by following the procedure mentioned in [Annex A.49](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with @token = *profile.Configurations.Metadata.@token* if it was changed at step 4.
7. ONVIF Client restores Media Profile with @token = *profile.@token* if it was changed at step 4.
8. 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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.2 Multicast

5.4.2.1 METADATA STREAMING (RTP-Multicast/UDP)

Test Case ID: MEDIA2_RTSS-4-2-1

Specification Coverage: RTP data transfer via UDP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Multicast/UDP, IPv4

WSDL Reference: None

Test Purpose: To verify metadata streaming using RTP-Multicast/UDP transport for IPv4.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP-Multicast/UDP 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspMulticast* - Transport protocol

- in IPv4 - IP version of Multicast streaming
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
4. ONVIF Client tries to start and decode media streaming over RTP-Multicast/UDP by following the procedure mentioned in [Annex A.36](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in IPv4 - IP version for multicast streaming
 - in *metadataConfiguration* - Metadata Configuration
 5. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 3.
 6. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.4.2.2 METADATA STREAMING (RTP-Multicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-4-2-2

Specification Coverage: RTP data transfer via UDP, RTP for Metadata stream, RTCP, Stream control, RTSP session for a Metadata stream.

Feature Under Test: Metadata Streaming, RTP-Multicast/UDP, IPv6

WSDL Reference: None

Test Purpose: To verify metadata streaming using RTP-Multicast/UDP transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Metadata feature is supported by the DUT. Real-time streaming is supported by DUT. RTP-Multicast/UDP 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 Metadata Configuration by following the procedure mentioned in [Annex A.30](#) with the following input and output parameters
 - in *RtspMulticast* - Transport protocol
 - in *IPv6* - IP type
 - out *profile* - Media Profile with Metadata Configuration
 - out *streamUri* - Uri for media streaming
 - out *metadataConfiguration* - Metadata Configuration
5. ONVIF Client tries to start and decode media streaming over RTP-Multicast/UDP by following the procedure mentioned in [Annex A.36](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in *IPv6* - IP version for multicast streaming
 - in *metadataConfiguration* - Metadata Configuration
6. ONVIF Client restores settings of Metadata Configuration with `@token = profile.Configurations.Metadata.@token` if it was changed at step 4.
7. ONVIF Client restores Media Profile with `@token = profile.@token` if it was changed at step 4.

8. 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 pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5 Audio & Video Streaming

5.5.1 Unicast

5.5.1.1 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP)

Test Case ID: MEDIA2_RTSS-5-1-1

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

Feature Under Test: Streaming over RTP-Unicast/UDP, G.711

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Unicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
5. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding
6. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.2 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/TCP)

Test Case ID: MEDIA2_RTSS-5-1-2

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, G.711

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Unicast/RTSP/HTTP/TCP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
5. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding

- in G.711 - expected audio stream encoding
6. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.3 MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP)

Test Case ID: MEDIA2_RTSS-5-1-3

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

Feature Under Test: Streaming over RTP/RTSP/TCP, G.711

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP/RTSP/TCP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. G.711 encoding is supported by DUT. RTP/RTSP/TCP 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters

- in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. Set *videoEncoding* := *profile*.Configurations.VideoEncoder.Encoding
 5. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding
 6. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.4 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-4

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

Feature Under Test: Streaming over RTP-Unicast/UDP, G.711, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Unicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding

7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
8. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.5 MEDIA2 STREAMING – H.26X/G.711 (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-5

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, G.711, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Unicast/RTSP/HTTP/TCP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. G.711 encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding
7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
8. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.6 MEDIA2 STREAMING – H.26X/G.711 (RTP/RTSP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-6

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

Feature Under Test: Streaming over RTP/RTSP/TCP, G.711, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP/RTSP/TCP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. G.711 encoding is supported by DUT. RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming

5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. 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* - 1st media type
 - in *audio* - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *G.711* - expected audio stream encoding
7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
8. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.7 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP)

Test Case ID: MEDIA2_RTSS-5-1-7

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

Feature Under Test: Streaming over RTP-Unicast/UDP, AAC

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Unicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
5. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.8 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/TCP)

Test Case ID: MEDIA2_RTSS-5-1-8

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, AAC

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Unicast/RTSP/HTTP/TCP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming

- out *profile* - Media profile with required configurations
4. Set *videoEncoding* := *profile*.Configurations.VideoEncoder.Encoding
 5. Set *audioEncoding* := *profile*.Configurations.AudioEncoder.Encoding
 6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
 7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.9 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP)

Test Case ID: MEDIA2_RTSS-5-1-9

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

Feature Under Test: Streaming over RTP/RTSP/TCP, AAC

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP/RTSP/TCP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. AAC encoding is supported by DUT. RTP/RTSP/TCP 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
5. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.10 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-10

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

Feature Under Test: Streaming over RTP-Unicast/UDP, AAC, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Unicast/UDP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspUnicast - Transport Protocol

- in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
 6. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
 7. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
 8. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
 9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.11 MEDIA2 STREAMING – H.26X/AAC (RTP-Unicast/RTSP/HTTP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-11

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

Feature Under Test: Streaming over RTP-Unicast/RTSP/HTTP/TCP, AAC, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Unicast/RTSP/HTTP/TCP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. AAC encoding is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspOverHttp - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
7. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
8. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.1.12 MEDIA2 STREAMING – H.26X/AAC (RTP/RTSP/TCP, IPv6)

Test Case ID: MEDIA2_RTSS-5-1-12

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

Feature Under Test: Streaming over RTP/RTSP/TCP, AAC, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP/RTSP/TCP Transport for IPv6.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding is supported by DUT. AAC encoding is supported by DUT.

RTP/RTSP/TCP is supported by DUT. Real-time streaming is supported by DUT. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RTSP - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
7. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
8. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.

9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.2 Multicast

5.5.2.1 MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast/UDP)

Test Case ID: MEDIA2_RTSS-5-2-1

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

Feature Under Test: Streaming over RTP-Multicast, G.711

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Multicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters

- in PCMU - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
 5. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding
 6. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.2.2 MEDIA2 STREAMING – H.26X/G.711 (RTP-Multicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-5-2-2

Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP, IPv6.

Feature Under Test: Streaming over RTP-Multicast, G.711, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/G.711 or H.265/G.711 video and audio media streaming based on RTP-Multicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in G.711 - expected audio stream encoding

7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
8. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.2.3 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP)

Test Case ID: MEDIA2_RTSS-5-2-3

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

Feature Under Test: Streaming over RTP-Multicast, AAC

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Multicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters

- in AAC - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
 5. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
 6. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
 7. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.5.2.4 MEDIA2 STREAMING – H.26X/AAC (RTP-Multicast/UDP, IPv6)

Test Case ID: MEDIA2_RTSS-5-2-4

Specification Coverage: RTP data transfer via UDP, RTP, RTCP, Stream control, RTSP, IPv6.

Feature Under Test: Streaming over RTP-Multicast, AAC, IPv6

WSDL Reference: None

Test Purpose: To verify H.264/AAC or H.265/AAC video and audio media streaming based on RTP-Multicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio streaming is supported by DUT. H.264 encoding OR H.265 encoding 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. IPv6 is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client 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 configures a media profile and retrieves a stream uri for video and audio streaming by following the procedure mentioned in [Annex A.45](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in RtspMulticast - Transport Protocol
 - in IPv6 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - Media profile with required configurations
5. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
6. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
7. 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 - 1st media type
 - in audio - 2nd media type
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
8. ONVIF Client restores settings of Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 4.
 9. 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 does not pass all assertions.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

5.6 Audio Backchannel & Video & Audio Streaming

5.6.1 Unicast

5.6.1.1 MEDIA2 STREAMING – G.711 BACKCHANNEL AND H.26X VIDEO AND G.711/AAC AUDIO (RTP-Unicast/UDP)

Test Case ID: MEDIA2_RTSS-6-1-1

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

Feature Under Test: Audio Backchannel G.711 with Video and Audio streaming, RTP-Unicast/UDP, IPv4

WSDL Reference: None

Test Purpose: To verify G.711 audio backchannel with H.264 or H.265 video and with G.711 or AAC audio media streaming based on RTP-Unicast/UDP Transport.

Pre-Requisite: Media2 Service is received from the DUT. Audio Outputs is supported by DUT. G.711 decoder is supported by DUT. Audio is supported by DUT. G.711 OR AAC audio encoding is supported by DUT. Video is supported by DUT. H.264 OR H.265 video 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 configures a media profile and retrieves a stream uri for audio backchannel and video and audio streaming by following the procedure mentioned in [Annex A.63](#) with the following input and output parameters
 - in PCMU - required supported audio decoder
 - in RtspUnicast - Transport Protocol
 - in IPv4 - IP version
 - out *streamUri* - Uri for media streaming
 - out *profile* - configured media profile
4. Set *videoEncoding* := *profile.Configurations.VideoEncoder.Encoding*
5. Set *audioEncoding* := *profile.Configurations.AudioEncoder.Encoding*
6. ONVIF Client tries to start audio backchannel streaming and start and decode media streaming over RTP-Unicast/UDP by following the procedure mentioned in [Annex A.66](#) with the following input and output parameters
 - in *streamUri* - Uri for media streaming
 - in G.711 - audio backchannel stream encoding
 - in *videoEncoding* - expected video stream encoding
 - in *audioEncoding* - expected audio stream encoding
7. ONVIF Client restores settings of Audio Decoder Configuration, Video Encoder Configuration, Audio Encoder Configuration, and Media Profile changed at step 3.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

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. If DUT supports Discovery:
 - 14.1. ONVIF Client waits for HELLO message from the default network interface.
15. If DUT does not support Discovery:
 - 15.1. ONVIF Client waits during *rebootTimeout*.
16. ONVIF Client gets current network interfaces via GetNetworkInterfacesRequest message.
17. ONVIF Client verifies GetNetworkInterfacesResponse message and checks that set settings were applied. 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 and DUT supports Discovery, wait for HELLO message from the default network interface.
4. If DUT does not support Discovery:
 - 4.1. ONVIF Client waits during *rebootTimeout*.

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] := VideoSource
 - Type[1] := VideoEncoder
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*

3. For each Media Profile *profile1* in *profileList* with both Configuration.VideoSource and Configuration.VideoEncoder repeat the following steps:
 - 3.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
 - ConfigurationToken := *profile1*.Configuration.VideoEncoder.@token
 - ProfileToken := *profile1*.@token
 - 3.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
 - Options list =: *optionsList*
 - 3.3. If *optionsList* list contains item with Encoding = *requiredVideoEncoding*:
 - 3.3.1. Set *profile* := *profile1*.
 - 3.3.2. Set *vecOptions* := item with Encoding = *requiredVideoEncoding* from *optionsList* list.
 - 3.3.3. Skip other steps in procedure.
4. For each Media Profile *profile1* in *profileList* that contains VideoSource configuration repeat the following steps:
 - 4.1. If *profile1*.Configurations.VideoSource.@token is different from video source configuration token of previous profiles in cycle:
 - 4.1.1. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile1*.@token
 - 4.1.2. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList*
 - 4.1.3. For each Video Encoder Configuration *videoEncoderConfiguration1* in *videoEncoderConfList* repeat the following steps:
 - 4.1.3.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *videoEncoderConfiguration1.@token*
 - ProfileToken := *profile1.@token*
- 4.1.3.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
- Options list =: *optionsList*
- 4.1.3.3. If *optionsList* list contains item with Encoding = *requiredVideoEncoding*:
- 4.1.3.3.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile1.@token*
 - Name skipped
 - Configuration[0].Type := VideoEncoder
 - Configuration[0].Token := *videoEncoderConfiguration1.@token*
- 4.1.3.3.2. The DUT responds with **AddConfigurationResponse** message.
- 4.1.3.3.3. Set *profile* := *profile1*.
- 4.1.3.3.4. Set *vecOptions* := item with Encoding = *requiredVideoEncoding* from *optionsList* list.
- 4.1.3.3.5. Skip other steps in procedure.
5. Set *profile1* := *profileList*[0]
6. Set *confTypeList* := (configurations that are contained in profile *profile1*)
7. ONVIF Client removes all configurations from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters
- in *confTypeList* - list of configuration type to remove from Media Profile
 - in *profile1* - Media Profile to update
8. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters

- ConfigurationToken skipped
 - ProfileToken := *profile1.@token*
9. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
- Configurations list =: *videoSourceConfList*
10. For each Video Source Configuration *videoSourceConfiguration1* in *videoSourceConfList* repeat the following steps:
- 10.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile1.@token*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfiguration1.@token*
- 10.2. The DUT responds with **AddConfigurationResponse** message.
- 10.3. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
- ConfigurationToken skipped
 - ProfileToken := *profile1.@token*
- 10.4. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters
- Configurations list =: *videoEncoderConfList*
- 10.5. For each Video Encoder Configuration *videoEncoderConfiguration1* in *videoEncoderConfList* repeat the following steps:
- 10.5.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
- ConfigurationToken := *videoEncoderConfiguration1.@token*
 - ProfileToken := *profile1.@token*
- 10.5.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

10.5.3. If *optionsList* list contains item with Encoding = *requiredVideoEncoding*:

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

- ProfileToken := *profile1.@token*
- Name skipped
- Configuration[0].Type := VideoEncoder
- Configuration[0].Token := *videoEncoderConfiguration1.@token*

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

10.5.3.3. Set *profile* := *profile1*.

10.5.3.4. Set *vecOptions* := item with Encoding = *requiredVideoEncoding* from *optionsList* list.

10.5.3.5. Skip other steps in procedure.

11. 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.
- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** 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 *requiredAudioEncoding* = AAC:

- 3.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
 - 3.3.1.1. Set *profile* := *profile1*.
 - 3.3.1.2. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if exists, otherwise item with Encoding = "MPEG4-GENERIC".
 - 3.3.1.3. Skip other steps in procedure.
- 3.4. If *requiredAudioEncoding* = !AAC:
 - 3.4.1. If *optionsList* list contains item with Encoding = *requiredAudioEncoding*:
 - 3.4.1.1. Set *profile* := *profile1*.
 - 3.4.1.2. Set *aecOptions* := item with Encoding = *requiredAudioEncoding* from *optionsList* list.
 - 3.4.1.3. 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 *requiredAudioEncoding* = AAC:
- 4.3.5.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
- 4.3.5.3.1.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile1.@token*
 - Name skipped
 - Configuration[0].Type := AudioEncoder
 - Configuration[0].Token := *audioEncoderConfiguration1.@token*

- 4.3.5.3.1.2. The DUT responds with **AddConfigurationResponse** message.
 - 4.3.5.3.1.3. Set *profile* := *profile1*.
 - 4.3.5.3.1.4. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if exists, otherwise item with Encoding = "MPEG4-GENERIC".
 - 4.3.5.3.1.5. Skip other steps in procedure.
- 4.3.5.4. If *requiredAudioEncoding* = !AAC:
- 4.3.5.4.1. If *optionsList* list contains item with Encoding = *requiredAudioEncoding*:
 - 4.3.5.4.1.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile1*.@token
 - Name skipped
 - Configuration[0].Type := AudioEncoder
 - Configuration[0].Token := *audioEncoderConfiguration1*.@token
 - 4.3.5.4.1.2. The DUT responds with **AddConfigurationResponse** message.
 - 4.3.5.4.1.3. Set *profile* := *profile1*.
 - 4.3.5.4.1.4. Set *aecOptions* := item with Encoding = *requiredAudioEncoding* from *optionsList* list.
 - 4.3.5.4.1.5. 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: None.

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: None.

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*). Media type2 (*mediaType2*) (optional parameter). Expected media stream encoding for Media type2 (*encoding2*) (optional parameter).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = *mediaType* with rtpmap value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
4. If *mediaType2* is specified and *sdp* does not contain Media Type = *mediaType2* with rtpmap value corresponding to *encoding2*, FAIL the test and skip other steps.
5. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
6. 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*
7. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
8. If *mediaType2* is specified:
 - ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to *mediaType2* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port3-port4*
 - The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*

9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
11. If DUT does not send *encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If *mediaType2* is specified and DUT does not send *encoding2* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
13. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
14. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
15. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

Note: If *encoding2* = MP4A-LATM, then *rtpmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 4.

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*). Media type2 (*mediaType2*) (optional parameter). Expected media stream encoding for Media type2 (*encoding2*) (optional parameter).

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*
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = *mediaType* with *rtpmap* value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. If *mediaType2* is specified and *sdp* does not contain Media Type = *mediaType2* with *rtpmap* value corresponding to *encoding2* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
7. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters

- in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
8. 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*
9. The DUT responds with **200 OK** message on *connection1* with parameters
- Transport
 - Session =: *session*
10. If *mediaType2* is specified:
- ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to *mediaType2* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;client_port=*port3-port4*
 - The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
11. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
- Session := *session*
12. The DUT responds with **200 OK** message on *connection1* with parameters
- Session
 - RTP-Info
13. If DUT does not send *encoding* RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.
14. If *mediaType2* is specified and DUT does not send *encoding2* RTP media stream to ONVIF Client over *connection1*, 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 **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters

- Session := *session*

17. ONVIF Client closes *connection2*.

18. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.

Note: If *encoding2* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 6.

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*). Media type2 (*mediaType2*) (optional parameter). Expected media stream encoding for Media type2 (*encoding2*) (optional parameter).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = *mediaType* with rtpmap value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
4. If *mediaType2* is specified and *sdp* does not contain Media Type = *mediaType2* with rtpmap value corresponding to *encoding2* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
5. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
6. 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
7. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
8. If *mediaType2* is specified:
 - ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to *mediaType2* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;interleaved=0-1
 - The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*

9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
11. If DUT does not send *encoding* RTP media stream to ONVIF Client over RTSP control connection, FAIL the test and skip other steps.
12. If *mediaType2* is specified and DUT does not send *encoding2* RTP media stream to ONVIF Client over RTSP control connection, FAIL the test and skip other steps.
13. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
14. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
15. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

Note: If *encoding2* = MP4A-LATM, then *rtpmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 4.

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*). Media type2 (*mediaType2*) (optional parameter). Expected media stream encoding for Media type2 (*encoding2*) (optional parameter).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = *mediaType* with *rtpmap* value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
4. If *mediaType2* is specified and *sdp* does not contain Media Type = *mediaType2* with *rtpmap* value corresponding to *encoding2* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
5. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
6. 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*
7. The DUT responds with **200 OK** message with parameters
- Transport
 - Session := *session*
8. If *mediaType2* is specified:
- ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to *mediaType2* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;multicast;client_port=*port3-port4*
 - The DUT responds with **200 OK** message with parameters
 - Transport
 - Session := *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
10. The DUT responds with **200 OK** message with parameters
- Session
 - RTP-Info
11. If DUT does not send *encoding* RTP *ipVersion* multicast media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If *mediaType2* is specified and DUT does not send *encoding2* RTP *ipVersion* multicast media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
13. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
14. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
15. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

Note: If *encoding2* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 4.

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 rtp address on *connection2*.

4. The DUT responds with **200 OK** message with parameters on *connection1*
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = *mediaType* with rtpmap value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. 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*
8. The DUT responds with **200 OK** message on *connection1* with parameters
 - Transport
 - Session =: *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message on *connection1* with parameters
 - Session
 - RTP-Info
11. If DUT does not send *encoding* RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters

- Session := *session*

14. ONVIF Client closes *connection2*.

15. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmpmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.

A.15 Configuring HTTPS using Security Configuration Service

Name: HelperConfigureHTTPS

Procedure Purpose: Helper Procedure to configure HTTPS using Security Configuration Service.

Pre-requisite: Security Configuration 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. ONVIF Client invokes **GetAssignedServerCertificates**.
2. The DUT responds with a **GetAssignedServerCertificatesResponse** message with parameters
 - CertificationPathID list =: *initialCertificationPathList*
3. If number of items in *initialCertificationPathList* >= 1, go to the step 6.
4. If Create self-signed certificate is supported by the DUT:
 - 4.1. ONVIF 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](#).
 - 4.2. Go to the step 6.
5. 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](#).
6. ONVIF Client invokes **SetNetworkProtocols** request with parameters
 - NetworkProtocols[0].Name := HTTPS
 - NetworkProtocols[0].Enabled := true
 - NetworkProtocols[0].Port := 443
 - NetworkProtocols[0].Extension skipped
7. The DUT responds with **SetNetworkProtocolsResponse** message.
8. ONVIF Client waits until *operationDelay* timeout expires.
9. ONVIF Client checks that HTTPS protocol Port 443 is open. If HTTPS protocol port 443 is not open, FAIL the test and skip other steps.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: *operationDelay* 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_SSACertificate

Procedure Purpose: Helper Procedure to configure HTTPS using Security Configuration Service.

Pre-requisite: Security Configuration 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
 - CertificateIDs.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.
 8. ONVIF Client waits for time *operationDelay*.

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.

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

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 Security Configuration Service.

Pre-requisite: Security Configuration Service is received from the DUT. TLS Server is supported by the DUT. Create PKCS#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. ONVIF Client creates an CA certificate by following the procedure mentioned in [Annex A.37](#) with the following input and output parameters
 - out *CAcert* - CA certificate
 - out *privateKey* - private key for the CA certificate
 - out *publicKey* - public key for the CA certificate
5. 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*)
6. ONVIF Client invokes **UploadCertificate** with parameters
- Certificate := *cert*
 - Alias := "ONVIF_Test1"
 - PrivateKeyRequired := true
7. The DUT responds with a **UploadCertificateResponse** message with parameters
- CertificateID =: *certID*
 - KeyID =: *keyID*
8. ONVIF Client invokes **CreateCertificationPath** with parameters
- CertificateIDs.CertificateID[0] := *certID*
 - Alias := "ONVIF_Test2"
9. The DUT responds with a **CreateCertificationPathResponse** message with parameters
- CertificationPathID =: *certPathID*
10. ONVIF Client invokes **AddServerCertificateAssignment** with parameters
- CertificationPathID := *certPathID*
11. The DUT responds with an **AddServerCertificateAssignmentResponse** message.
12. ONVIF Client waits for time *operationDelay*.

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.

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

A.18 Create an RSA key pair

Name: HelperCreateRSAKeyPair

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

Pre-requisite: Security Configuration 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* + *operationDelay* expires repeat the following steps:
 - 6.1. ONVIF Client waits for 5 seconds.
 - 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", skip other steps of the procedure.
 - 6.5. If *keyStatus* is equal to "corrupt", FAIL the test and skip other steps.
7. If *duration* + *operationDelay* 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: *operationDelay* 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

A.20 Media2 Service – Media Profile Configuration for Audio Backchannel Streaming

Name: HelperConfigureMediaProfileForBackchannelStreaming

Procedure Purpose: Helper procedure to configure Media Profile to contain Audio Output Configuration and Audio Decoder Configuration which supports a required audio decoding and send primacy with not only `www.onvif.org/ver20/HalfDuplex/Server` value and retrieves stream uri.

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

Input: Required audio decoding (*requiredAudioDecoding*). Transport protocol (*transportProtocol*). IP type *ipType*.

Returns: Media Profile (*profile*) containing Audio Output Configuration and Audio Decoder Configuration with the required audio decoding. Uri for media streaming (*streamUri*). Audio decoding set in profile (*requiredAudioDecoding*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := AudioOutput
 - Type[1] := AudioDecoder
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*
3. For each Media Profile *profile1* in *profileList* with both Configuration.AudioOutput and Configuration.AudioDecoder repeat the following steps:
 - 3.1. ONVIF Client invokes **GetAudioOutputConfigurationOptions** request with parameters
 - ConfigurationToken := *profile1*.Configuration.AudioOutput.@token
 - ProfileToken := *profile1*.@token
 - 3.2. DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
 - Options =: *aocOptions*
 - 3.3. If *aocOptions*.SendPrimacyOptions list is not skipped and contains only one item which is equal to `www.onvif.org/ver20/HalfDuplex/Server` go to the next item at step 3.
 - 3.4. Set *audioOutputConfiguration* := *profile1*.Configuration.AudioOutput.

- 3.5. ONVIF Client invokes **GetAudioDecoderConfigurationOptions** request with parameters
 - ConfigurationToken := *profile1*.Configuration.AudioDecoder.@token
 - ProfileToken := *profile1*.@token
- 3.6. DUT responds with **GetAudioDecoderConfigurationOptionsResponse** message with parameters
 - Options list =: *adcOptionsList*
- 3.7. If *requiredAudioDecoding* = AAC:
 - 3.7.1. If *adcOptionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
 - 3.7.1.1. Set *profile* := *profile1*.
 - 3.7.1.2. Set *requiredAudioDecoding* := "MP4A-LATM" if *adcOptionsList* contains item with Encoding = "MP4A-LATM", otherwise "MPEG4-GENERIC".
 - 3.7.1.3. Go to step 6.
- 3.8. If *requiredAudioDecoding* != AAC:
 - 3.8.1. If *adcOptionsList* list contains item with Encoding = *requiredAudioDecoding*:
 - 3.8.1.1. Set *profile* := *profile1*.
 - 3.8.1.2. Go to step 6.
4. For each Media Profile *profile1* in *profileList* repeat the following steps:
 - 4.1. ONVIF Client invokes **GetAudioOutputConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile1*.@token
 - 4.2. The DUT responds with **GetAudioOutputConfigurationsResponse** with parameters
 - Configurations list =: *audioOutputConfList*
 - 4.3. For each Audio Output Configuration *audioOutputConfiguration1* in *audioOutputConfList* repeat the following steps:

- 4.3.1. ONVIF Client invokes **GetAudioOutputConfigurationOptions** request with parameters
 - ConfigurationToken := *audioOutputConfiguration1.@token*
 - ProfileToken := *profile1.@token*
- 4.3.2. DUT responds with **GetAudioOutputConfigurationOptionsResponse** message with parameters
 - Options =: *aocOptions*
- 4.3.3. If *aocOptions.SendPrimacyOptions* list is not skipped and contains only one item which is equal to www.onvif.org/ver20/HalfDuplex/Server go to the next item at step [4.3](#).
- 4.3.4. Set *audioOutputConfiguration* := *audioOutputConfiguration1*.
- 4.3.5. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile1.@token*
 - Name skipped
 - Configuration[0].Type := AudioOutput
 - Configuration[0].Token := *audioOutputConfiguration1.@token*
- 4.3.6. The DUT responds with **AddConfigurationResponse** message.
- 4.3.7. ONVIF Client invokes **GetAudioDecoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile1.@token*
- 4.3.8. The DUT responds with **GetAudioDecoderConfigurationsResponse** with parameters
 - Configurations list =: *audioDecoderConfList*
- 4.3.9. For each Audio Decoder Configuration *audioDecoderConfiguration1* in *audioDecoderConfList* repeat the following steps:

- 4.3.9.1. ONVIF Client invokes **GetAudioDecoderConfigurationOptions** request with parameters
- ConfigurationToken := *audioDecoderConfiguration1.@token*
 - ProfileToken := *profile1.@token*
- 4.3.9.2. DUT responds with **GetAudioDecoderConfigurationOptionsResponse** message with parameters
- Options list =: *adcOptionsList*
- 4.3.9.3. If *requiredAudioDecoding* = AAC:
- 4.3.9.3.1. If *adcOptionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
- 4.3.9.3.1.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile1.@token*
 - Name skipped
 - Configuration[0].Type := AudioDecoder
 - Configuration[0].Token := *audioDecoderConfiguration1.@token*
- 4.3.9.3.1.2. The DUT responds with **AddConfigurationResponse** message.
- 4.3.9.3.1.3. Set *profile* := *profile1*.
- 4.3.9.3.1.4. Set *requiredAudioDecoding* := "MP4A-LATM" if *adcOptionsList* contains item with Encoding = "MP4A-LATM", otherwise "MPEG4-GENERIC".
- 4.3.9.3.1.5. Go to step 6.
- 4.3.9.4. If *requiredAudioDecoding* = !AAC:
- 4.3.9.4.1. If *adcOptionsList* list contains item with Encoding = *requiredAudioDecoding*:

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

- ProfileToken := *profile1.@token*
- Name skipped
- Configuration[0].Type := AudioDecoder
- Configuration[0].Token := *audioDecoderConfiguration1.@token*

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

4.3.9.4.1.3. Set *profile* := *profile1*.

4.3.9.4.1.4. Go to step 6.

5. FAIL the test and skip other steps.

6. If *audioOutputConfiguration.SendPrimacy* = www.onvif.org/ver20/HalfDuplex/Server:

6.1. ONVIF Client invokes **SetAudioOutputConfiguration** request with parameters

- Configuration.@token := *audioOutputConfiguration.@token*
- Configuration.Name := *audioOutputConfiguration.Name*
- Configuration.UseCount := *audioOutputConfiguration.UseCount*
- Configuration.OutputToken := *audioOutputConfiguration.OutputToken*
- Configuration.SendPrimacy := the highest value from *aocOptions.SendPrimacyOptions* list according to the following order - www.onvif.org/ver20/HalfDuplex/Client, www.onvif.org/ver20/HalfDuplex/Auto
- Configuration.OutputLevel := *audioOutputConfiguration.OutputLevel*

6.2. DUT responds with **SetAudioOutputConfigurationResponse** message.

7. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters

- in *transportProtocol* - Transport protocol

- in *ipType* - IP Type
- in *profile.@token* - Media profile token
- out *streamUri* - Stream URI

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetAudioOutputConfigurationOptionsResponse** message.
- DUT did not send **GetAudioDecoderConfigurationOptionsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetAudioOutputConfigurationsResponse** message.
- DUT did not send **GetAudioDecoderConfigurationsResponse** message.
- DUT did not send **SetAudioOutputConfigurationResponse** message.

A.21 Audio Backchannel streaming over RTP-Unicast/UDP

Name: HelperBackchannelStreamingRTPUnicastUDP

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

Pre-requisite: Audio Backchannel is supported by DUT.

Input: Uri for audio backchannel streaming (*streamUri*). Expected audio stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters

- Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
 4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
 5. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to audio backchannel media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port1-port2*
 6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 7. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 9. ONVIF Client sends RTP Unicast audio stream with *encoding* to DUT over UDP.
 10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 11. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters

- Session := *session*

12. 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: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

A.22 Audio Backchannel Streaming over RTP/RTSP/TCP

Name: HelperBackchannelStreamingRTPRTSPTCP

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

Pre-requisite: Audio Backchannel is supported by DUT. RTP/RTSP/TCP is supported by DUT.

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

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*

- SDP information =: *sdp*
3. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
 4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
 5. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to *mediaType* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;interleaved=0-1
 6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 7. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 9. ONVIF Client sends RTP Unicast audio stream with *encoding* to DUT over RTSP control connection.
 10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 11. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters

- Session := *session*

12. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

A.23 Audio Backchannel Streaming over RTP-Multicast

Name: HelperBackchannelStreamingRTPMulticast

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

Pre-requisite: Audio Backchannel is supported by DUT. RTP-Multicast is supported by DUT.

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

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters

- Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
 4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
 5. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to *mediaType* media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;multicast;client_port=*port1-port2*
 6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 7. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 9. ONVIF Client sends RTP *ipVersion* Multicast audio stream with *encoding* to DUT over UDP.
 10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 11. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters

- Session := *session*

12. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtptime* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

A.24 Removing Video Encoder Configuration, Audio Encoder Configuration and Metadata Configuration from Media Profile

Name: HelperRemoveAudioEncoderConfigAndMetadataConfigAndVideoConfigFromMediaProfile

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

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

Input: Media Profile (*profile*)

Returns: None.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token := *profile.@token*

- Type[0] := AudioEncoder
 - Type[1] := Metadata
 - Type[2] := VideoEncoder
2. The DUT responds with **GetProfilesResponse** message with parameters
- Profiles list =: *profileList*
3. If *profileList*[0] contains Configuration.VideoEncoder or 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
 - If *profileList*[0] contains Configuration.VideoEncoder:
 - Configuration[2].Type := VideoEncoder
 - Configuration[2].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.25 Get Media2 Service Capabilities

Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to get Media2 Service Capabilities from the DUT.

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

Input: None

Returns: The service capabilities (*cap*).

Procedure:

1. ONVIF Client invokes **GetServiceCapabilities** request.
2. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
 - Capabilities =: *cap*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.26 Web Socket Handshake

Name: HelperWebSocketHandshake

Procedure Purpose: Helper procedure to establish a WebSocket Connection.

Pre-requisite: WebSocket is supported by the DUT.

Input: Web Socket Uri (*uri*)

Returns: None.

Procedure:

1. ONVIF Client generates a Sec-WebSocket-Key value by following the procedure mentioned in [Annex A.27](#) with the following input and output parameters
 - out *websocketKey* - Sec-WebSocket-Key value.

2. If scheme component of *uri* is equal to **ws**:
 - 2.1. ONVIF Client invokes **HTTP GET** request to *uri* with parameters
 - Upgrade =: "websocket"
 - Connection =: "Upgrade"
 - Sec-WebSocket-Key =: *webSocketKey*
 - Sec-WebSocket-Protocol =: "rtsp.onvif.org"
 - Sec-WebSocket-Version =: "13"
 - 2.2. The DUT responds with **HTTP 101 Switching Protocols** message with parameters
 - Upgrade =: *upgrade*
 - Connection =: *connection*
 - Sec-WebSocket-Accept =: *accept*
 - Sec-WebSocket-Protocol =: *protocol*
3. If scheme component of *uri* is equal to **wss**:
 - 3.1. If the DUT does not support TLS Server, FAIL the test and skip other steps.
 - 3.2. ONVIF Client invokes **GetNetworkProtocols** request.
 - 3.3. The DUT responds with **GetNetworkProtocolsResponse** with parameters
 - NetworkProtocols list =: *networkProtocolsList*
 - 3.4. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, go to step [3.6](#).
 - 3.5. ONVIF Client configures HTTPS by following the procedure mentioned in [Annex A.15](#).
 - 3.6. ONVIF Client performs a TLS handshake by following the procedure mentioned in [Annex A.28](#)
 - 3.7. ONVIF Client invokes **HTTPS GET** request to *uri* with parameters
 - Upgrade =: "websocket"
 - Connection =: "Upgrade"
 - Sec-WebSocket-Key =: *webSocketKey*

- Sec-WebSocket-Protocol =: "rtsp.onvif.org"
- Sec-WebSocket-Version =: "13"

3.8. The DUT responds with **HTTPS 101 Switching Protocols** message with parameters

- Upgrade =: *upgrade*
- Connection =: *connection*
- Sec-WebSocket-Accept =: *accept*
- Sec-WebSocket-Protocol =: *protocol*

4. If *upgrade* is not equal to "websocket", FAIL the test and skip other steps.

5. If *connection* is not equal to "Upgrade", FAIL the test and skip other steps.

6. If *accept* other than the base64-encoded SHA-1 of the concatenation of the *webSocketKey* (see RFC[6455] 4.1. Client Requirements), FAIL the test and skip other steps.

7. If *protocol* is not equal to "rtsp.onvif.org", FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **HTTP 101**.
- DUT did not send **GetNetworkProtocolsResponse**.

A.27 Sec-WebSocket-Key value generation

Name: HelperGenerateSecWebSocketKey

Procedure Purpose: Helper procedure to generate a Sec-WebSocket-Key value that is compliant to [RFC6455] and [RFC4648].

Pre-requisite: None.

Input: None.

Returns: Sec-WebSocket-Key value (*webSocketKey*)

Procedure:

1. ONVIF Client generates a nonce consisting of a randomly selected 16-byte Sec-WebSocket-Key value that has been base64-encoded (see Section 4 of [RFC4648] and section 4.1 of [RFC6455]).

A.28 Basic TLS handshake

Name: HelperBasicTLShandshake

Procedure Purpose: Helper procedure to execute basic TLS handshake.

Pre-requisite: TLS is supported by the DUT as indicated by the TLSServerSupported capability. TLS is configured. HTTPS protocol is enabled.

Input: None.

Returns: None.

Procedure:

1. ONVIF Client invokes **ClientHello** with parameters
 - ClientVersion := 3,1
 - Random number := *ClientRandom[32]*, that is 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number
 - CipherSuites := list of common CipherSuites used by TLS 1.0, SSL 2.0 and 3.0
 - Compression methods list := NONE
 - SessionID skipped
 - Extension: server_name := Server Name List
2. The DUT TLS server responds with a **ServerHello** message with parameters
 - Version =: the highest version number supported by both sides
 - Random number =: *ServerRandom[32]*, that is 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number
 - CipherSuite =: the strongest cipher that both the client and server support
 - Compression method =: NONE
 - Session ID =: *SessionID*
3. The DUT TLS server responds with **Certificate** message with parameters

- Certificate.CertificateID =: *CertificateID*
 - Certificate.KeyID =: *KeyID*
4. The DUT TLS server responds with a **ServerHelloDone** message.
 5. ONVIF Client invokes **ClientKeyExchange** message with parameters
 - Premaster Secret := *PreMasterSecret* encrypted with *KeyID*
 6. ONVIF Client computes *MasterSecret* using ClientRandom[32], ServerRandom[32] and PreMasterSecret.
 7. The DUT TLS server computes *MasterSecret* using ClientRandom[32], ServerRandom[32] and *PreMasterSecret*.
 8. ONVIF Client invokes **ChangeCipherSpec** message.
 9. ONVIF Client invokes encrypted **Finished** message, containing a hash := *hash1* and MAC := *MAC1* over the previous handshake messages.
 10. The DUT TLS server decrypts the client's **Finished** message and verify the hash and MAC.
 11. The DUT TLS server responds its encrypted **Finished** message, containing a hash =: *hash2* and MAC =: *MAC2* over the previous handshake messages.
 12. If *hash1* is not equal to *hash2*, FAIL the test.
 13. If *MAC1* is not equal to *MAC2*, FAIL the test.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- The DUT TLS server did not send **ServerHello** message.
- The DUT TLS server did not send **Certificate** message.
- The DUT TLS server did not send **ServerHelloDone** message.
- The DUT TLS server did not send **ChangeCipherSpec** message.
- The DUT TLS server did not send **Finished** message.
- The DUT TLS server sends Alert Message.

A.29 Media Streaming over WebSocket

Name: HelperStreamingOverWebSocket

Procedure Purpose: Helper procedure to verify media streaming over WebSocket.

Pre-requisite: WebSocket is supported by the DUT.

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

Returns: None

Procedure:

1. ONVIF Client gets Web Socket Uri by following the procedure mentioned in [Annex A.57](#) with the following output parameters
 - out *uri* - Web Socket Uri
2. ONVIF Client establishes a WebSocket Connection by following the procedure mentioned in [Annex A.26](#) with the following input and output parameters
 - in *uri* - Web Socket Uri
3. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address over WebSocket.
4. The DUT responds with **200 OK** message over WebSocket with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = *mediaType* with rtpmap value corresponding to *encoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. ONVIF Client invokes **RTSP SETUP** request over WebSocket to uri address, which corresponds to *mediaType* media type (see [RFC2326] for details), with parameters

- Transport := RTP/AVP/TCP;unicast;interleaved=0-1
8. The DUT responds with **200 OK** message over WebSocket with parameters
- Transport
 - Session := *session*
9. ONVIF Client invokes **RTSP PLAY** request over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
10. The DUT responds with **200 OK** message over WebSocket with parameters
- Session
 - RTP-Info
11. If DUT does not send *encoding* RTP media stream to ONVIF Client over RTSP control connection, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **RTSP TEARDOWN** request over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
14. The DUT responds with **200 OK** message over WebSocket with parameters
- Session

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response over WebSocket 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.

A.30 Media2 Service – Media Profile Configuration for Metadata Streaming

Name: HelperConfigureMediaProfileForMetadataStreaming

Procedure Purpose: Helper procedure to configure Media Profile to contain Video Source Configuration and Metadata Configuration.

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

Input: Transport protocol (*transportProtocol*), IP type (*ipType*) (optional parameter, IPv4 by default).

Returns: Media Profile (*profile*) containing Video Source Configuration, Metadata Configuration, PTZ Configuration (if found), and Analytics Configuration (if found). Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type := All
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*
3. For each Media Profile *profile* in *profileList* with both Configuration.VideoSource and Configuration.Metadata repeat the following steps:
 - 3.1. ONVIF Client invokes **GetMetadataConfigurationOptions** request with parameters
 - ConfigurationToken := *profile.Configuration.Metadata.@token*
 - ProfileToken := *profile.@token*
 - 3.2. DUT responds with **GetMetadataConfigurationOptionsResponse** message with parameters
 - Options =: *metadataOptions*

- 3.3. If *metadataOptions* does not contain Extension element or *metadataOptions.Extension* contains CompressionType with value equals to None, go to step 6.
4. For each Media Profile *profile* in *profileList* repeat the following steps:
 - 4.1. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
 - 4.2. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfList*
 - 4.3. For each Video Source Configuration *videoSourceConfiguration* in *videoSourceConfList* repeat the following steps:
 - 4.3.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfiguration.@token*
 - 4.3.2. The DUT responds with **AddConfigurationResponse** message.
 - 4.3.3. ONVIF Client invokes **GetMetadataConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
 - 4.3.4. The DUT responds with **GetMetadataConfigurationsResponse** with parameters
 - Configurations list =: *metadataConfList*
 - 4.3.5. For each Metadata Configuration *metadataConf* in *metadataConfList* repeat the following steps:
 - 4.3.5.1. ONVIF Client invokes **GetMetadataConfigurationOptions** request with parameters

- ConfigurationToken := *metadataConf.@token*
 - ProfileToken := *profile.@token*
- 4.3.5.2. DUT responds with **GetMetadataConfigurationOptionsResponse** message with parameters
- Options =: *metadataOptions*
- 4.3.5.3. If *metadataOptions* does not contain Extension element or *metadataOptions.Extension* contains CompressionType with value equals to None:
- 4.3.5.3.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := Metadata
 - Configuration[0].Token := *metadataConf.@token*
- 4.3.5.3.2. The DUT responds with **AddConfigurationResponse** message.
- 4.3.5.3.3. Go to step 6.
5. If cycle for step 4 ends without *metadataOptions* that does not contain Extension element or that contains *metadataOptions.Extension* with CompressionType with value equals to None, FAIL the test and skip other steps.
6. If the DUT supports PTZ Service and if *metadataOptions.PTZStatusFilterOptions.PanTiltStatusSupported* is equal to true or *metadataOptions.PTZStatusFilterOptions.ZoomStatusSupported* is equal to true or *metadataOptions.PTZStatusFilterOptions.PanTiltPositionSupported* is equal to true or *metadataOptions.PTZStatusFilterOptions.ZoomPositionSupported* is equal to true, ONVIF Client adds PTZ Configuration to a Media Profile by following the procedure mentioned in [Annex A.31](#) with the following input and output parameters
- in *profile* - Media Profile

7. If The DUT supports Analytics, ONVIF Client adds Analytics Configuration to a Media Profile by following the procedure mentioned in [Annex A.32](#) with the following input and output parameters
 - in *profile* - Media Profile
8. ONVIF Client sets the following:
 - *metadataConfiguration.@token* := *profile.Configurations.Metadata.@token*
 - *metadataConfiguration.Name* := *profile.Configurations.Metadata.Name*
 - *metadataConfiguration.UseCount* := *profile.Configurations.Metadata.UseCount*
 - if *mcOptions.Extention.CompressionType* skipped:
 - *metadataConfiguration.@CompressionType* skipped
 - If *mcOptions.Extention.CompressionType* is not skipped:
 - *metadataConfiguration.@CompressionType* := None
 - If *mcOptions.PTZStatusFilterOptions.PanTiltStatusSupported* is equal to false and *mcOptions.PTZStatusFilterOptions.ZoomStatusSupported* is equal to false and *mcOptions.PTZStatusFilterOptions.PanTiltPositionSupported* is equal to false and *mcOptions.PTZStatusFilterOptions.ZoomPositionSupported* is equal to false:
 - *metadataConfiguration.PTZStatus* skipped
 - If at least on element value within *mcOptions.PTZStatusFilterOptions* is equal to true:
 - *metadataConfiguration.PTZStatus.Status* := true if *mcOptions.PTZStatusFilterOptions.PanTiltStatusSupported* or *mcOptions.PTZStatusFilterOptions.ZoomStatusSupported* is equal to true. Otherwise *metadataConfiguration.PTZStatus.Status* := false
 - *metadataConfiguration.PTZStatus.Position* := true if *mcOptions.PTZStatusFilterOptions.PanTiltPositionSupported* or *mcOptions.PTZStatusFilterOptions.ZoomPositionSupported* is equal to true. Otherwise *metadataConfiguration.PTZStatus.Position* := false
 - *metadataConfiguration.Events*
 - *metadataConfiguration.Events.Filter* skipped
 - *metadataConfiguration.Events.SubscriptionPolicy* skipped

- If *profile.Configurations* contains Analytics:
 - *metadataConfiguration.Analytics* := true
 - If *profile.Configurations* does not contain Analytics:
 - *metadataConfiguration.Analytics* skipped
 - If *transportProtocol* is not equal to RtspMulticast:
 - *metadataConfiguration.Multicast* := *profile.Configurations.Metadata.Multicast*
 - If *transportProtocol* is equal to RtspMulticast:
 - If *ipType* is equal to IPv4:
 - *metadataConfiguration.Multicast.Address.Type* := IPv4
 - *metadataConfiguration.Multicast.Address.IPv4Address* := multicast IPv4 address
 - *metadataConfiguration.Multicast.Address.IPv6Address* skipped
 - If *ipType* is equal to IPv6:
 - *metadataConfiguration.Multicast.Address.Type* := IPv6
 - *metadataConfiguration.Multicast.Address.IPv4Address* skipped
 - *metadataConfiguration.Multicast.Address.IPv6Address* := multicast IPv6 address
 - *metadataConfiguration.Multicast.Port* := port for multicast streaming
 - *metadataConfiguration.Multicast.TTL* := 1
 - *metadataConfiguration.Multicast.AutoStart* := false
 - *metadataConfiguration.SessionTimeout* := *profile.Configurations.Metadata.SessionTimeout*
9. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
- Configuration := *metadataConfiguration*
 - Configuration.SessionTimeout := *profile.Configurations.Metadata.SessionTimeout*
10. The DUT responds with **SetMetadataConfigurationResponse** message.
11. if *protocol* = RtspMulticast:

11.1. Set *confTypeList* := (VideoEncoder, AudioEncoder)

11.2. ONVIF Client removes Video Encoder Configuration and Audio Encoder from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters

- in *confTypeList* - list of configuration type to remove from Media Profile
- in *profile* - Media Profile to update

12. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters

- in *transportProtocol* - Transport protocol
- in *ipType* - IP type
- in *profile.@token* - Media profile token
- out *streamUri* - Stream URI

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetMetadataConfigurationOptionsResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetMetadataConfigurationsResponse** message.
- DUT did not send **SetMetadataConfigurationResponse** message.

A.31 Media2 Service – Add PTZ Configuration to Media Profile

Name: HelperAddPTZConfiguration

Procedure Purpose: Helper procedure to configure Media Profile to contain PTZ Configuration.

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

Input: Media Profile (*profile*).

Returns: None.

Procedure:

1. If *profile.Configurations* does not contain PTZ:
 - 1.1. ONVIF Client invokes **GetCompatibleConfigurations** request with parameters
 - ProfileToken := *profile.@token*
 - 1.2. The DUT responds with **GetCompatibleConfigurationsResponse** message with parameters
 - PTZConfiguration list =: *ptzConfigurationList*
 - 1.3. If *ptzConfigurationList* is empty, skip other steps.
 - 1.4. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := PTZ
 - Configuration[0].Token := *ptzConfigurationList[0].@token*
 - 1.5. The DUT responds with **AddConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetCompatibleConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.

A.32 Media2 Service – Add Analytics Configuration to Media Profile

Name: HelperAddAnalyticsConfiguration

Procedure Purpose: Helper procedure to configure Media Profile to contain Analytics Configuration.

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

Input: Media Profile (*profile*)

Returns: None.

Procedure:

1. If *profile*.Configurations does not contain Analytics:
 - 1.1. ONVIF Client invokes **GetAnalyticsConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile*.@token
 - 1.2. The DUT responds with **GetAnalyticsConfigurationsResponse** message with parameters
 - Configurations list =: *acList*
 - 1.3. If *acList* is not empty:
 - 1.3.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile*.@token
 - Name skipped
 - Configuration[0].Type := Analytics
 - Configuration[0].Token := *acList*[0].@token
 - 1.3.2. The DUT responds with **AddConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetAnalyticsConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.

A.33 Metadata Streaming over RTP-Unicast/UDP

Name: HelperMetadataStreamingRTPUnicastUDP

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

Pre-requisite: None

Input: Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
5. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtmpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port1-port2*
6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
7. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters

- Session := *session*
8. The DUT responds with **200 OK** message with parameters
- Session
 - RTP-Info
9. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
- Configuration.@token := *metadataConfiguration.@token*
 - Configuration.Name := *metadataConfiguration.Name*
 - Configuration.UseCount := *metadataConfiguration.UseCount*
 - Configuration.@CompressionType := *metadataConfiguration.CompressionType*
 - Configuration.PTZStatus := *metadataConfiguration.PTZStatus*
 - Configuration.PTZStatus.Status := *metadataConfiguration.PTZStatus.Status*
 - Configuration.PTZStatus.Position := *metadataConfiguration.PTZStatus.Position*
 - Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
 - Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
 - Configuration.Events.Filter.MessageContent skipped
 - Configuration.Events.SubscriptionPolicy skipped
 - Configuration.Analytics := *metadataConfiguration.Analytics*
 - Configuration.Multicast.Address.Type := *metadataConfiguration.Multicast.Address.Type*
 - Configuration.Multicast.Address.IPv4Address := *metadataConfiguration.Multicast.Address.IPv4Address*
 - Configuration.Multicast.Address.IPv6Address := *metadataConfiguration.Multicast.Address.IPv6Address*
 - Configuration.Multicast.Port := *metadataConfiguration.Multicast.Port*
 - Configuration.Multicast.TTL := *metadataConfiguration.Multicast.TTL*
 - Configuration.Multicast.AutoStart := *metadataConfiguration.Multicast.AutoStart*

- Configuration.SessionTimeout := *metadataConfiguration.SessionTimeout*
10. The DUT responds with **SetMetadataConfigurationResponse** message.
11. If DUT does not send Metadata RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
14. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.34 Metadata Streaming over RTP-Unicast/RTSP/HTTP/TCP

Name: HelperMetadataStreamingRTPUnicastRTSPHTTPTCP

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

Pre-requisite: None

Input: Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

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*
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtmpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;client_port=*port1-port2*
8. The DUT responds with **200 OK** message on *connection1* with parameters
 - Transport
 - Session =: *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters

- Session := *session*
10. The DUT responds with **200 OK** message on *connection1* with parameters
- Session
 - RTP-Info
11. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
- Configuration.@token := *metadataConfiguration.@token*
 - Configuration.Name := *metadataConfiguration.Name*
 - Configuration.UseCount := *metadataConfiguration.UseCount*
 - Configuration.@CompressionType := *metadataConfiguration.CompressionType*
 - Configuration.PTZStatus := *metadataConfiguration.PTZStatus*
 - Configuration.PTZStatus.Status := *metadataConfiguration.PTZStatus.Status*
 - Configuration.PTZStatus.Position := *metadataConfiguration.PTZStatus.Position*
 - Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
 - Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
 - Configuration.Events.Filter.MessageContent skipped
 - Configuration.Events.SubscriptionPolicy skipped
 - Configuration.Analytics := *metadataConfiguration.Analytics*
 - Configuration.Multicast.Address.Type := *metadataConfiguration.Multicast.Address.Type*
 - Configuration.Multicast.Address.Ipv4Address := *metadataConfiguration.Multicast.Address.Ipv4Address*
 - Configuration.Multicast.Address.Ipv6Address := *metadataConfiguration.Multicast.Address.Ipv6Address*
 - Configuration.Multicast.Port := *metadataConfiguration.Multicast.Port*
 - Configuration.Multicast.TTL := *metadataConfiguration.Multicast.TTL*
 - Configuration.Multicast.AutoStart := *metadataConfiguration.Multicast.AutoStart*

- Configuration.SessionTimeout := *metadataConfiguration.SessionTimeout*
12. The DUT responds with **SetMetadataConfigurationResponse** message.
 13. If DUT does not send Metadata RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.
 14. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 15. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
 16. ONVIF Client closes *connection2*.
 17. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.35 Metadata Streaming over RTP/RTSP/TCP

Name: HelperMetadataStreamingRTPRTSPTCP

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

Pre-requisite: None

Input: Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
5. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtmpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;interleaved=0-1
6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
7. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
9. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
 - Configuration.@token := *metadataConfiguration.@token*

- Configuration.Name := *metadataConfiguration*.Name
- Configuration.UseCount := *metadataConfiguration*.UseCount
- Configuration.@CompressionType := *metadataConfiguration*.CompressionType
- Configuration.PTZStatus := *metadataConfiguration*.PTZStatus
- Configuration.PTZStatus.Status := *metadataConfiguration*.PTZStatus.Status
- Configuration.PTZStatus.Position := *metadataConfiguration*.PTZStatus.Position
- Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
- Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
- Configuration.Events.Filter.MessageContent skipped
- Configuration.Events.SubscriptionPolicy skipped
- Configuration.Analytics := *metadataConfiguration*.Analytics
- Configuration.Multicast.Address.Type := *metadataConfiguration*.Multicast.Address.Type
- Configuration.Multicast.Address.IPv4Address :=
metadataConfiguration.Multicast.Address.IPv4Address
- Configuration.Multicast.Address.IPv6Address :=
metadataConfiguration.Multicast.Address.IPv6Address
- Configuration.Multicast.Port := *metadataConfiguration*.Multicast.Port
- Configuration.Multicast.TTL := *metadataConfiguration*.Multicast.TTL
- Configuration.Multicast.AutoStart := *metadataConfiguration*.Multicast.AutoStart
- Configuration.SessionTimeout := *metadataConfiguration*.SessionTimeout

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

11. If DUT does not send Metadata RTP media stream to ONVIF Client over RTSP control connection, FAIL the test and skip other steps.

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

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

- Session := *session*

14. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.36 Metadata Streaming over RTP-Unicast/UDP

Name: HelperMetadataStreamingRTPMulticastUDP

Procedure Purpose: Helper procedure to verify metadata streaming over RTP-Multicast/UDP.

Pre-requisite: None

Input: Uri for media streaming (*streamUri*), Expected media stream encoding (*encoding*), IP version (*ipVersion*), Metadata Configuration (*metadataConfiguration*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*

- SDP information =: *sdp*
3. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
 4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
 5. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtmpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;multicast;client_port=*port1-port2*
 6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 7. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 9. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
 - Configuration.@token := *metadataConfiguration.@token*
 - Configuration.Name := *metadataConfiguration.Name*
 - Configuration.UseCount := *metadataConfiguration.UseCount*
 - Configuration.@CompressionType := *metadataConfiguration.CompressionType*

- Configuration.PTZStatus := *metadataConfiguration*.PTZStatus
 - Configuration.PTZStatus.Status := *metadataConfiguration*.PTZStatus.Status
 - Configuration.PTZStatus.Position := *metadataConfiguration*.PTZStatus.Position
 - Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
 - Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
 - Configuration.Events.Filter.MessageContent skipped
 - Configuration.Events.SubscriptionPolicy skipped
 - Configuration.Analytics := *metadataConfiguration*.Analytics
 - Configuration.Multicast.Address.Type := *metadataConfiguration*.Multicast.Address.Type
 - Configuration.Multicast.Address.IPv4Address := *metadataConfiguration*.Multicast.Address.IPv4Address
 - Configuration.Multicast.Address.IPv6Address := *metadataConfiguration*.Multicast.Address.IPv6Address
 - Configuration.Multicast.Port := *metadataConfiguration*.Multicast.Port
 - Configuration.Multicast.TTL := *metadataConfiguration*.Multicast.TTL
 - Configuration.Multicast.AutoStart := *metadataConfiguration*.Multicast.AutoStart
 - Configuration.SessionTimeout := *metadataConfiguration*.SessionTimeout
10. The DUT responds with **SetMetadataConfigurationResponse** message.
11. If DUT does not send Metadata RTP *ipVersion* multicast media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
- Session := *session*
14. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.37 Provide CA certificate

Name: HelperCreateCACertificate

Procedure Purpose: Helper procedure to create an X.509 CA certificate.

Pre-requisite: None

Input: The subject (*subject*) of certificate(optional input parameter, could be skipped).

Returns: An X.509 CA certificate (*CAcert*) that is compliant to [RFC5280] and a corresponding private key (*privateKey*) and public key (*publicKey*).

Procedure:

1. ONVIF Client determines the length of the key to generate (out *length*) by following the procedure mentioned in [Annex A.56](#).
2. If *subject* is skipped set:
 - *subject* := "CN=ONVIF TT,C=US"
3. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC5280] and has the following properties:
 - version := v3
 - signature := sha1-WithRSAEncryption

- validity := not before 19700101000000Z and not after 99991231235959Z
- subject := *subject*
- length of the key to be used := *length*

Procedure Result:**PASS –**

- None.

FAIL –

- None.

Note: ONVIF Client may return the same CA certificate in subsequent invocations of this procedure for the same subject.

A.38 Get Stream Uri

Name: HelperGetStreamUri

Procedure Purpose: Helper procedure to get stream URI from the DUT.

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

Input: Protocol (*protocol*). Media Profile token (*token*). IP type (*ipType*) (optional parameter, IPv4 by default).

Returns: Stream Uri (*streamUri*).

Procedure:

1. ONVIF Client invokes **GetStreamUri** request with parameters
 - Protocol := *protocol*
 - ProfileToken := *token*
2. The DUT responds with **GetStreamUriResponse** message with parameters
 - Uri =: *streamUri*
3. If *streamUri* is longer than 128 octets, FAIL the test and skip other steps.

4. If *ipType* skipped, set *ipType* := IPv4.
5. If *streamUri* ip type is not equal to *ipType*, FAIL the test and skip other steps.
6. If *protocol* = RtspsOverHttp:
 - 6.1. If *streamUri* doesn't have the same port with the web service, FAIL the test and skip other steps.
 - 6.2. If *streamUri* doesn't have the same scheme with the web service ('http' or 'https'), FAIL the test and skip other steps.
7. If *protocol* != RtspsOverHttp:
 - 7.1. If *streamUri* doesn't have scheme equal to 'rtsp', FAIL the test and skip other steps.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.39 Device Configuration for Video Streaming

Name: HelperDeviceConfigurationForVideoStreaming

Procedure Purpose: Helper procedure to configure Media profile, Video Encoder Configuration, and get stream URI from the DUT for video streaming.

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

Input: Required video encoding (*requiredVideoEncoding*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: Stream Uri (*streamUri*).

Procedure:

1. 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 *requiredVideoEncoding* - 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
2. if *protocol* = RtspMulticast:
- 2.1. ONVIF Client removes Audio Encoder Configuration and Metadata Configuration from media profile by following the procedure mentioned in [Annex A.9](#) with the following input and output parameters
- in *profile* - Media Profile
3. 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 (or skipped if *vecOptions*.@GovLengthRange skipped)
 - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline (or skipped if *vecOptions*.@ProfilesSupported skipped)
 - Configuration.Encoding := *requiredVideoEncoding*
 - Configuration.Resolution := resolution closest to 640x480 from *vecOptions*.ResolutionsAvailable list
 - if *vecOptions*.@FrameRatesSupported skipped and *profile*.Configurations.VideoEncoder.RateControl skipped:
 - Configuration.RateControl skipped
 - if *vecOptions*.@FrameRatesSupported or *profile*.Configurations.VideoEncoder.RateControl is not skipped:
 - Configuration.RateControl.@ConstantBitRate := *vecOptions*.@ConstantBitRateSupported
 - Configuration.RateControl.FrameRateLimit := value closest to 25 but greater than 1 from *vecOptions*.@FrameRatesSupported

- list (or *profile.Configurations.VideoEncoder.RateControl.FrameRateLimit* if *vecOptions.@FrameRatesSupported* skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile.Configurations.VideoEncoder.RateControl.BitrateLimit*, *vecOptions.BitrateRange.Min*}, *vecOptions.BitrateRange.Max*}
 - if *protocol* is not equal to *RtspMulticast*:
 - Configuration.Multicast := *profile.Configurations.VideoEncoder.Multicast*
 - if *protocol* = *RtspMulticast* and *ipVersion* = IPv4:
 - 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
 - if *protocol* = *RtspMulticast* and *ipVersion* = IPv6:
 - 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*
4. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
 5. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters
 - in *protocol* - Transport protocol

- in *ipVersion* - IP Type
- in *profile.@token* - Media profile token
- out *uri* - Stream URI

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

A.40 Device Configuration for Audio Streaming

Name: HelperDeviceConfigurationForAudioStreaming

Procedure Purpose: Helper procedure to configure Media profile, Audio Encoder Configuration, and get stream URI from the DUT for audio streaming.

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

Input: Required audio encoding (*requiredAudioEncoding*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: Stream Uri (*streamUri*), Audio encoding set in profile (*requiredAudioEncoding*).

Procedure:

1. 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 *requiredAudioEncoding* - 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
2. if *protocol* = RtspMulticast:

- 2.1. ONVIF Client removes Video Encoder Configuration and Metadata Configuration from media profile by following the procedure mentioned in [Annex A.8](#) with the following input and output parameters
 - in *profile* - Media Profile
3. 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 := *aecOptions*.Encoding
 - if *protocol* is not equal to RtpMulticast:
 - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
 - if *protocol* = RtpMulticast and *ipVersion* = IPv4:
 - 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
 - if *protocol* = RtpMulticast and *ipVersion* = IPv6:
 - 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 *aecOptionsBitrateList.Items* list
 - Configuration.SampleRate := the nearest value to *profile.Configurations.AudioEncoder.SampleRate* from *aecOptionsSampleRateList.Items* list
4. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
 5. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters
 - in *protocol* - Transport protocol
 - in *ipVersion* - IP Type
 - in *profile.@token* - Media profile token
 - out *uri* - Stream URI

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

A.41 Backchannel Streaming over WebSocket

Name: HelperBackchannelStreamingOverWebSocket

Procedure Purpose: Helper procedure to verify audio backchannel streaming over WebSocket.

Pre-requisite: WebSocket is supported by the DUT. Audio Backchannel is supported by DUT.

Input: Uri for audio backchannel streaming (*streamUri*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client retrieves Media2 Service capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap* - Media2 Service capabilities
2. Set *uri* := *cap*.StreamingCapabilities.RTSPWebSocketUri
3. If scheme component of *uri* is not equal to **ws** or **wss**, FAIL the test and skip other steps.
4. ONVIF Client establishes a WebSocket Connection by following the procedure mentioned in [Annex A.26](#) with the following input and output parameters
 - in *uri* - Web Socket Uri
5. ONVIF Client invokes **RTSP DESCRIBE** request with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address over WebSocket.
6. The DUT responds with **200 OK** message over WebSocket with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
7. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
8. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
9. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag over WebSocket to uri address, which corresponds to audio backchannel media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;interleaved=0-1
10. The DUT responds with **200 OK** message over WebSocket with parameters
 - Transport
 - Session =: *session*

11. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
12. The DUT responds with **200 OK** message over WebSocket with parameters
 - Session
 - RTP-Info
13. ONVIF Client sends RTP Unicast audio stream with *encoding* to DUT over UDP.
14. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
15. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
16. The DUT responds with **200 OK** message over WebSocket with parameters
 - Session

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- DUT did not send **RTSP 200 OK** response over WebSocket 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 7.

A.42 Configuring HTTPS if Required

Name: HelperCheckAndConfigureHTTPS

Procedure Purpose: Helper Procedure to check and configure HTTPS using Security Configuration Service if required.

Pre-requisite: RTP/RTSP/HTTPS feature is supported by DUT. HTTPS is configured on the DUT, if TLS Server is not supported by DUT. Security Configuration Service is received from the DUT, if TLS Server is supported by DUT.

Input: None.

Returns: None.

Procedure:

1. ONVIF Client invokes **GetNetworkProtocols** request.
2. The DUT responds with **GetNetworkProtocolsResponse** with parameters
 - NetworkProtocols list =: *networkProtocolsList*
3. If *networkProtocolsList* contains item with Name = HTTPS and Enabled = true, return to the test and skip other procedure steps.
4. If the DUT does not support TLS Server, FAIL the test and skip other steps.
5. ONVIF Client configures HTTPS by following the procedure mentioned in [Annex A.15](#).

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.43 Removing Configurations from Media Profile

Name: HelperRemoveConfigurationsFromMediaProfile

Procedure Purpose: Helper Procedure to remove configurations from Media Profile.

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

Input: Media Profile (*profile*). List of configuration type to remove from profile (*confTypeList*).

Returns: None.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token := *profile.@token*
 - Type[0] := All
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*
3. If *profileList*[0] contains at least one Configuration with type equals to configuration type from *confTypeList*:
 - 3.1. ONVIF Client invokes **RemoveConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - If *profileList*[0] contains Configuration.VideoSource and *confTypeList* contains VideoSource:
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token skipped
 - If *profileList*[0] contains Configuration.VideoEncoder and *confTypeList* contains VideoEncoder:
 - Configuration[1].Type := VideoEncoder
 - Configuration[1].Token skipped
 - If *profileList*[0] contains Configuration.AudioSource and *confTypeList* contains AudioSource:
 - Configuration[2].Type := AudioSource
 - Configuration[2].Token skipped
 - If *profileList*[0] contains Configuration.AudioEncoder and *confTypeList* contains AudioEncoder:
 - Configuration[3].Type := AudioEncoder

- Configuration[3].Token skipped
- If *profileList*[0] contains Configuration.AudioOutput and *confTypeList* contains AudioOutput:
 - Configuration[4].Type := AudioOutput
 - Configuration[4].Token skipped
- If *profileList*[0] contains Configuration.AudioDecoder and *confTypeList* contains AudioDecoder:
 - Configuration[5].Type := AudioDecoder
 - Configuration[5].Token skipped
- If *profileList*[0] contains Configuration.Metadata and *confTypeList* contains Metadata:
 - Configuration[6].Type := Metadata
 - Configuration[6].Token skipped
- If *profileList*[0] contains Configuration.Analytics and *confTypeList* contains Analytics:
 - Configuration[7].Type := Analytics
 - Configuration[7].Token skipped
- If *profileList*[0] contains Configuration.PTZ and *confTypeList* contains PTZ:
 - Configuration[8].Type := PTZ
 - Configuration[8].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.44 Device Configuration for Audio Streaming using Media Profile that contains only Audio Configurations

Name: HelperDeviceConfigurationForAudioStreamingUsingOnlyAudioInProfile

Procedure Purpose: Helper procedure to configure Media profile that contains only Audio Source and Audio Encoder configurations, to configure Audio Encoder Configuration, and get stream URI from the DUT for audio streaming.

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

Input: Required audio encoding (*requiredAudioEncoding*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: Stream Uri (*streamUri*).

Procedure:

1. 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 *requiredAudioEncoding* - 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
2. 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 := *requiredAudioEncoding*
 - if *protocol* is not equal to RtspMulticast:
 - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
 - if *protocol* = RtspMulticast and *ipVersion* = IPv4:
 - 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
 - if *protocol* = RtspMulticast and *ipVersion* = IPv6:
 - 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.Bitrates := the nearest value to *profile*.Configurations.AudioEncoder.Bitrates from *aecOptionsBitratesList.Items* list
 - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptionsSampleRateList.Items* list
3. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
 4. Set *confTypeList* := (VideoSource, VideoEncoder, AudioOutput, AudioDecoder, Metadata, Analytics, PTZ)
 5. ONVIF Client removes all configurations except AudioSource and AudioEncoder from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters
 - in *confTypeList* - list of configuration type to remove from Media Profile
 - in *profile* - Media Profile to update
 6. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters
 - in *protocol* - Transport protocol

- in *ipVersion* - IP Type
- in *profile.@token* - Media profile token
- out *uri* - Stream URI

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

A.45 Device Configuration for Video and Audio Streaming

Name: HelperDeviceConfigurationForVideoAndAudioStreaming

Procedure Purpose: Helper procedure to configure Media profile, Video Encoder Configuration, Audio Encoder Configuration, and get stream URI from the DUT for video and audio streaming.

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

Input: Required audio encoding (*requiredAudioEncoding*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: Stream Uri (*streamUri*). Media profile with required configurations (*profile*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := All
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*
3. For each Media Profile *profile* in *profileList* with not empty Configuration.VideoSource, Configuration.VideoEncoder, Configuration.AudioSource, and Configuration.AudioEncoder repeat the following steps:
 - 3.1. If *profile*.Configuration.VideoEncoder = H264 or H265

- 3.1.1. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** request with parameters
 - ConfigurationToken := *profile*.Configuration.AudioEncoder.@token
 - ProfileToken := *profile*.@token
- 3.1.2. DUT responds with **GetAudioEncoderConfigurationOptionsResponse** message with parameters
 - Options list =: *optionsList*
- 3.1.3. If *requiredAudioEncoding* = AAC:
 - 3.1.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
 - 3.1.3.1.1. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if exists, otherwise item with Encoding = "MPEG4-GENERIC".
- 3.1.4. If *requiredAudioEncoding* != AAC:
 - 3.1.4.1. If *optionsList* list contains item with Encoding = *requiredAudioEncoding*:
 - 3.1.4.1.1. Set *aecOptions* := item with Encoding = *requiredAudioEncoding* from *optionsList* list.
- 3.1.5. If *aecOptions* != NULL
 - 3.1.5.1. ONVIF Client sets audio encoder configuration by following the procedure mentioned in [Annex A.62](#) with the following input and output parameters
 - in *profile* - Media profile
 - in *aecOptions* - audio encoder configuration options
 - in *protocol* - Transport protocol
 - in *ipVersion* - IP Type
 - 3.1.5.2. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *profile*.Configuration.VideoEncoder.@token
- ProfileToken := *profile*.@token

3.1.5.3. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

3.1.5.4. Set *vecOptions* := item with Encoding = *profile*.Configuration.VideoEncoder.Encoding from *optionsList* list.

3.1.5.5. ONVIF Client sets video encoder configuration by following the procedure mentioned in [Annex A.61](#) with the following input and output parameters

- in *profile* - Media profile
- in *vecOptions* - video encoder configuration options
- in *protocol* - Transport protocol
- in *ipVersion* - IP Type

3.1.5.6. Go to step 10.

4. Set *profile* = *profileList*[0].
5. Set *confTypeList* := (configurations that are contained in profile *profile*)
6. ONVIF Client removes all configurations from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters
 - in *confTypeList* - list of configuration type to remove from Media Profile
 - in *profile* - Media Profile to update
7. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile*.@token
8. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfList*

9. For each Video Source Configuration *videoSourceConfiguration* in *videoSourceConfList* repeat the following steps:
 - 9.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfiguration.@token*
 - 9.2. The DUT responds with **AddConfigurationResponse** message.
 - 9.3. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
 - 9.4. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList*
 - 9.5. For each Video Encoder Configuration *videoEncoderConfiguration* in *videoEncoderConfList* repeat the following steps:
 - 9.5.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
 - ConfigurationToken := *videoEncoderConfiguration.@token*
 - ProfileToken := *profile.@token*
 - 9.5.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
 - Options list =: *optionsList*
 - 9.5.3. If *optionsList* list contains item with Encoding = H264 or H265:
 - 9.5.3.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*

- Name skipped
 - Configuration[0].Type := VideoEncoder
 - Configuration[0].Token := *videoEncoderConfiguration.@token*
- 9.5.3.2. The DUT responds with **AddConfigurationResponse** message.
- 9.5.3.3. Set *vecOptions* := item with Encoding = H264 from *optionsList* list if present, otherwise item with Encoding = H265.
- 9.5.3.4. ONVIF Client sets video encoder configuration by following the procedure mentioned in [Annex A.61](#) with the following input and output parameters
- in *profile* - Media profile
 - in *vecOptions* - video encoder configuration options
 - in *protocol* - Transport protocol
 - in *ipVersion* - IP Type
- 9.5.3.5. ONVIF Client tries to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding support to the Media Profile by following the procedure mentioned in [Annex A.46](#) with the following input and output parameters
- in *requiredAudioEncoding* - required audio encoding
 - in *profile* - Media profile
 - out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile
- 9.5.3.6. If *aecOptions* != NULL
- 9.5.3.6.1. ONVIF Client sets audio encoder configuration by following the procedure mentioned in [Annex A.62](#) with the following input and output parameters

- in *profile* - Media profile
- in *aecOptions* - audio encoder configuration options
- in *protocol* - Transport protocol
- in *ipVersion* - IP Type

9.5.3.6. Go to step 10.

9.5.3.7. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile.@token*
- Configuration[0].Type = VideoEncoder
- Configuration[0].Token skipped

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

10. If step 9 ends with *aecOptions* = NULL, fail the test, restore DUT settings, and skip other steps.

11. if *protocol* = RtspMulticast:

11.1. Set *confTypeList* := (Metadata)

11.2. ONVIF Client removes Metadata Configuration from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters

- in *confTypeList* - list of configuration type to remove from Media Profile
- in *profile* - Media Profile to update

12. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters

- in *protocol* - Transport protocol
- in *ipVersion* - IP Type
- in *profile.@token* - Media profile token

- out *uri* - Stream URI

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

A.46 Media2 Service – Adding AudioSource and AudioEncoder with Specified Audio Encoder Value to Media Profile

Name: HelperAddAudioConfigurationsWithSpecificEncoderToMediaProfile

Procedure Purpose: Helper procedure to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding to the Media Profile if corresponding AudioEncoder Configuration found.

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

Input: Required audio encoding (*requiredAudioEncoding*). Media Profile (*profile*)

Returns: Audio Encoder Configuration Options for the Media Profile (*aecOptions*) (optional, returned in case profile was configured with audio).

Procedure:

1. ONVIF Client invokes **GetAudioSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
2. The DUT responds with **GetAudioSourceConfigurationsResponse** with parameters
 - Configurations list =: *audioSourceConfList*
3. For each Audio Source Configuration *audioSourceConfiguration* in *audioSourceConfList* repeat the following steps:
 - 3.1. ONVIF Client invokes **AddConfiguration** request with parameters

- ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := AudioSource
 - Configuration[0].Token := *audioSourceConfiguration.@token*
- 3.2. The DUT responds with **AddConfigurationResponse** message.
- 3.3. ONVIF Client invokes **GetAudioEncoderConfigurations** request with parameters
- ConfigurationToken skipped
 - ProfileToken := *profile.@token*
- 3.4. The DUT responds with **GetAudioEncoderConfigurationsResponse** with parameters
- Configurations list =: *audioEncoderConfList*
- 3.5. For each Audio Encoder Configuration *audioEncoderConfiguration* in *audioEncoderConfList* repeat the following steps:
- 3.5.1. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** request with parameters
- ConfigurationToken := *audioEncoderConfiguration.@token*
 - ProfileToken := *profile.@token*
- 3.5.2. DUT responds with **GetAudioEncoderConfigurationOptionsResponse** message with parameters
- Options list =: *optionsList*
- 3.5.3. If *requiredAudioEncoding* = AAC:
- 3.5.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
- 3.5.3.1.1. ONVIF Client invokes **AddConfiguration** request with parameters
- ProfileToken := *profile.@token*

- Name skipped
- Configuration[0].Type := AudioEncoder
- Configuration[0].Token := *audioEncoderConfiguration.@token*

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

3.5.3.1.3. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if exists, otherwise item with Encoding = "MPEG4-GENERIC".

3.5.3.1.4. Skip other steps in procedure.

3.5.4. If *requiredAudioEncoding* != AAC:

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

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

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

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

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

3.5.4.1.4. Skip other steps in procedure.

3.6. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile.@token*
- Configuration[0].Type = AudioSource

- Configuration[0].Token skipped

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.47 RTSP Authentication Check

Name: HelperRTSPAuthenticationCheck

Procedure Purpose: Helper procedure to check that DUT uses Digest authentication type for RTSP.

Pre-requisite: Real-time Streaming is supported by DUT.

Input: None.

Returns: None.

Procedure:

1. If Device supports Profile T according to feature definition results:
 - 1.1. If DUT does not send RTSP 401 Unauthorized message to any RTSP anonymous request, FAIL the test and skip other steps.
 - 1.2. If RTSP 401 Unauthorized message does not have **WWW-Authenticate: Digest** header, FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send response message to RTSP request.

A.48 Metadata Streaming over RTP-Unicast/RTSP/HTTPS/ TCP

Name: HelperMetadataStreamingRTPUnicastRTSPHTTPSTCP

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

Pre-requisite: HTTPS is configured on the DUT.

Input: Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

Returns: None

Procedure:

1. ONVIF Client invokes **HTTP GET** request to *streamUri* address to establish DUT to ONVIF Client secured connection for RTP data transfer (*connection1*).
2. ONVIF Client invokes **HTTP 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*
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming

7. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtptime (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP/TCP;unicast;client_port=*port1-port2*
8. The DUT responds with **200 OK** message on *connection1* with parameters
 - Transport
 - Session := *session*
9. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message on *connection1* with parameters
 - Session
 - RTP-Info
11. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
 - Configuration.@token := *metadataConfiguration.@token*
 - Configuration.Name := *metadataConfiguration.Name*
 - Configuration.UseCount := *metadataConfiguration.UseCount*
 - Configuration.@CompressionType := *metadataConfiguration.CompressionType*
 - Configuration.PTZStatus := *metadataConfiguration.PTZStatus*
 - Configuration.PTZStatus.Status := *metadataConfiguration.PTZStatus.Status*
 - Configuration.PTZStatus.Position := *metadataConfiguration.PTZStatus.Position*
 - Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
 - Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
 - Configuration.Events.Filter.MessageContent skipped
 - Configuration.Events.SubscriptionPolicy skipped
 - Configuration.Analytics := *metadataConfiguration.Analytics*

- Configuration.Multicast.Address.Type := *metadataConfiguration.Multicast.Address.Type*
- Configuration.Multicast.Address.IPv4Address :=
metadataConfiguration.Multicast.Address.IPv4Address
- Configuration.Multicast.Address.IPv6Address :=
metadataConfiguration.Multicast.Address.IPv6Address
- Configuration.Multicast.Port := *metadataConfiguration.Multicast.Port*
- Configuration.Multicast.TTL := *metadataConfiguration.Multicast.TTL*
- Configuration.Multicast.AutoStart := *metadataConfiguration.Multicast.AutoStart*
- Configuration.SessionTimeout := *metadataConfiguration.SessionTimeout*

12. The DUT responds with **SetMetadataConfigurationResponse** message.

13. If DUT does not send Metadata RTP media stream to ONVIF Client over *connection1*, FAIL the test and skip other steps.

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

15. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters

- Session := *session*

16. ONVIF Client closes *connection2*.

17. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.49 Metadata Streaming over WebSocket

Name: HelperMetadataStreamingWebSocket

Procedure Purpose: Helper procedure to verify metadata streaming over WebSocket.

Pre-requisite: WebSocket is supported by the DUT.

Input: Uri for media streaming (*streamUri*), Metadata Configuration (*metadataConfiguration*).

Returns: None

Procedure:

1. ONVIF Client retrieves Media2 Service capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap* - Media2 Service capabilities
2. Set *uri* := *cap*.StreamingCapabilities.RTSPWebSocketUri
3. If scheme component of *uri* is not equal to **ws** or **wss**, FAIL the test and skip other steps.
4. ONVIF Client establishes a WebSocket Connection by following the procedure mentioned in [Annex A.26](#) with the following input and output parameters
 - in *uri* - Web Socket Uri
5. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri* address over WebSocket.
6. The DUT responds with **200 OK** message over WebSocket with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
7. If *sdp* does not contain Media Type = application and with 'vnd.onvif.metadata' encoding name in a=rtmpmap and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
8. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming

9. ONVIF Client invokes **RTSP SETUP** request over WebSocket to uri address, which corresponds to 'application' media type with 'vnd.onvif.metadata' encoding name in a=rtpmap (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port1-port2*
10. The DUT responds with **200 OK** message over WebSocket with parameters
 - Transport
 - Session =: *session*
11. ONVIF Client invokes **RTSP PLAY** request over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
12. The DUT responds with **200 OK** message over WebSocket with parameters
 - Session
 - RTP-Info
13. ONVIF Client invokes **SetMetadataConfiguration** request with parameters
 - Configuration.@token := *metadataConfiguration.@token*
 - Configuration.Name := *metadataConfiguration.Name*
 - Configuration.UseCount := *metadataConfiguration.UseCount*
 - Configuration.@CompressionType := *metadataConfiguration.CompressionType*
 - Configuration.PTZStatus := *metadataConfiguration.PTZStatus*
 - Configuration.PTZStatus.Status := *metadataConfiguration.PTZStatus.Status*
 - Configuration.PTZStatus.Position := *metadataConfiguration.PTZStatus.Position*
 - Configuration.Events.Filter.TopicExpression.Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
 - Configuration.Events.Filter.TopicExpression := "tns1:Media/ConfigurationChanged"
 - Configuration.Events.Filter.MessageContent skipped
 - Configuration.Events.SubscriptionPolicy skipped
 - Configuration.Analytics := *metadataConfiguration.Analytics*

- Configuration.Multicast.Address.Type := *metadataConfiguration.Multicast.Address.Type*
- Configuration.Multicast.Address.IPv4Address :=
metadataConfiguration.Multicast.Address.IPv4Address
- Configuration.Multicast.Address.IPv6Address :=
metadataConfiguration.Multicast.Address.IPv6Address
- Configuration.Multicast.Port := *metadataConfiguration.Multicast.Port*
- Configuration.Multicast.TTL := *metadataConfiguration.Multicast.TTL*
- Configuration.Multicast.AutoStart := *metadataConfiguration.Multicast.AutoStart*
- Configuration.SessionTimeout := *metadataConfiguration.SessionTimeout*

14. The DUT responds with **SetMetadataConfigurationResponse** message.

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

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

17. ONVIF Client invokes **RTSP TEARDOWN** request over WebSocket to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters

- Session := *session*

18. The DUT responds with **200 OK** message over WebSocket 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.50 Remove all non-fixed Media Profiles and remove all configurations from fixed Media Profiles

Name: HelperMediaProfilesCleanUp

Procedure Purpose: Helper procedure, which removes all non-fixed Media Profiles and removes all configurations from fixed Media Profiles.

Pre-requisite: Media2 Service is supported by the DUT.

Input: Media Profiles List (*profileList*).

Returns: None.

Procedure:

1. For each Media Profile *profile1* in *profileList* repeat the following steps:
 - 1.1. If *profile1*.@fixed = true:
 - 1.1.1. ONVIF Client invokes **RemoveConfiguration** request with parameters
 - ProfileToken := *profile1*.@token
 - Configuration[0].Type := All
 - Configuration[0].Token skipped
 - 1.1.2. The DUT responds with **RemoveConfigurationResponse** message.
 - 1.2. If *profile1*.@fixed = false or skipped:
 - 1.2.1. ONVIF Client invokes **DeleteProfile** request with parameters
 - Token := *profile1*.@token
 - 1.2.2. The DUT responds with **DeleteProfileResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.51 Get Video Source Configurations List

Name: HelperGetVideoSourceConfigurationsList

Procedure Purpose: Helper procedure to retrieve Video Source Configurations List.

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

Input: None.

Returns: Video Source Configurations list (*videoSourceConfList*).

Procedure:

1. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken skipped
2. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfList*
3. If *videoSourceConfList* is empty, FAIL the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.52 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration

Name: HelperFindGuaranteedProfiles4

Procedure Purpose: Helper procedure, which tries to create new Media Profiles to reach number of guaranteed encoder instances without changing already configured profiles.

Pre-requisite: Media2 Service is supported by the DUT. Profile T is supported by the DUT.

Input: Information about guaranteed encoder instances (*info*). Video Source Configuration (*videoSourceConfig*). List of configured media profiles (*configuredProfilesList*).

Returns: List of configured Media Profiles (*configuredProfilesList*).

Procedure:

1. Set *configuredProfilesListForVSC1* := empty.
2. If number of items in *configuredProfilesListForVSC1* is equal to *info.Total*, skip other steps of procedure.
3. ONVIF Client invokes **CreateProfile** request with parameters
 - Name := "testMedia"
 - Configuration list - skipped
4. DUT responds with **CreateProfileResponse** message with parameters
 - Token =: *clearProfileToken1*
5. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *clearProfileToken1*
6. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfigurationList1*
7. If *videoSourceConfigurationList1* does not contain item with @token = *videoSourceConfig.@token*, FAIL the test and skip other steps.
8. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *clearProfileToken1*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfig.@token*
9. The DUT responds with **AddConfigurationResponse** message.
10. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken skipped

- ProfileToken := *clearProfileToken1*
11. The DUT responds with compatible video encoder configurations in **GetVideoEncoderConfigurationsResponse** with parameters
- Configurations list =: *videoEncoderConfList1*
12. If *videoEncoderConfList1* is empty, FAIL the test and skip other steps.
13. If *videoEncoderConfList1* contains only items that were used in Media Profiles from *configuredProfilesList* list, FAIL the test and skip other steps.
14. For each Video Encoder Configuration *videoEncoderConf1* from *videoEncoderConfList1*, which was not used in Media Profiles from *configuredProfilesList* list repeat the following steps:
- 14.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
- ConfigurationToken := *videoEncoderConf1.@token*
 - ProfileToken := *clearProfileToken1*
- 14.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
- Options list =: *optionsList1*
- 14.3. For each Video Encoder Options *vecOptions1* in *optionsList1* repeat the following steps:
- 14.3.1. If *info.Codec* list contains no items with Encoding = *vecOptions1.Encoding* or if *info.Codec* list contains an item with Encoding = *vecOptions1.Encoding* and number of Media Profiles with Configurations.VideoEncoder.Encoding = *vecOptions1.Encoding* in *configuredProfilesListForVSC1* is less than *info.Codec.Number* for this *vecOptions1.Encoding*:
- 14.3.1.1. Set *options1* := *vecOptions1*
- 14.3.1.2. Set *videoEncoderConfToAdd1* := *videoEncoderConf1*
- 14.3.1.3. Go to step 16.
- 14.4. Go to the next Video Encoder Configuration for the step 14.
15. FAIL the test and skip other steps.

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

- Configuration.@token := *videoEncoderConfToAdd1.@token*
- Configuration.Name := *videoEncoderConfToAdd1.Name*
- Configuration.@GovLength skipped
- Configuration.@Profile skipped
- Configuration.Encoding := *options1.Encoding*
- Configuration.Resolution := *options1.ResolutionsAvailable[0]*
- Configuration.RateControl skipped
- Configuration.Multicast := *videoEncoderConfToAdd1.Multicast*
- Configuration.Quality := *options1.QualityRange.Min*

17. DUT responds with **SetVideoEncoderConfigurationResponse** message.

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

- ProfileToken := *clearProfileToken1*
- Name skipped
- Configuration[0].Type := VideoEncoder
- Configuration[0].Token := *videoEncoderConfToAdd.@token*

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

20. Add Media Profile with @token = *clearProfileToken1* to *configuredProfilesListForVSC* list.

21. Add Media Profile with @token = *clearProfileToken1* to *configuredProfilesList* list.

22. Go to step 2.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **CreateProfileResponse** message.

A.53 Concurrent Video Streaming over RTP-Unicast/UDP

Name: HelperStreamingRTPUnicastUDPInstances

Procedure Purpose: Helper procedure to verify concurrent video streaming over RTP-Unicast/UDP for provided list of media profiles.

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

Input: List of media profiles (*configuredProfilesList*). Expected media stream encoding (*encoding*).

Returns: None

Procedure:

1. For each Media Profile *profile1* from *configuredProfilesList* repeat the following steps:
 - 1.1. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters
 - in *RtspUnicast* - Transport protocol
 - in *profile1.@token* - Media profile token
 - out *streamUri1* - Stream URI
 - 1.2. ONVIF Client invokes **RTSP DESCRIBE** request to *streamUri1* address.
 - 1.3. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
 - 1.4. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information

- in *streamUri* - Uri for media streaming
- 1.5. 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*
 - 1.6. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 - 1.7. ONVIF Client invokes **RTSP PLAY** request to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 - 1.8. The DUT responds with **200 OK** message with parameters
 - Session
 - RTP-Info
 - 1.9. If DUT does not send *profile1.Configurations.VideoEncoder.Encoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
 - 1.10. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
2. For each media stream *mediaStream* which was invoked at step 1 repeat the following steps:
 - 2.1. ONVIF Client invokes **RTSP TEARDOWN** request to uri address, which corresponds to aggregate control of *mediaStream*, with parameters
 - Session := session which corresponds to *mediaStream*
 - 2.2. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

A.54 Concurrent Video Streaming over RTP-Unicast/UDP

Name: HelperNumberOfInstancesForEncoding

Procedure Purpose: Helper procedure to calculate number of instances for specified encoding.

Pre-requisite: None.

Input: Media stream encoding (*encoding*). Information about number of instances for Video Source Configuration (*info*).

Returns: Number of instances specified encoding (*encodingInstances*)

Procedure:

1. If *info* contains Codec.Encoding = *encoding*:
 - 1.1. Set *encodingInstances* := *info*.Codec[Encoding = *encoding*].Number.
 - 1.2. Skip other steps in procedure.
2. Set *encodingInstances* := *info*.Total.

A.55 Create New Media Profiles to Get Guaranteed Number of Media Profiles for Video Source Configuration for Specified Encoding

Name: HelperFindGuaranteedProfiles1

Procedure Purpose: Helper procedure, which tries to create new Media Profiles to reach number of guaranteed encoder instances without changing already configured profiles for specified encoding.

Pre-requisite: Media2 Service is supported by the DUT. Profile T is supported by the DUT.

Input: Information about guaranteed encoder instances (*info*). Video Source Configuration (*videoSourceConfig*). List of configured media profiles (*configuredProfilesList*). Encoding (*encoding*).

Returns: List of configured Media Profiles (*configuredProfilesList*).

Procedure:

1. Set *configuredProfilesListForVSC1* := empty.
2. If number of items in *configuredProfilesListForVSC1* is equal to limitation for H.264 encoding from *info* which calculates by following the procedure mentioned in [Annex A.54](#), skip other steps of procedure.
3. ONVIF Client invokes **CreateProfile** request with parameters
 - Name := "testMedia"
 - Configuration list - skipped
4. DUT responds with **CreateProfileResponse** message with parameters
 - Token =: *clearProfileToken1*
5. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *clearProfileToken1*
6. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfigurationList1*
7. If *videoSourceConfigurationList1* does not contain item with @token = *videoSourceConfig.@token*, FAIL the test and skip other steps.
8. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *clearProfileToken1*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfig.@token*
9. The DUT responds with **AddConfigurationResponse** message.
10. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *clearProfileToken1*

11. The DUT responds with compatible video encoder configurations in **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList1*
12. If *videoEncoderConfList1* is empty, FAIL the test and skip other steps.
13. If *videoEncoderConfList1* contains only items that were used in Media Profiles from *configuredProfilesList* list, FAIL the test and skip other steps.
14. For each Video Encoder Configuration *videoEncoderConf1* from *videoEncoderConfList1*, which was not used in Media Profiles from *configuredProfilesList* list repeat the following steps:
 - 14.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
 - ConfigurationToken := *videoEncoderConf1.@token*
 - ProfileToken := *clearProfileToken1*
 - 14.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters
 - Options list =: *optionsList1*
 - 14.3. For each Video Encoder Options *vecOptions1* in *optionsList1* repeat the following steps:
 - 14.3.1. If *vecOptions1.Encoding* = *encoding*:
 - 14.3.1.1. Set *options1* := *vecOptions1*
 - 14.3.1.2. Set *videoEncoderConfToAdd1* := *videoEncoderConf1*
 - 14.3.1.3. Go to step 16.
 - 14.4. Go to the next Video Encoder Configuration for the step 14.
15. FAIL the test and skip other steps.
16. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
 - Configuration.@token := *videoEncoderConfToAdd1.@token*
 - Configuration.Name := *videoEncoderConfToAdd1.Name*
 - Configuration.@GovLength skipped

- Configuration.@Profile skipped
- Configuration.Encoding := *options1.Encoding*
- Configuration.Resolution := *options1.ResolutionsAvailable[0]*
- Configuration.RateControl skipped
- Configuration.Multicast := *videoEncoderConfToAdd1.Multicast*
- Configuration.Quality := *options1.QualityRange.Min*

17. DUT responds with **SetVideoEncoderConfigurationResponse** message.

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

- ProfileToken := *clearProfileToken1*
- Name skipped
- Configuration[0].Type := VideoEncoder
- Configuration[0].Token := *videoEncoderConfToAdd.@token*

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

20. Add Media Profile with @token = *clearProfileToken1* to *configuredProfilesListForVSC* list.

21. Add Media Profile with @token = *clearProfileToken1* to *configuredProfilesList* list.

22. Go to step 2.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **CreateProfileResponse** message.

A.56 Determine RSA key length

Name: HelperDetermineRSAKeyLength

Procedure Purpose: Helper procedure to determine the RSA key length to use during testing.

Pre-requisite: Security Configuration Service is received from the DUT. On-board RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability.

Input: None

Returns: The smallest supported RSA key length (*keyLength*).

Procedure:

1. ONVIF Client gets the service capabilities (out *cap*) by the following the procedure mentioned in [Annex A.25](#).
2. ONVIF Client loops through the supported Key length list (*cap.RSAKeyLengths*) and selects the smallest supported key length (*keyLength*).

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- No supported key length was found at step 2.

A.57 Get WebSocket URI

Name: HelperGetWebSocketURI

Procedure Purpose: Helper procedure to get WebSocket URI.

Pre-requisite: WebSocket is supported by the DUT.

Input: None.

Returns: WebSocket URI *uri*.

Procedure:

1. ONVIF Client retrieves Media2 Service capabilities by following the procedure mentioned in [Annex A.25](#) with the following input and output parameters
 - out *cap* - Media2 Service capabilities

2. Set *uri* := *cap*.StreamingCapabilities.RTSPWebSocketUri
3. If hierarchical component (hier_part in [rfc2396]) of *uri* is absolute path construction (abs_path in [rfc2396]):
 - 3.1.ONVIF Client configures WebSocket URI (*uri*) with host and port based on *uri*, URI of the DUT, and HTTP/HTTPS port of the DUT.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- None.

A.58 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTP/TCP

Name: HelperBackchannelStreamingRTPUnicastRTSPHTTPTCP

Procedure Purpose: Helper procedure to verify audio backchannel streaming over RTP-Unicast/RTSP/HTTP/TCP when streaming is sent by POST connection.

Pre-requisite: Audio Backchannel is supported by DUT.

Input: Uri for media streaming (*streamUri*). Expected audio 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 with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address converted to rtsp address on *connection2*.
4. The DUT responds with **200 OK** message with parameters on *connection1*
 - Response header =: *responseHeader*
 - SDP information =: *sdp*

5. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag 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*
8. The DUT responds with **200 OK** message on *connection1* with parameters
 - Transport
 - Session := *session*
9. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
10. The DUT responds with **200 OK** message on *connection1* with parameters
 - Session
 - RTP-Info
11. ONVIF Client sends audio stream with *encoding* to DUT over *connection2*.
12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
13. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
14. ONVIF Client closes *connection2*.

15. 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: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmpmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.

A.59 Audio Backchannel by POST over RTP-Unicast/RTSP/HTTPS/TCP

Name: HelperBackchannelStreamingRTPUnicastRTSPHTTPSTCP

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

Pre-requisite: Audio Backchannel is supported by DUT. HTTPS is configured on the DUT.

Input: Uri for media streaming (*streamUri*). Expected audio stream encoding (*encoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **HTTP GET** request to *streamUri* address to establish DUT to ONVIF Client secured connection for RTP data transfer (*connection1*).
2. ONVIF Client invokes **HTTP 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 with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address converted to rtsp address on *connection2*.
4. The DUT responds with **200 OK** message with parameters on *connection1*

- Response header =: *responseHeader*
 - SDP information =: *sdp*
5. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *encoding*, FAIL the test and skip other steps.
 6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE
 - in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
 7. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag 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*
 8. The DUT responds with **200 OK** message on *connection1* with parameters
 - Transport
 - Session =: *session*
 9. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters
 - Session := *session*
 10. The DUT responds with **200 OK** message on *connection1* with parameters
 - Session
 - RTP-Info
 11. ONVIF Client sends audio stream with *encoding* to DUT over *connection2*.
 12. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 13. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details) on *connection2*, with parameters

- Session := *session*

14. ONVIF Client closes *connection2*.

15. 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: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *encoding* = MP4A-LATM, then *rtmpmap* value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.

A.60 Check of IP address type in response to RTSP DESCRIBE

Name: HelperIPAddressTypeInRTSP

Procedure Purpose: Helper procedure to check IP addresses types returned by DUT in response to RTSP DESCRIBE.

Pre-requisite: None.

Input: Header of response to DESCRIBE (*responseHeader*). SDP (*sdp*). Stream Uri (*streamUri*).

Returns: None.

Procedure:

1. Set *ipType* := *streamUri* IP type.
2. For each **Content-Base** field in *responseHeader* (*contentBase*) that has absolute IP value:
 - 2.1. If *contentBase* IP value does not correspond to *ipType*, FAIL the test and skip other steps (see [RFC2326] for details).

3. For each **Content-Location** field in *responseHeader* (*contentLocation*) that has absolute IP value:
 - 3.1. If *contentLocation* IP value does not correspond to *ipType*, FAIL the test and skip other steps (see [RFC2326] for details).
4. For each **"a=control"** attribute in *sdp* (*aControl*) that has absolute IP value:
 - 4.1. If *aControl* IP value does not correspond to *ipType*, FAIL the test and skip other steps (see [RFC2326] for details).
5. If *ipType* = IPv4:
 - 5.1. If *sdp* contains at least one origin field ("o=") with **addrtype** != "IP4", FAIL the test and skip other steps (see [RFC4566] for details).
 - 5.2. If *sdp* contains at least one origin field ("o=") with IP type of **unicast-address** sub-field != IPv4 type, FAIL the test and skip other steps (see [RFC4566] for details).
 - 5.3. If *sdp* contains at least one connection data field ("c=") with **addrtype** != "IP4", FAIL the test and skip other steps (see [RFC4566] for details).
 - 5.4. If *sdp* contains at least one connection data field ("c=") with IP type of **connection address** sub-field != IPv4 type, FAIL the test and skip other steps (see [RFC4566] for details).
6. If *ipType* = IPv6:
 - 6.1. If *sdp* contains at least one origin field ("o=") with **addrtype** != "IP6", FAIL the test and skip other steps (see [RFC4566] for details).
 - 6.2. If *sdp* contains at least one origin field ("o=") with IP type of **unicast-address** sub-field != IPv6 type, FAIL the test and skip other steps (see [RFC4566] for details).
 - 6.3. If *sdp* contains at least one connection data field ("c=") with **addrtype** != "IP6", FAIL the test and skip other steps (see [RFC4566] for details).
 - 6.4. If *sdp* contains at least one connection data field ("c=") with IP type of **connection address** sub-field != IPv6 type, FAIL the test and skip other steps (see [RFC4566] for details).

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

- None.

A.61 Set Video Encoder Configuration for Streaming

Name: HelperSetVEC

Procedure Purpose: Helper procedure to configure video encoder configuration for streaming.

Pre-requisite: Media2 is supported by the DUT.

Input: Media profile with video encoder configuration (*profile*), video encoder configuration options (*vecOptions*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: None.

Procedure:

1. 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 (or skipped if *vecOptions*.@GovLengthRange skipped)
 - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline (or skipped if *vecOptions*.@ProfilesSupported skipped)
 - Configuration.Encoding := *vecOptions*.Encoding
 - Configuration.Resolution := resolution closest to 640x480 from *vecOptions*.ResolutionsAvailable list
 - if *vecOptions*.@FrameRatesSupported skipped and *profile*.Configurations.VideoEncoder.RateControl skipped:
 - Configuration.RateControl skipped
 - if *vecOptions*.@FrameRatesSupported or *profile*.Configurations.VideoEncoder.RateControl is not skipped:

- Configuration.RateControl.@ConstantBitRate :=
vecOptions.@ConstantBitRateSupported
 - Configuration.RateControl.FrameRateLimit := value closest to 25
but greater than 1 from vecOptions.@FrameRatesSupported
list (or profile.Configurations.VideoEncoder.RateControl.FrameRateLimit if
vecOptions.@FrameRatesSupported skipped)
 - Configuration.RateControl.BitrateLimit := min {max
{profile.Configurations.VideoEncoder.RateControl.BitrateLimit,
vecOptions.BitrateRange.Min}, vecOptions.BitrateRange.Max}
 - if protocol is not equal to RtspMulticast:
 - Configuration.Multicast := profile.Configurations.VideoEncoder.Multicast
 - if protocol = RtspMulticast and ipVersion = IPv4:
 - 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
 - if protocol = RtspMulticast and ipVersion = IPv6:
 - 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
2. The DUT responds with **SetVideoEncoderConfigurationResponse** message.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.62 Set Audio Encoder Configuration for Streaming

Name: HelperSetAEC

Procedure Purpose: Helper procedure to configure audio encoder configuration for streaming.

Pre-requisite: Media2 is supported by the DUT. Media2 Audio is supported by the DUT.

Input: Media profile with audio encoder configuration (*profile*), audio encoder configuration options (*aecOptions*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: None.

Procedure:

1. 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 := *aecOptions*.Encoding
 - if *protocol* is not equal to RtspMulticast:
 - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
 - if *protocol* = RtspMulticast and *ipVersion* = IPv4:
 - 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
 - if *protocol* = RtspMulticast and *ipVersion* = IPv6:
 - 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.Bitrates := the nearest value to *profile*.Configurations.AudioEncoder.Bitrates from *aecOptionsBitrateList.Items* list
 - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptionsSampleRateList.Items* list
2. The DUT responds with **SetAudioEncoderConfigurationResponse** message.

Procedure Result:**PASS –**

- DUT passes all assertions.

FAIL –

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

A.63 Device Configuration for Audio Backchannel and Video and Audio Streaming

Name: HelperDeviceConfigurationForBackchannelAndVideoAndAudioStreaming

Procedure Purpose: Helper procedure to configure Media profile, Video Encoder Configuration, Audio Encoder Configuration, Audio Decoder Configuration, and get stream URI from the DUT for audio backchannel and video and audio streaming.

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

Input: Required audio decoder (*requiredAudioDecoder*), Transport protocol (*protocol*), IP version (*ipVersion*).

Returns: Stream Uri (*streamUri*). Media profile with required configurations (*profile*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
 - Token skipped
 - Type[0] := All
2. The DUT responds with **GetProfilesResponse** message with parameters
 - Profiles list =: *profileList*
3. Set *profile* = *profileList*[0].
4. Set *confTypeList* := (configurations that are contained in profile *profile*)
5. ONVIF Client removes all configurations from the Media Profile by following the procedure mentioned in [Annex A.43](#) with the following input and output parameters
 - in *confTypeList* - list of configuration type to remove from Media Profile
 - in *profile* - Media Profile to update
6. ONVIF Client invokes **GetAudioOutputConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile*.@token
7. The DUT responds with **GetAudioOutputConfigurationsResponse** with parameters
 - Configurations list =: *audioOutputConfList*
8. For each Audio Output Configuration *audioOutputConfiguration* in *audioOutputConfList* repeat the following steps:
 - 8.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile*.@token
 - Name skipped
 - Configuration[0].Type := AudioOutput
 - Configuration[0].Token := *audioOutputConfiguration*.@token

- 8.2. The DUT responds with **AddConfigurationResponse** message.
- 8.3. ONVIF Client invokes **GetAudioDecoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
- 8.4. The DUT responds with **GetAudioDecoderConfigurationsResponse** with parameters
 - Configurations list =: *audioDecoderConfList*
- 8.5. For each Audio Decoder Configuration *audioDecoderConfiguration* in *audioDecoderConfList* repeat the following steps:
 - 8.5.1. ONVIF Client invokes **GetAudioDecoderConfigurationOptions** request with parameters
 - ConfigurationToken := *audioDecoderConfiguration.@token*
 - ProfileToken := *profile.@token*
 - 8.5.2. DUT responds with **GetAudioDecoderConfigurationOptionsResponse** message with parameters
 - Options list =: *adcOptionsList*
 - 8.5.3. If *requiredAudioDecoder* = AAC:
 - 8.5.3.1. If *adcOptionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":
 - 8.5.3.1.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := AudioDecoder
 - Configuration[0].Token := *audioDecoderConfiguration.@token*

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

8.5.3.1.3. Set *requiredAudioDecoder* := "MP4A-LATM" if *adcOptionsList* contains item with Encoding = "MP4A-LATM", otherwise "MPEG4-GENERIC".

8.5.4. If *requiredAudioDecoder* = !AAC:

8.5.4.1. If *adcOptionsList* list contains item with Encoding = *requiredAudioDecoding*:

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

- ProfileToken := *profile.@token*
- Name skipped
- Configuration[0].Type := AudioDecoder
- Configuration[0].Token := *audioDecoderConfiguration.@token*

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

8.5.5. If Audio Decoder was added at step [8.5.3](#) or at step [8.5.4](#):

8.5.5.1. ONVIF Client tries to add VideoSource Configuration, VideoEncoder Configuration, AudioSource Configuration and AudioEncoder Configuration to the Media Profile by following the procedure mentioned in [Annex A.64](#) with the following input and output parameters

- in *profile* - Media profile
- out (optional) *vecOptions* - Video Encoder Configuration Options for the Media Profile
- out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile

8.5.5.2. If *profile* was configured with Video and Audio configurations by step [8.5.5.1](#), go to step [10](#).

8.5.5.3. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile.@token*
- Configuration[0].Type = AudioDecoder
- Configuration[0].Token skipped

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

9. If *profile* was not configured with AudioOutput, AudioDecoder, VideoSource, VideoEncoder, AudioSource, and AudioEncoder at step 4.3, FAIL the test, restore DUT settings, and skip other steps.

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

- Configuration.@token := *profile.Configurations.AudioOutput.@token*
- Configuration.Name := *profile.Configurations.AudioOutput.Name*
- Configuration.UseCount := *profile.Configurations.AudioOutput.UseCount*
- Configuration.OutputToken := *profile.Configurations.AudioOutput.OutputToken*
- Configuration.SendPrimacy skipped
- Configuration.OutputLevel := *profile.Configurations.AudioOutput.OutputLevel*

11. DUT responds with **SetAudioOutputConfigurationResponse** message.

12. ONVIF Client retrieves a stream uri for Media Profile for required transport protocol by following the procedure mentioned in [Annex A.38](#) with the following input and output parameters

- in *protocol* - Transport protocol
- in *ipVersion* - IP Type
- in *profile.@token* - Media profile token
- out *uri* - Stream URI

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetAudioOutputConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetAudioDecoderConfigurationsResponse** message.
- DUT did not send **SetAudioOutputConfigurationResponse** message.
- DUT did not send **RemoveConfigurationResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **GetAudioSourceConfigurationsResponse** message.
- DUT did not send **GetAudioEncoderConfigurationsResponse** message.
- DUT did not send **SetVideoEncoderConfigurationResponse** message.
- DUT did not send **SetAudioEncoderConfigurationResponse** message.

Note: See [Annex A.3](#) for Name and Token Parameters Length limitations.

A.64 Media2 Service – Adding VideoSource, VideoEncoder, AudioSource and AudioEncoder configurations to Media Profile

Name: HelperAddVideoAndAudioConfigurationsToMediaProfile

Procedure Purpose: Helper procedure to add VideoSource Configuration, VideoEncoder Configuration, AudioSource Configuration and AudioEncoder Configuration to the Media Profile.

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

Input: Media Profile (*profile*)

Returns: Video Encoder Configuration Options for the Media Profile (*vecOptions*) (optional, returned in case profile was configured with video and audio). Audio Encoder Configuration Options for the Media Profile (*aecOptions*) (optional, returned in case profile was configured with video and audio).

Procedure:

1. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
2. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
 - Configurations list =: *videoSourceConfList*
3. For each Video Source Configuration *videoSourceConfiguration* in *videoSourceConfList* repeat the following steps:
 - 3.1. ONVIF Client invokes **AddConfiguration** request with parameters
 - ProfileToken := *profile.@token*
 - Name skipped
 - Configuration[0].Type := VideoSource
 - Configuration[0].Token := *videoSourceConfiguration.@token*
 - 3.2. The DUT responds with **AddConfigurationResponse** message.
 - 3.3. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters
 - ConfigurationToken skipped
 - ProfileToken := *profile.@token*
 - 3.4. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters
 - Configurations list =: *videoEncoderConfList*
 - 3.5. For each Video Encoder Configuration *videoEncoderConfiguration* in *videoEncoderConfList* repeat the following steps:
 - 3.5.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters
 - ConfigurationToken := *videoEncoderConfiguration.@token*
 - ProfileToken := *profile.@token*
 - 3.5.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

3.5.3. If *optionsList* list contains item with Encoding = H264 or H265:

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

- ProfileToken := *profile.@token*
- Name skipped
- Configuration[0].Type := VideoEncoder
- Configuration[0].Token := *videoEncoderConfiguration.@token*

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

3.5.3.3. Set *vecOptions* := item with Encoding = H264 from *optionsList* list if present, otherwise item with Encoding = H265.

3.5.3.4. ONVIF Client sets video encoder configuration by following the procedure mentioned in [Annex A.61](#) with the following input and output parameters

- in *profile* - Media profile
- in *vecOptions* - video encoder configuration options
- in *protocol* - Transport protocol
- in *ipVersion* - IP Type

3.5.3.5. ONVIF Client tries to add AudioSource Configuration and AudioEncoder Configuration to the Media Profile by following the procedure mentioned in [Annex A.65](#) with the following input and output parameters

- in *profile* - Media profile
- out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile

3.5.3.6. If *aecOptions* != NULL

3.5.3.6.1. ONVIF Client sets audio encoder configuration by following the procedure mentioned in [Annex A.62](#) with the following input and output parameters

- in *profile* - Media profile
- in *aecOptions* - audio encoder configuration options
- in *protocol* - Transport protocol
- in *ipVersion* - IP Type

3.5.3.6.2. Skip other steps.

3.5.3.7. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile.@token*
- Configuration[0].Type = VideoEncoder
- Configuration[0].Token skipped

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

3.6. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile.@token*
- Configuration[0].Type = VideoSource
- Configuration[0].Token skipped

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.65 Media2 Service – Adding AudioSource and AudioEncoder to Media Profile

Name: HelperAddAudioConfigurationsToMediaProfile

Procedure Purpose: Helper procedure to add AudioSource Configuration and AudioEncoder Configuration to the Media Profile.

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

Input: Media Profile (*profile*)

Returns: Video Encoder Configuration Options for the Media Profile (*vecOptions*) (optional, returned in case profile was configured with audio). Audio Encoder Configuration Options for the Media Profile (*aecOptions*) (optional, returned in case profile was configured with audio).

Procedure:

1. ONVIF Client tries to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding support to the Media Profile by following the procedure mentioned in [Annex A.46](#) with the following input and output parameters
 - in PCMU - required audio encoding
 - in *profile* - Media profile
 - out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile
2. If *aecOptions* != NULL, skip other steps in procedure..
3. ONVIF Client tries to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding support to the Media Profile by following the procedure mentioned in [Annex A.46](#) with the following input and output parameters
 - in AAC - required audio encoding
 - in *profile* - Media profile
 - out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.

A.66 Audio Backchannel and Media Streaming over RTP-Unicast/UDP

Name: HelperBackchannelAndMediaStreamingRTPUnicastUDP

Procedure Purpose: Helper procedure to verify audio backchannel, video, and audio streaming over RTP-Unicast/UDP.

Pre-requisite: Audio Backchannel is supported by DUT. Audio is supported by DUT. Video is supported by DUT. Real-time streaming is supported by DUT.

Input: Uri for media streaming (*streamUri*). Audio backchannel stream encoding (*audioBackchannelEncoding*). Expected video stream encoding (*videoEncoding*). Expected audio stream encoding (*audioEncoding*).

Returns: None

Procedure:

1. ONVIF Client invokes **RTSP DESCRIBE** request with "**Require: www.onvif.org/ver20/backchannel**" tag to *streamUri* address.
2. The DUT responds with **200 OK** message with parameters
 - Response header =: *responseHeader*
 - SDP information =: *sdp*
3. If *sdp* does not contain Media Type = audio and with a=sendonly and with rtpmap value corresponding to *audioBackchannelEncoding*, FAIL the test and skip other steps.
4. If *sdp* does not contain Media Type = video with rtpmap value corresponding to *videoEncoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
5. If *sdp* does not contain Media Type = audio with rtpmap value corresponding to *audioEncoding* and without session attribute "sendonly" (a=sendonly), FAIL the test and skip other steps.
6. ONVIF Client checks types of IP addresses returned in response to DESCRIBE by following the procedure mentioned in [Annex A.60](#) with the following input parameters
 - in *responseHeader* - header of response to DESCRIBE

- in *sdp* - SDP information
 - in *streamUri* - Uri for media streaming
7. ONVIF Client invokes **RTSP SETUP** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to audio backchannel media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port1-port2*
 8. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 9. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to video media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port3-port4*
 10. The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 11. ONVIF Client invokes **RTSP SETUP** request to uri address, which corresponds to audio media type (see [RFC2326] for details), with parameters
 - Transport := RTP/AVP;unicast;client_port=*port5-port6*
 - The DUT responds with **200 OK** message with parameters
 - Transport
 - Session =: *session*
 12. ONVIF Client invokes **RTSP PLAY** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 13. The DUT responds with **200 OK** message with parameters
 - Session

- RTP-Info
14. ONVIF Client sends RTP Unicast audio stream with *audioBackchannelEncoding* to DUT over UDP.
 15. If DUT does not send *videoEncoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
 16. If DUT does not send *audioEncoding* RTP media stream to ONVIF Client over UDP, FAIL the test and skip other steps.
 17. If DUT does not send valid RTCP packets, FAIL the test and skip other steps.
 18. ONVIF Client invokes **RTSP TEARDOWN** request with "**Require: www.onvif.org/ver20/backchannel**" tag to uri address, which corresponds to aggregate control (see [RFC2326] for details), with parameters
 - Session := *session*
 19. 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.

Note: ONVIF Client checks authentication type for all RTSP requests by following the procedure mentioned in [Annex A.47](#).

Note: If *audioBackchannelEncoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 3.

Note: If *audioEncoding* = MP4A-LATM, then rtpmap value may be equal either MP4A-LATM or MPEG4-GENERIC at step 5.