ONVIF®

Profile T Client Test Specification

Version 22.06

June 2022

REVISION HISTORY

Vers.	Date	Description
22.06	Dec 27, 2022	Profile Normative Reference were removed from test cases according to #364
21.12	Oct 12, 2021	The following was done according to #425: Check Condition based on Device Features of Discovery feature was changed from 'All' to 'Discovery'
21.06	Jun 03, 2021	The following was updated according to #325: SETSYNCHRONIZATIONPOINT-1 test name was changed from SET SYNCHRONIZATION POINT to SET SYNCHRONIZATION POINT (EVENT SERVICE). Set Synchronization Point feature was renamed to Set Synchronization Point (Event Service)
21.06	Jun 21, 2021	The following was updated according to #325: Auxiliary Commands feature was rename to Auxiliary Commands (Device Management Service)
21.06	Jan 21, 2021	In the scope of #364 format of the following features were updated to show dependent test cases inside feature: Media Streaming Using Media2 Motion Alarm OSD Configuration Using Media2 Media Profile Management PTZ Media2 Profile Configuration Get Digital Inputs Test Motion Detection
21.06	Jan 21, 2021	The following was done according to #364: PTZ Using Media2 Zoom Continuous Positioning feature updated: requirement about PtzContinuousPositioning feature was removed to be compatible with implementation.
21.06	Jan 21, 2021	The following was done according to #364: PTZ Using Media2 Pan Tilt Continuous Positioning feature updated: requirement about PtzContinuousPositioning feature was removed to be compatible with implementation.
20.12	Dec 8, 2020	DEVICEDISCOVERYTYPEFILTER-1 DEVICE DISCOVERY TYPE FILTER was updated according to #406: Types value check was updated to accept QName list instead of one QName value.
20.12	Oct 27, 2020	The following was done according to #394:

		Check Condition based on Device Features of Network Configuration feature was changed from 'All' to 'Network Configuration'
20.12	Oct 27, 2020	The following was done according to #393: Check Condition based on Device Features of User Handling feature was changed from 'All' to 'User Configuration'
20.12	Aug 31, 2020	Motion Detection Feature: Check Condition based on Device Features was changed according to #325.
20.12	Aug 31, 2020	Set Synchronization Point Feature: Check Condition based on Device Features was changed according to #325.
20.12	Aug 31, 2020	Unsubscribe Feature: Check Condition based on Device Features was changed according to #325.
20.12	Aug 31, 2020	Keep Alive for Pull Point Event Handling Feature: Check Condition based on Device Features was changed according to #325.
20.06	Jan 20, 2020	The following was done according to #326: Pre-Requisite of Video Source Configuration Using Media2 feature and their test cases changed (from 'Media2_Video' to 'Media2_Video OR Media2_VideoSource')
19.12	Oct 29, 2019	The following was done according to #325: Metadata Streaming Using Media2 conditional feature was added
19.12	Oct 22, 2019	The following was done according to #325: Check Condition based on Device Features of HTTP Digest Authentication for RTSP feauture was changed from 'Profile T' to 'Profile T or Profile M'.
19.12	Aug 13, 2019	The following was done according to #325: EVENTHANDLING-3 METADATA STREAMING test was removed from Event Handling Feature and moved to Metadata Streaming Using Media2. Test case ID was changed to MEDIA2_METADATASTREAMING-1. Event Handling will use link to this test. EVENTHANDLING-4 METADATA STREAMING USING MEDIA was added for Profile S Devices.
19.12	Sep 6, 2019	DEVICEDISCOVERYTYPEFILTER-1 DEVICE DISCOVERY TYPE FILTER was updated according to #323: Unnecessary step with check that ProbeMatch is sent to Client IP address was removed.
19.12	Aug 13, 2019	The following was done according to #341: Motion Detection section and Motion Detection Cases section was moved from ONVIF Analytics Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: HTTP Digest section and HTTP Digest Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341:

		Capabilities section and Capabilities Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Get Services section and Get Services Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Event Handling section and Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Set Synchronization Point section and Set Synchronization Point Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Discovery section and Discovery Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Device Discovery Type Filter section and Device Discovery Type Filter Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Network Configuration section and Network Configuration Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: HTTP Digest Authentication for RTSP section and HTTP Digest Authentication for RTSP Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Get Imaging Settings section and Get Imaging Settings Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Imaging Settings Configuration section and Imaging Settings Configuration Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Motion Alarm section and Motion Alarm Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341:

		PTZ Pan Tilt Continuous Positioning section and PTZ Pan Tilt Continuous Positioning Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Zoom Continuous Positioning section and PTZ Zoom Continuous Positioning Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees section and PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space section and PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Using Media2 Absolute Positioning - Zoom Position Generic Space section and PTZ Using Media2 Absolute Positioning - Zoom Position Generic Space section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Using Media2 Pan Tilt Continuous Positioning section and PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Using Media2 Zoom Continuous Positioning section and PTZ Using Media2 Zoom Continuous Positioning Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: System section and System Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: User Handling section and User Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: NTP section and NTP Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.

19.12	Aug 13, 2019	The following was done according to #341: Auxiliary Commands section and Auxiliary Commands Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Motion Alarm section and Motion Alarm Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Focus Control section and Focus Control Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Tampering section and Tampering Test Cases section was moved from ONVIF Imaging Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ - Listing section and PTZ - Listing Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Presets section and PTZ Presets Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Home Position section and PTZ Home Position Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ - Set Preset section and PTZ - Set Preset Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Get Compatible Configurations section and PTZ Get Compatible Configurations Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: PTZ Media2 Profile Configuration section and PTZ Media2 Profile Configuration Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341:

		PTZ Set Configuration section and PTZ Set Configuration Test Cases section was moved from ONVIF PTZ Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Relay Outputs Using Device IO section and Relay Outputs Using Device IO Test Cases section was moved from ONVIF Device IO Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Get Digital Inputs section and Get Digital Inputs Test Cases section was moved from ONVIF Device IO Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Get Supported Rules section and Get Supported Rules Test Cases section was moved from ONVIF Analytics Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Get Rules section and Get Rules Test Cases section was moved from ONVIF Analytics Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Unsubscribe section and Unsubscribe Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.12	Aug 13, 2019	The following was done according to #341: Keep Alive for Pull Point Event Handling section and Keep Alive for Pull Point Event Handling Test Cases section was moved from ONVIF Core Client Test Specification to ONVIF Profile T Client Test Specifications.
19.06	Jun 14, 2019	The following was done according to #309: 'Validated Feature' section for each feature updated to be synchronized with feature ID used in feature list. 'Feature Under Test' section for each test case updated to be synchronized with sub-feature ID used in feature list. 'Validated Feature List' test case section removed.
19.06	Mar 28, 2019	The following was updated in the scope of #319: MEDIA2_AUDIOSTREAMING_AAC-1 AAC AUDIO STREAMING USING MEDIA2 (MP4A-LATM encoding name added) MEDIA2_AUDIOBACKCHANNELSTREAMING-2 AAC AUDIO BACKCHANNEL STREAMING USING MEDIA2 (MP4A-LATM encoding name added)
18.06	Jun 21, 2018	Reformatting document using new template
18.06	Jun 15, 2018	Audio Backchannel Streaming Using Media2 updated to exclude multicast streaming according to #274

18.06	May 22, 2018	HTTPS Streaming Using Media2 feature was added according to #224
18.06	May 15, 2018	Audio Output Configuration Using Media2 feature requirement level was changed from Conditional to Optional according to #261
18.06	Apr 28, 2018	Get Snapshot Uri Using Media2 feature requirement level was changed from Conditional to Optional according to #253
18.06	Apr 27, 2018	Video Streaming Using Media2 feature was replaced with H264 Video Streaming Using Media2 and H265 Video Streaming Using Media2 features according to #242 Audio Streaming Using Media2 feature was replaced with G.711 Audio Streaming Using Media2 and AAC Audio Streaming Using Media2 features according to #243
18.06	Apr 05, 2018	'Required Number of Devices Summary' Annex added according to #241
18.06	Feb 16, 2018	The following were updated in the scope of #241: Feature Level Requirement (updated with new rules) Each Feature Level Requirement (updated with Check Condition based on Device Features and Required Number of Devices)
17.12	Sept 1, 2017	The following test cases added according to #226: • Media Profile Management Test Cases
17.12	Aug 15, 2017	Requirement level of Profile T of the following features was changed to Optional according to #220: Audio Source Configuration Using Media2 Audio Decoder Configuration Using Media2
17.12	Aug 15, 2017	Requirement level of Profile T of the following features was changed from Mandatory to Conditional according to #220: Audio Streaming Using Media2 Audio Profile Configuration Using Media2 Audio Backchannel Streaming Using Media2 Audio Output Profile Configuration Using Media2
17.12	Aug 14, 2017	The following test cases added according to #222: • Multicast Streaming Test Cases • Video Source Configuration Using Media2 Test Cases • Metadata Configuration Using Media2 Test Cases • Metadata Profile Configuration Using Media2 Test Cases • Video Profile Configuration Using Media2 Test Cases
17.06	Jun 15, 2017	Links in Normative references section were updated.
17.06	Jun 13, 2017	The following test cases added according to #201: Get Snapshot Uri Using Media2.
17.06	Jun 9, 2017	The following test cases added according to #201: Analytics Profile Configuration Using Media2 Test Cases

		Video Source Mode Test Cases Audio Source Configuration Using Media2 Test Cases Audio Output Configuration Using Media2 Test Cases Audio Decoder Configuration Using Media2 Test Cases List Video Source Configurations Using Media2 Test Cases OSD Configuration Using Media2 Test Cases
17.06	Jun 07, 2017	The following Device IO test cases moved into ONVIF Device IO Client Test Specification according to #194: Relay Outputs Using Device IO Get Digital Inputs
17.06	Jun 06, 2017	The following PTZ test cases moved into ONVIF PTZ Client Test Specification according to #194: PTZ Using Media2 Absolute Positioning PTZ Using Media2 Continuous Positioning PTZ - Set Preset Annex A.1 Get default PTZ space of PTZ Configuration corresponding to Move Operation
17.06	May 29, 2017	• Audio Profile Configuration Using Media Test Cases updated according to #191 • Audio Output Profile Configuration Using Media2 Test Cases updated according to #193 • Get Audio Output Configurations Using Media2 Test Cases removed according to #193
17.06	Apr 04, 2017	• PTZ - Set Preset Test Cases added • Audio Profile Configuration Using Media2 Test Cases added
17.06	Apr 03, 2017	• PTZ Using Media2 Continuous Positioning Test Cases added
17.06	Mar 29, 2017	• Audio Output Profile Configuration Using Media2 added
17.06	Mar 28, 2017	• Get Audio Output Configurations Using Media2 Test Cases added
17.06	Mar 27, 2017	• PTZ Using Media2 Absolute Positioning Test Cases added
17.06	Mar 24, 2017	• Digital Inputs Test Cases added • Audio Backchannel Streaming Using Media2 Test Cases added
17.06	Mar 23, 2017	• Relay Outputs Using DeviceIO Test Cases added
17.06	Mar 21, 2017	• Media2 Media Streaming Test Cases added
17.06	Mar 20, 2017	• Media2 Audio Encoder Configuration Test Cases added
17.06	Mar 17, 2017	• Media2 Video Streaming Test Cases added • Media2 Video Encoder Configuration Test Cases added
17.06	Feb 06, 2017	• Initial version: General parts added

Table of Contents

1 Introduction

The goal of the ONVIF Test Specification set is to make it possible to realize fully interoperable IP physical security implementations from different vendors. This specification also acts as an input document to the development of a test tool which will be used to test the ONVIF Client implementation conformance towards ONVIF standard. This Client Test Tool analyzes network communications between ONVIF Devices and Clients being tested and determines whether a specific Client is ONVIF conformant (see ONVIF Conformance Process Specification).

This particular document defines test cases required for testing Profile T features of a Client application e.g. Video Streaming, Video Encoder Configuration, Audio Streaming, Configuration of Audio Profile, Audio Source Configuration, Audio Encoder Configuration, Audio Output Streaming, Configuration of Audio Output Profile, Audio Output Configuration, Metadata Configuration, Relay Outputs, Digital Inputs, and Tampering. It also describes the test framework, test setup, prerequisites, test policies needed for the execution of the described test cases.

1.1 Scope

This ONVIF Profile T Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of Profile T features. Conformance testing is meant to be black-box network traces analysis and verification. The objective of this specification is to provide the test cases to test individual requirements of ONVIF Clients in the scope of Profile T features according to ONVIF Profile Specifications.

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for Profile T features.

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

- 3rd parties Client use cases
- Non-functional (performance and regression) testing and analysis.
- SOAP Implementation Interoperability test i.e. Web Services Interoperability Basic Profile version 2.0 (WS-I BP2.0).
- Network protocol implementation Conformance test for HTTPS and HTTP protocols.

The following sections cover test cases needed for the verification of relevant features as mentioned in the ONVIF Profile Specifications.

1.2 Test Cases for Profile Mandatory Features

This section defines test cases which are mandatory for Profile T Client conformance.

1.2.1 HTTP Digest

HTTP Digest section defines security mechanism for HTTP Digest Authentication.

1.2.2 Capabilities

Capabilities section specifies Client ability to retrieve available services and advanced functionalities which are offered by a Device.

1.2.3 Get Services

Get Services section specifies Client ability to retrieve list of services with using GetServices operation.

1.2.4 Event Handling

Event Handling section defines Client ability to initiate and receive notifications (events) from a Device.

The event handling test cases cover the following mandatory interfaces:

- Pull Point Notification Interface
  - This test specification provides test cases to verify the implementation of the PullPoint Interface of a Client.
- Basic Notification Interface
  - This test specification provides test cases to verify the implementation of the Basic Notification Interface of a Client.
- Metadata Streaming Interface
  - This test specification provides test cases to verify the implementation of the Metadata Streaming Interface of a Client using Media Service and using Media2 Service.

1.2.5 Set Synchronization Point (Event Service)

Set Synchronization Point section defines Client ability to synchronize its properties with the properties of the device using SetSynchronizationPoint operation.

1.2.6 Discovery

Discovery section defines Client ability to locate services on a local network using Web Services Dynamic Discovery (WS-Discovery) protocol. It uses IP multicast address 239.255.255.250 and TCP and UDP port 3702 and SOAP-over-UDP standard for communication between nodes.

1.2.7 Device Discovery Type Filter

Device Discovery Type Filter Test Cases section defines Client ability to locate services, which are support Device Discovery Type on a local network using Web Services Dynamic Discovery (WS-Discovery) protocol. It uses IPv4 address 239.255.255.250 or multicast IPv6 address [FF02::C] and port 3702 with Types filter that contains tds:Device or with skipped Types filter.

1.2.8 Network Configuration

Network Configuration section defines Client ability to obtain and configure of network settings on Device.

1.2.9 HTTP Digest Authentication for RTSP

HTTP Digest Authentication for RTSP section defines security mechanism for Digest Authentication for RTSP.

1.2.10 Get Stream Uri Using Media2

Get Stream Uri Using Media2 section specifies Client ability to retrieve a Media2 stream URI from Device.

1.2.11 Get Profiles Using Media2

Get Profiles Using Media2 section specifies Client ability to retrieve a list of Media2 profiles from Device.

1.2.12 Media Streaming Using Media2

Media Streaming Using Media2 section defines different streaming options based on RTP protocol which are required for all types of streams of video, audio and metadata. Media control is done using RTSP protocol.

1.2.13 H264 Video Streaming Using Media2

H264 Media Streaming Using Media2 section specifies Client ability to establish specific video streams in H264 video format.

1.2.14 H265 Video Streaming Using Media2

H265 Media Streaming Using Media2 section specifies Client ability to establish specific video streams in H265 video format.

1.2.15 Video Encoder Configuration Using Media2

Video Encoder Configuration Using Media2 section specifies modification of video encoder configurations on Device.

1.2.16 Get Imaging Settings

Get Imaging Settings section specifies Client ability to request imaging settings from Device.

1.2.17 Imaging Settings Configuration

Imaging Settings Configuration section specifies Client ability to change imaging settings on Device.

1.2.18 Motion Alarm

Motion Alarm section specifies Client ability to receive notifications of motion alarm event.

1.2.19 PTZ Pan Tilt Continuous Positioning

PTZ Pan Tilt Continuous Move section specifies Client ability to move a PTZ Device using ContinuousMove operation for Pan Tilt and stop ongoing pan tilt movement using Stop operation or sending zero values for Pan/Tilt.

1.2.20 PTZ Zoom Continuous Positioning

PTZ Zoom Continuous Move section specifies Client ability to move a PTZ Device using ContinuousMove operation for Zoom and stop ongoing pan tilt movement using Stop operation or sending zero values for Zoom.

1.2.21 PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees

PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees section specifies Client ability to move a PTZ Device using the AbsoluteMove operation for Media2 profile with Spherical Position Space.

1.2.22 PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space

PTZ Using Media2 Absolute Positioning Test Cases - Pan Tilt Position Generic Space section specifies Client ability to move a PTZ Device using the Pan Tilt AbsoluteMove operation for Media2 profile with Generic Space.

1.2.23 PTZ Using Media2 Absolute Positioning - Zoom Position Generic Space

PTZ Using Media2 Absolute Positioning Test Cases - Zoom Position Generic Space section specifies Client ability to move a PTZ Device using the Zoom AbsoluteMove operation for Media2 profile with Generic Space.

1.2.24 PTZ Using Media2 Pan Tilt Continuous Positioning

PTZ Using Media2 Pan Tilt Continuous Positioning section specifies Client ability to move a PTZ Device using pan tilt ContinuousMove operation for Media2 profile.

1.2.25 PTZ Using Media2 Zoom Continuous Positioning

PTZ Using Media2 Zoom Continuous Positioning section specifies Client ability to move a PTZ Device using zoom ContinuousMove operation for Media2 profile.

1.3 Test Cases for Profile Conditional Features

This section defines test cases which are mandatory for Profile T Client conformance.

1.3.1 System

System section defines Client ability to obtain Device information and configure of system settings on Device.

1.3.2 User Handling

User Handling section defines Client ability to manage users on Device.

1.3.3 NTP

NTP section defines Client ability to configure synchronization of time using NTP servers on Device.

1.3.4 Auxiliary Commands (Device Management Service)

Auxiliary Commands section defines Client ability to manage auxiliary commands supported by the Device.

1.3.5 Metadata Profile Configuration Using Media2

Metadata Profile Configuration Using Media2 section specifies Client ability to add metadata configuration to a media profile.

1.3.6 Metadata Configuration Using Media2

Metadata Configuration Using Media2 section specifies listing and modification of metadata configurations on Device.

1.3.7 Metadata Streaming Using Media2

Metadata Streaming Using Media2 section specifies receiving of metadata stream from Device using Madia2 Service.

1.3.8 Multicast Streaming Using Media2

Multicast Streaming Using Media2 section specifies Client ability to stream multicast to Device.

1.3.9 Video Profile Configuration Using Media2

Video Profile Configuration Using Media2 section specifies Client ability to add video encoder configuration to a media profile.

1.3.10 Audio Encoder Configuration Using Media2

Audio Encoder Configuration Using Media2 section specifies listing and modification of audio encoder configurations on Device.

1.3.11 G711 Audio Streaming Using Media2

G711 Audio Streaming Using Media2 section specifies Client ability to establish specific audio streams in G.711 audio format.

1.3.12 AAC1 Audio Streaming Using Media2

AAC Audio Streaming Using Media2 section specifies Client ability to establish specific audio streams in AAC audio format.

1.3.13 Audio Profile Configuration Using Media2

Audio Profile Configuration Using Media2 section specifies Client ability to configure or create media profile with audio source configuration and to add audio encoder configuration to a media profile.

1.3.14 Audio Backchannel Streaming Using Media2

Audio Backchannel Streaming Using Media2 section specifies Client ability to stream audio for backchannel to Device.

1.3.15 Audio Output Profile Configuration Using Media2

Audio Output Profile Configuration Using Media2 section specifies Client ability to configure or create media profile with audio output configuration and to add audio decoder configuration to a media profile.

1.3.16 Analytics Profile Configuration Using Media2

Analytics Profile Configuration Using Media2 section specifies Client ability to add analytics configuration to a media profile.

1.3.17 Video Source Mode

Video Source Mode section specifies Client ability to request the information for current video source mode and settable video source modes and to change current video source mode on device.

1.3.18 Video Source Configuration Using Media2

Video Source Configuration Using Media2 section specifies listing and modification of video source configurations on Device.

1.3.19 List Video Source Configurations Using Media2

List Video Source Configurations Using Media2 section specifies listing of video source configurations on Device.

1.3.20 OSD Configuration Using Media2

OSD Configuration Using Media2 section specifies listing and modification of OSD configurations on Device.

1.3.21 Media Profile Management

Media Profile Management section specifies Client ability to create media profile with video source configuration or audio source configuration or audio output configuration and to list video source configurations and video encoder instances on Device.

1.3.22 HTTPS Streaming Using Media2

HTTPS Streaming Using Media2 section specifies Client ability to establish specific media streams over RTP/RTSP/HTTPS/TCP.

1.3.23 Focus Move Capabilities

Focus Move Capabilities section specifies Client ability to retrieve focus move capabilities from Device.

1.3.24 Focus Control

Focus Control section specifies Client ability to control focus on Device.

1.3.25 Tampering

Tampering section specifies Client ability to receive notifications of Tampering events.

1.3.26 PTZ - Listing

PTZ - Listing section specifies Client ability to read PTZ capabilities.

1.3.27 PTZ Presets

PTZ Presets section specifies Client ability to list the presets of a PTZ Node and move a PTZ Device to a specific preset.

1.3.28 PTZ Home Position

PTZ Home Position section specifies Client ability to move a PTZ Device to its home position.

1.3.29 PTZ - Set Preset

PTZ - Set Preset section specifies Client ability to store a preset.

1.3.30 PTZ Get Compatible Configurations

PTZ Get Compatible Configurations specifies Client ability to get PTZ configurations compatible with media profile from the device.

1.3.31 PTZ Media2 Profile Configuration

PTZ Media2 Profile Configuration specifies Client ability to add compatible with media profile PTZ configuration to a Media2 profile.

1.3.32 PTZ Set Configuration

PTZ Set Configuration specifies Client ability to modify PTZ configuration on the device.

1.3.33 Relay Outputs Using Device IO

Relay Outputs Using Device IO section specifies Client ability to control of Relay Outputs connected to a device.

1.3.34 Get Digital Inputs

Get Digital Inputs section specifies Client ability to retrieve of Digital Inputs connected to a device.

1.3.35 Get Supported Rules

Get Supported Rules section specifies Client ability to retrieve supported rules.

1.3.36 Get Rules

Get Rules section specifies Client ability to retrieve available rules.

1.3.37 Motion Detection

Motion Detection section specifies Client ability to create Motion Region Detector rules and to receive notifications of Motion Region Detector events.

1.4 Test Cases for Profile Optional Features

This section defines test cases which are optional for Profile T Client conformance.

1.4.1 Unsubscribe

Unsubscribe section defines Client ability to terminete subscription using Unsubscribe operation.

1.4.2 Keep Alive for Pull Point Event Handling

Keep Alive for Pull Point Event Handling section specifies Client ability to use keep alive for Pull Point Event Handling using PullMessages or Renew approach.

1.4.3 Audio Source Configuration Using Media2

Audio Source Configuration Using Media2 section specifies listing and modification of audio source configurations on Device.

1.4.4 Audio Output Configuration Using Media2

Audio Output Configuration Using Media2 section specifies listing and modification of audio output configurations on Device.

1.4.5 Audio Decoder Configuration Using Media2

Audio Decoder Configuration Using Media2 section specifies listing and modification of audio decoder configurations on Device.

1.4.6 Get Snapshot Uri Using Media2

Get Snapshot Uri Using Media2 section specifies Client ability to obtain a JPEG snapshot from the device.

2 Normative references

- ONVIF Conformance Process Specification:

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

- ONVIF Profile Policy:

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

- ONVIF Network Interface Specifications:

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

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

  www.iso.org/directives

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

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

- WS-BaseNotification:

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

- W3C SOAP 1.2, Part 1, Messaging Framework:

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

- W3C XML Schema Part 1: Structures Second Edition:

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

- W3C XML Schema Part 2: Datatypes Second Edition:

  "http://www.w3.org/TR/xmlschema-2/ [http://www.w3.org/TR/xmlschema-2/]

- W3C Web Services Addressing 1.0 – Core:

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

- ONVIF Profile T Specification:

  https://www.onvif.org/profiles/profile-t/

- IETF RFC 3016, RTP Payload Format for MPEG-4 Audio/Visual Streams:

  http://www.ietf.org/rfc/rfc3016

- IETF RFC 3984, RTP Payload Format for H.264 Video:

  http://www.ietf.org/rfc/rfc3984

- IETF RFC 7798, RTP Payload Format for High Efficiency Video Coding (HEVC):

  http://www.ietf.org/rfc/rfc7798.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.

Address	An address refers to a URI.
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.
Conversation	A Conversation is all exchanges between two MAC addresses that contains SOAP request and response.
Network	A network is an interconnected group of devices communicating using the Internet protocol.
Network Trace Capture file	Data file created by a network protocol analyzer software (such as Wireshark). Contains network packets data recorded during a live network communications.
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.
Client Test Tool	ONVIF Client Test Tool that tests ONVIF Client implementation towards the ONVIF Test Specification set.
Valid Device Response	Device has responded to specific request with code HTTP or RTSP 200 OK and SOAP fault message has not appeared.
Profile T	The Profile T Specification.
Configuration Entity	A network video device media abstract component that produces or consumes a media stream on the network, i.e. video and/or audio stream.
Digital PTZ	Function that diminishes or crops an image to adjust the image position and ratio.
Media Profile	Maps a video and audio sources and outputs encoders as well as PTZ and analytics configurations.
Metadata	All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications).

Reference Token	Token provided by the device to uniquely reference an instance of a physicalIO, configuration or profile.
Video Analytics	Algorithms or programs used to analyze video data and to generate data describing object location and behaviour.

3.3 Abbreviations

This section describes abbreviations used in this document.

HTTP	Hyper Text Transport Protocol.
HTTPS	Hyper Text Transport Protocol over Secure Socket Layer.
IP	Internet Protocol.
IPv4	Internet Protocol version 4.
TCP	Transport Control Protocol.
UDP	User Datagram Protocol.
URI	Uniform Resource Identifier.
WSDL	Web Services Description Language.
XML	eXtensible Markup Language.
RTCP	RTP Control Protocol.
RTP	Realtime Transport Protocol.
RTSP	Real Time Streaming Protocol.
SDP	Session Description Protocol.

3.4 Namespaces

Prefix and namespaces used in this test specification are listed in Table 1. These prefixes are not part of the standard and an implementation can use any prefix.

Table 3.1. Defined namespaces in this specification

Prefix	Namespace URI	Description
soapenv	http://www.w3.org/2003/05/soap-envelope	Envelope namespace as defined by SOAP 1.2 [SOAP 1.2, Part 1]
xs	http://www.w3.org/2001/XMLSchema	Instance namespace as defined by XS [XML-Schema, Part1] and [XMLSchema,Part 2]
xsi	http://www.w3.org/2001/XMLSchema-instance	XML schema instance namespace
tns1	http://www.onvif.org/ver10/topics	The namespace for the ONVIF topic namespace

Prefix	Namespace URI	Description
tt	http://www.onvif.org/ver10/schema	ONVIF XML schema descriptions
tds	http://www.onvif.org/ver10/device/wsdl	The namespace for the WSDL device service
tev	http://www.onvif.org/ver10/events/wsdl	The namespace for the WSDL event service
ter	http://www.onvif.org/ver10/error	The namespace for ONVIF defined faults
wsnt	http://docs.oasis-open.org/wsn/b-2	Schema namespace of the [WS-BaseNotification] specification.
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [WS-Addressing].
tr2	http://www.onvif.org/ver20/media/wsdl	The namespace for the WSDL Media2 service
tr2	http://www.onvif.org/ver20/media/wsdl	The namespace for the WSDL media2 service

4 Test Overview

This section provides information for the test setup procedure and required prerequisites that should be followed during test case execution.

An ONVIF client compliant to Profile T can configure, request, and control streaming of video, audio, and audio output data over an IP network from an ONVIF Device compliant to the Profile T. The client can also retrieve and receive standardized Tampering related events.

An ONVIF Profile is described by a fixed set of functionalities through a number of services that are provided by the ONVIF standard. A number of services and functionalities are mandatory for each type of ONVIF Profile. An ONVIF Device and ONVIF Client may support any combination of Profiles and other optional services and functionalities.

4.1 General

Test Cases are grouped depending on features. Each Test Cases group provides description of feature requirement level for Profiles, expected scenario under test and related test cases:

- Feature Level Requirement
- Expected Scenarios Under Test
- List of Test Cases

4.1.1 Feature Level Requirement

Feature Level Requirement item contains a feature ID, check condition based on Device features, required number of Devices and feature requirement level for the Profiles, which will be used for Profiles conformance.

To claim this Feature as supported Client shall pass Expected Scenario Under Test:

- for each Device, which supports Device Features defined in Check Condition Based on Device Features
- for at least with number of Devices specified in Required Number of Devices

If Feature Level Requirement is defined as Mandatory for some Profile, Client shall support this Feature to claim this Profile Conformance.

4.1.2 Expected Scenarios Under Test

Expected Scenarios Under Test item contains expected scenario under test, conditions when the feature will be defined as supported and as not supported.

4.1.3 Test Cases

Test Case items contain list of test cases which are related to feature. Test cases provide exact procedure of testing feature support conditions.

Each Test Case contains the following parts:

- Test Label - Unique label for each test
- Test Case ID - Unique ID for each test
- Profile Normative References - Requirement level for the feature under test is defined in Profile Specification. This reference is informative and will not be used in conformance procedure.
- Feature Under Test - Feature which is under current test. Typically a particular command or an event.
- Test Purpose - The purpose of current test case.
- Pre-Requisite - The pre-requisite defines when the test should be performed. In case if pre-reqiusite does not match, the test result will be NOT DETECTED.
- Test Procedure - scenario expected to be reflected in network trace file.
- Test Result - Passed and failed criteria of the test case. Depending on these criteria test result will be defined as PASSED or FAILED.

4.2 Test Setup

Collect Network traces files required by the test cases.

Collect Feature List XML files for Devices detected in the Network Trace files.

Client shall support all mandatory and conditional features listed in the Device Feature List XML file supplied for the Profiles supported by the Client.

For compatibility with the Profile T, the ONVIF Client shall follow the requirements of the conformance process. For details please see the latest ONVIF Conformance Process Specification.

4.3 Prerequisites

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

The Device shall be configured with an IPv4 address.

The Device shall be able to be discovered by the Client.

5 Test Cases for Profile Mandatory Features

5.1 HTTP Digest Test Cases

5.1.1 Feature Level Requirement:

Validated Feature: HTTP Digest authentication (HTTPDigest)

Check Condition based on Device Features: Digest

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile D Requirement: Mandatory

Profile G Requirement: Mandatory

Profile S Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.1.2 Expected Scenarios Under Test:

1. Client invokes a specific command which is under testing without any user credentials (no UsernameToken, no HTTP Digest authentication header).
2. Device returns HTTP 401 Unauthorized error along with WWW-Authentication: Digest header.
3. Client re-sends request with HTTP Digest Authentication header corresponding to header provided in device response.
4. Device sends a valid response to this request.
5. Client is considered as supporting HTTP Digest if the following conditions are met:
   - Device returns a valid response to specific request with HTTP Digest authentication header.
6. Client is considered as NOT supporting HTTP Digest if the following is TRUE:
   - All HTTP Digest attempts detected are failed.

5.1.3 HTTP DIGEST

Test Label: Security - HTTP Digest Authentication.

Test Case ID: HTTPDIGEST-1

Feature Under Test: HTTP Digest (HTTPDigest_HTTPDigestAuthentication)

Test Purpose: To verify that the Client supports the HTTP Digest Authentication for HTTP level security.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with HTTP Digest Authentication present.

Test Procedure (expected to be reflected in network trace file):

1. Client sends a request that requires authentication (e.g. GetUsers) to the Device without any authentication.
2. Device rejects the request with HTTP error code 401 AND an HTTP Digest challenge.
3. Client sends a valid request with HTTP Digest Authentication.
4. Device accepts the correct request with response code HTTP 200 OK.

Test Result:

PASS -

- [S1] Client request contains (HTTP GET method OR HTTP POST method) without any authentication AND
- Client HTTP GET request has a proper hierarchy (refer to [RFC 1945]) AND
  - [S2] Device response contains "HTTP/* 401 Unauthorized" AND
  - [S3] Device response contains "realm=*" element AND
  - [S4] Device response contains "nonce=*" element AND
  - [S5] Client request contains (HTTP GET method OR HTTP POST method) with "Authorization: Digest username=*" element AND
- Client HTTP GET request with HTTP Authentication has a proper hierarchy (refer to [RFC 1945]) AND

- [S6] Client request contains "realm=*" element with value from Device response AND
- [S7] Client request contains "nonce=*" element with value from Device response AND
- [S8] Client request contains "uri=*" element AND
- [S9] Device response contains "HTTP/* 200 OK".

FAIL -

- The Client failed PASS criteria.

5.2 Capabilities Test Cases

5.2.1 Feature Level Requirement:

Validated Feature: Capabilities (Capabilities)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile S Requirement: Mandatory

Profile T Requirement: Mandatory

5.2.2 Expected Scenarios Under Test:

1. Client invokes a specific Capabilities command which is under testing.
2. Client is considered as supporting Capabilities if the following conditions are met:
   - Device returns a valid response to GetServices request OR
   - Device returns a valid response to GetCapabilities request.
3. Client is considered as NOT supporting Capabilities if the following is TRUE:
   - No Valid Device Response to GetServices request AND
   - No Valid Device Response to GetCapabilities request.

5.2.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

Test Purpose: To verify that Device Capabilities is received using GetServices request.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetServices command present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetServices request message to retrieve all services of the Device.
2. Verify that GetServicesResponse message from the Device contains code HTTP 200 OK without SOAP Fault.

Test Result:

PASS -

- Client **GetServices** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetServices** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetServices>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetServicesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.2.4 GET CAPABILITIES

Test Label: Capabilities - Get Device Capabilities

Test Case ID: CAPABILITIES-2

Feature Under Test: Get Capabilities (Capabilities_GetCapabilities)

Test Purpose: To verify that Device Capabilities is received using GetCapabilities request.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetCapabilities command present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetCapabilities request message to retrieve Device Capabilities of the Device.
2. Verify that GetCapabilitiesResponse response message from the Device contains code HTTP 200 OK without SOAP Fault.

Test Result:

PASS -

- Client **GetCapabilities** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetCapabilities** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetCapabilities>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetCapabilitiesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.3 Get Services Test Cases

5.3.1 Feature Level Requirement:

Validated Feature: Get Services (GetServices)

Check Condition based on Device Features: GetServices is supported by Device.

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile D Requirement: Mandatory

Profile C Requirement: Mandatory

Profile G Requirement: Mandatory

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.3.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a services using **GetServices** commad.
2. Client is considered as supporting Get Services if the following conditions are met:
   - Client supports Capabilities_GetServicesRequest feature (please see CAPABILITIES-1 GET SERVICES section).
3. Client is considered as NOT supporting Get Services if ANY of the following is TRUE:
   - Client does not support Capabilities_GetServicesRequest feature (please see CAPABILITIES-1 GET SERVICES section).

5.3.3 GET SERVICES

Test Label: Capabilities - Determine the available Services

Test Case ID: CAPABILITIES-1

Feature Under Test: Get Services (Capabilities_GetServicesRequest)

Test Purpose: To verify that Device Capabilities is received using GetServices request.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetServices command present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetServices request message to retrieve all services of the Device.
2. Verify that GetServicesResponse message from the Device contains code HTTP 200 OK without SOAP Fault.

Test Result:

PASS -

- Client **GetServices** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetServices** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetServices>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetServicesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.4 Event Handling Test Cases

5.4.1 Feature Level Requirement:

Validated Feature: Event Handling (EventHandling)

Check Condition based on Device Features: Pull Point Notification OR WS Basic Notification OR Profile S OR Metadata under Media2 service is supported by Device.

Required Number of Devices: 3

Profile S Requirement: Conditional

Profile G Requirement: Conditional

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

5.4.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Event Handling.
2. Client is considered as supporting Event Handling if the following conditions are met:
   - Client is able to handle the Pull Point Event mechanism OR
   - Client is able to handle the Base Notification Event mechanism OR

- Client is able to handle the Metadata Streaming by supporting EventHandling_MetadataStreamingUsingMedia feature (please see EVENTHANDLING-4 METADATA STREAMING USING MEDIA section) OR Media2_MetadataStreaming_MetadataStreamingUsingMedia2 feature (please see MEDIA2_METADATASTREAMING-1 METADATA STREAMING USING MEDIA2 section).

3. Client is considered as NOT supporting Event Handling if the following is TRUE:
   - All Pull Point attempts detected have failed AND
   - All Base Notification attempts detected have failed AND
   - All Metadata Streaming attempts detected have failed.

5.4.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Test Purpose: To verify that the Client is able to retrieve events using Pull Point.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Pull Point event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes PullMessages command with Timeout and MessageLimit elements.
4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.4.4 BASE NOTIFICATION

Test Label: Event Handling - Basic Notification

Test Case ID: EVENTHANDLING-2

Feature Under Test: Base Notification (EventHandling_WSBaseNotification)

Test Purpose: To verify that the Client is able to retrieve events using WS-Base Notification.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Basic Notification event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes Subscribe message with ConsumerReference element.
2. Device responds with code HTTP 200 OK and SubscribeResponse message.

Test Result:

PASS -

- Client **Subscribe** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Subscribe** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<Subscribe>" tag after the "<Body>" tag AND
  - [S4] Device response contains "HTTP/* 200 OK" AND
  - [S5] Device response contains "<SubscribeResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.4.5 METADATA STREAMING USING MEDIA

Test Label: Event Handling - Metadata Streaming Using Media Streaming

Test Case ID: EVENTHANDLING-4

Feature Under Test: Metadata Streaming (EventHandling_MetadataStreamingUsingMedia)

Test Purpose: To verify that the Client is able to retrieve the Metadata Streaming using Media Service.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Metadata Streaming event type using Media Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for Media service for media profile that contains Video Source Configuration and Metadata Configuration. GetStreamUri request is set for RTP-Unicast/UDP OR RTP-Multicast/UDP OR RTP/RTSP/TCP OR RTP-Unicast/RTSP/HTTP/TCP transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "application" and with encoding name "vnd.onvif.metadata" or "vnd.onvif.metadata.gzip" or "vnd.onvif.metadata.exi.onvif" or "vnd.onvif.metadata.exi.ext".

5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for metadata streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "application" (m=application) with sessions attribute "rtpmap" with encoding name "vnd.onvif.metadata" OR "vnd.onvif.metadata.gzip" OR "vnd.onvif.metadata.exi.onvif" OR "vnd.onvif.metadata.exi.ext" (see ONVIF Streaming Spec) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S3] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S4] It invoked after the Client **RTSP DESCRIBE** request AND
  - [S5] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S6] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND

- There is a Device response on the **GetStreamUri** request invoked for Media Service in Test Procedure fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S10] It received before the Client **RTSP DESCRIBE** request AND
  - [S11] It contains **trt:MediaUri\tt:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S12] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S13] It invoked after the Client **RTSP SETUP** request AND
  - [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S16] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S17] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S18] It invoked after the Client **RTSP PLAY** request AND
  - [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.5 Set Synchronization Point (Event Service) Test Cases

5.5.1 Feature Level Requirement:

Validated Feature: Set Synchronization Point (SetSynchronizationPoint)

Check Condition based on Device Features: Pull Point Notification OR WS-Basic Notification is supported by Device.

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile S Requirement: Optional

Profile G Requirement: Optional

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

5.5.2 Expected Scenarios Under Test:

1. Client connects to Device to synchronize property states.
2. Client is considered as supporting Set Synchronization Point (Event Service) if the following conditions are met:
   - Client is able to synchronize property states using **SetSynchronizationPoint** operation for subscribtions AND
3. Client is considered as NOT supporting Set Synchronization Point (Event Service) if the following is TRUE:
   - No valid responses for **SetSynchronizationPoint** request OR
   - **SetSynchronizationPoint** request does not contains valid **wsa:Action** header.

5.5.3 SET SYNCHRONIZATION POINT (EVENT SERVICE)

Test Label: Set Synchronization Point - Set Synchronization Point

Test Case ID: SETSYNCHRONIZATIONPOINT-1

Feature Under Test: Set Synchronization Point (SetSynchronizationPoint_SetSynchronizationPointAction)

Test Purpose: To verify that the Client is able to use **SetSynchronizationPoint** operation for subscribtion.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetSynchronizationPoint** operations present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetSynchronizationPoint** message with valid **wsa:Action** header to synchronize its properties with the properties of the device.
2. Device responses with code HTTP 200 OK and **SetSynchronizationPointResponse** message.

Test Result:

PASS -

- Client **SetSynchronizationPoint** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetSynchronizationPoint** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tev:SetSynchronizationPoint** AND
  - [S2] It contains **wsa:Action** element in header equal to "http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointRequest" AND
- Device response on the **SetSynchronizationPoint** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tev:SetSynchronizationPointResponse**

FAIL -

- The Client failed PASS criteria.

5.6 Discovery Test Cases

5.6.1 Feature Level Requirement:

Validated Feature: Discovery (Discovery)

Check Condition based on Device Features: Discovery

Required Number of Devices: 3

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile A Requirement: Mandatory

Profile T Requirement: Mandatory

Profile D Requirement: Mandatory

Profile M Requirement: Mandatory

5.6.2 Expected Scenarios Under Test:

1. Client sends Probe message to multicast IP address 239.255.255.250 and port 3702 to locate services on a local network.
2. Client is considered as supporting Discovery if the following conditions are met:
   - Probe request detected AND at least one ProbeMatch response detected
3. Client is considered as NOT supporting Discovery if the following is TRUE:
   - No Valid Device Response to Probe request.

5.6.3 WS-DISCOVERY

Test Label: Discovery - WS-Discovery

Test Case ID: DISCOVERY-1

Feature Under Test: WS-Discovery (Discovery_WSDiscovery)

Test Purpose: To verify that Client is able to send Probe request and receive ProbeMatch response from Device.

Pre-Requisite:

- The Network Trace Capture files contain at least one Client Probe request to multicast IP address and one ProbeMatch response from Device directly to the Client.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes Probe request message to multicast IP address 239.255.255.250 and port 3702.
2. Device sends ProbeMatch message directly to the Client.

Test Result:

PASS -

- Client **Probe** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Probe** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<Action>" tag after the "<Header>" tag AND
  - [S2] "<Action>" includes URL address which ends with "Probe" value AND
  - [S3] Client request contains "<MessageID>" with non-empty string value AND
  - [S4] Client request contains "<Probe>" tag after the "<Body>" tag AND
  - [S5] Device response message contains "<ProbeMatches>" tag after the "<Body>" tag.

FAIL -

- The Client failed PASS criteria.

5.7 Device Discovery Type Filter Test Cases

5.7.1 Feature Level Requirement:

Validated Feature: Device Discovery Type Filter (DeviceDiscoveryTypeFilter)

Check Condition based on Device Features: Device Discovery Type is supported by Device.

Required Number of Devices: 3

Profile S Requirement: None

Profile A Requirement: Mandatory

Profile C Requirement: Conditional

Profile D Requirement: Mandatory

Profile G Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.7.2 Expected Scenarios Under Test:

1. Client sends Probe message to multicast IPv4 address 239.255.255.250 or multicast IPv6 address [FF02::C] and port 3702 with Types filter that contains **tds:Device** or with skipped Types filter.
2. Client is considered as supporting Device Discovery Type if the following conditions are met:
   - **Probe** Client message that fulfills the following requirement is detected:
     - Types filter contains tds:Device or empty or skipped AND
     - Probe is sent to multicast IPv4 address 239.255.255.250 or multicast IPv6 address [FF02::C] AND
     - Probe is sent to UDP port 3702 AND
   - There is **ProbeMatch** Device message that correspond to Client **Probe**.
3. Client is considered as NOT supporting Device Discovery Type if the following is TRUE:
   - No valid Device **ProbeMatch** message that is correspond to Client **Probe** message.

5.7.3 DEVICE DISCOVERY TYPE FILTER

Test Label: Discovery - Device Discovery Type Filter

Test Case ID: DEVICEDISCOVERYTYPEFILTER-1

Feature Under Test: Device Discovery Type Filter (DeviceDiscoveryTypeFilter_DeviceDiscoveryFilter)

Test Purpose: To verify that Client is able to discover devices with Device Discovery Type.

Pre-Requisite:

- The Network Trace Capture files contains at least one Client Probe message that does not filter out devices with Device Discovery Type that is sent to multicast WS-Discovery address.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes Probe request message to multicast IPv4 address 239.255.255.250 or multicast IPv6 address [FF02::C] and port 3702 with **Types** that contains tds:Device.
2. Device sends ProbeMatch message to the Client.

Test Result:

PASS -

- Client **Probe** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Probe** request in Test Procedure fulfills the following requirements:
  - [S1] It is sent to 239.255.255.250 IPv4 address OR [FF02::C] IPv6 address AND
  - [S2] It is sent to 3702 UDP port AND
  - [S3] **soapenv:Envelope/soapenv:Header** element has child element **wsadis:Action** AND
  - [S4] **wsadis:Action** includes URL address which ends with "Probe" value AND
  - [S5] **soapenv:Envelope/soapenv:Header** element has child element **wsadis:MessageID** with non-empty string value AND
  - [S6] **soapenv:Body** element has child element **d:Probe** AND
  - [S7] IF **d:Probe** element has child element **d:Types** THEN it contains value is equal to **tds:Device** OR empty string value AND
  - [S8] There is Device **ProbeMatches** message in test procedure that fulfills the following requirements:
    - [S9] **soapenv:Body** element has child element **d:ProbeMatches** AND
    - [S10] **soapenv:Envelope/soapenv:Header/wsadis:RelatesTo** element value is equeal to **soapenv:Envelope/soapenv:Header/wsadis:MessageID** value in **Probe** message AND

PASS WITH WARNING -

- **d:Probe/d:Types** element is skipped OR
- **d:Probe/d:Types** element has empty string value.

FAIL -

- The Client failed PASS criteria.

5.8 Network Configuration Test Cases

5.8.1 Feature Level Requirement:

Validated Feature: Network Configuration (NetworkConfiguration)

Check Condition based on Device Features: Network Configuration

Required Number of Devices: 3

Profile A Requirement: Conditional

Profile C Requirement: Conditional

Profile D Requirement: Mandatory

Profile G Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.8.2 Expected Scenarios Under Test:

1. Client connects to Device to configure network settings.
2. Client is considered as supporting Network Configuration if the following conditions are met:
   - Client is able to list network interfaces of Device using the GetNetworkInterfaces operation AND
   - Client is able to set network interfaces of Device using the SetNetworkInterfaces operation AND
   - Client is able to list default gateway of Device using the GetNetworkDefaultGateway operation AND
   - Client is able set default gateway of Device using the SetNetworkDefaultGateway operation.

3. Client is considered as NOT supporting Network Configuration if ANY of the following is TRUE:
   - No Valid Device Response to GetNetworkInterfaces request OR
   - No Valid Device Response to SetNetworkInterfaces request OR
   - No Valid Device Response to GetNetworkDefaultGateway request OR
   - No Valid Device Response to SetNetworkDefaultGateway request.

5.8.3 GET NETWORK INTERFACES

Test Label: Network Configuration - Get Network Interfaces

Test Case ID: NETWORKCONFIGURATION-1

Feature Under Test: Get Network Interfaces (NetworkConfiguration_GetNetworkInterfaces)

Test Purpose: To verify that Client is able to list network interfaces of Device using the GetNetworkInterfaces operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetNetworkInterfaces operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetNetworkInterfaces request message to get network interface configuration from Device.
2. Device responds with code HTTP 200 OK and GetNetworkInterfacesResponse message.

Test Result:

PASS -

- Client **GetNetworkInterfaces** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetNetworkInterfaces** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNetworkInterfaces>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetNetworkInterfacesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.8.4 SET NETWORK INTERFACES

Test Label: Network Configuration - Set Network Interfaces

Test Case ID: NETWORKCONFIGURATION-2

Feature Under Test: Set Network Interfaces (NetworkConfiguration_SetNetworkInterfaces)

Test Purpose: To verify that Client is able to set network interfaces of Device using the SetNetworkInterfaces operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with SetNetworkInterfaces operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetNetworkInterfaces request message to set the network interface configuration on Device.
2. Device responds with code HTTP 200 OK and SetNetworkInterfacesResponse message.

Test Result:

PASS -

- Client **SetNetworkInterfaces** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetNetworkInterfaces** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetNetworkInterfaces>" tag after the "<Body>" tag AND
  - [S2] "<SetNetworkInterfaces>" includes tag: "<InterfaceToken>" with non-empty string value of specific token AND
  - [S4] Device response contains "HTTP/* 200 OK" AND
  - [S5] Device response contains "<SetNetworkInterfacesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.8.5 GET NETWORK DEFAULT GATEWAY

Test Label: Network Configuration - Get Network Default Gateway

Test Case ID: NETWORKCONFIGURATION-3

Feature Under Test: Get Network Default Gateway (NetworkConfiguration_GetNetworkDefaultGateway)

Test Purpose: To verify that Client is able to list default gateway of Device using the GetNetworkDefaultGateway operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetNetworkDefaultGateway operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetNetworkDefaultGateway request message to get the default gateway settings from Device.
2. Device responds with code HTTP 200 OK and GetNetworkDefaultGatewayResponse message.

Test Result:

PASS -

- Client **GetNetworkDefaultGateway** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetNetworkDefaultGateway** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNetworkDefaultGateway>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetNetworkDefaultGatewayResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.8.6 SET NETWORK DEFAULT GATEWAY

Test Label: Network Configuration - Set Network Default Gateway

Test Case ID: NETWORKCONFIGURATION-4

Feature Under Test: Set Network Default Gateway (NetworkConfiguration_SetNetworkDefaultGateway)

Test Purpose: To verify that Client is able to set default gateway of Device using the SetNetworkDefaultGateway operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with SetNetworkDefaultGateway operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetNetworkDefaultGateway request message to set the default gateway settings on Device.
2. Device responds with code HTTP 200 OK and SetNetworkDefaultGatewayResponse message.

Test Result:

PASS -

- Client **SetNetworkDefaultGateway** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetNetworkDefaultGateway** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetNetworkDefaultGateway>" tag after the "<Body>" tag AND
  - [S2] "<SetNetworkDefaultGateway>" includes tag: EITHER "<IPv4Address>" OR "<IPv6Address>" with specific IP address value AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<SetNetworkDefaultGatewayResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.9 HTTP Digest Authentication for RTSP Test Cases

5.9.1 Feature Level Requirement:

Validated Feature: HTTP Digest Authentication for RTSP (HTTPDigestForRTSP)

Check Condition based on Device Features: Profile T or Profile M

Required Number of Devices: 3

Profile S Requirement: None

Profile G Requirement: None

Profile A Requirement: None

Profile C Requirement: None

Profile T Requirement: Mandatory

Profile M Requirement: Mandatory

5.9.2 Expected Scenarios Under Test:

1. Client invokes a specific RTSP command which is under testing without any user credentials (no UsernameToken, no HTTP Digest authentication header).
2. IF Device returns HTTP 401 Unauthorized error along with WWW-Authentication: Digest header, then Client resends RTSP command with WWW-Authenticate header.
3. Client is considered as supporting HTTP Digest Authentication for RTSP if the following conditions are met:
   - Device returns a valid response to specific RTSP request with HTTP Digest authentication header.
4. Client is considered as NOT supporting HTTP Digest Authentication for RTSP if the following is TRUE:
   - All HTTP Digest attempts detected for RTSP are failed.

5.9.3 HTTP DIGEST AUTHENTICATION FOR RTSP

Test Label: HTTP Digest For RTSP

Test Case ID: HTTPDIGESTFORRTSP-1

Feature Under Test: HTTP Digest For RTSP (HTTPDigestForRTSP_HTTPDigestForRTSPTest)

Test Purpose: To verify that the Client supports the HTTP Digest Authentication for RTSP level security.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with HTTP Digest Authentication for RTSP commands present

Test Procedure (expected to be reflected in network trace file):

1. Client sends a RTSP request that requires authentication (e.g. DESCRIBE) to the Device without any authentication.
2. Device rejects the request with a RTSP 401 status code, AND a WWW-Authenticate Response Header.
3. Client re-sends the RTSP request with a Authorization Request Header.
4. Device accepts the correct request with RTSP 200 OK status code.

Test Result:

PASS -

- There is Client RTSP request in Test Procedure that does not contain any authentication AND
- Device response on the Client RTSP request fulfills the following requirements:
  - It has RTSP 401 status code AND
  - WWW-Authenticate Response Header contains challenge = "Digest" element AND
  - WW-Authenticate Response Header contains "realm=*" element AND
  - WW-Authenticate Response Header contains "nonce=*" element AND
- There is Client RTSP request in Test Procedure that fulfills the following requirements
  - WW-Authenticate Request Header credentials = "Digest" element AND
  - WW-Authenticate Request Header contains "realm=*" element with value from Device response AND
  - WW-Authenticate Request Header contains "nonce=*" element with value from Device response AND

- WW-Authenticate Request Header contains "uri=*" element AND

- Device responds with code RTSP 200 OK.

FAIL -

- The Client failed PASS criteria.

5.10 Get Profiles Using Media2 Test Cases

5.10.1 Feature Level Normative Reference:

Validated Feature: Get Profiles Using Media2 (Media2_GetProfiles)

Check Condition based on Device Features: Media2 Service is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.10.2 Expected Scenarios Under Test:

1. Client connects to Device to retrive media profiles from the DUT.
2. Client is considered as supporting Get Profiles if the following conditions are met:
   - Client is able to retrieve media profiles using **GetProfiles** operation (Media2 Service).
3. Client is considered as NOT supporting Get Profiles if ANY of the following is TRUE:
   - No valid response to **GetProfiles** request (Media2 Service).

5.10.3 GET PROFILES USING MEDIA2

Test Label: GetProfiles

Test Case ID: MEDIA2_GETPROFILES-1

Feature Under Test: Get Profiles Using Media2 (Media2_GetProfiles_Media2_GetProfilesRequest)

Test Purpose: To verify that media profiles provided by Device are received by Client using the **GetProfiles** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetProfiles** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetProfiles** request message to retrieve a media profile or a list of media profiles from the Device.
2. Device responds with code HTTP 200 OK and **GetProfilesResponse** message.

Test Result:

PASS -

- Client **GetProfiles** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetProfiles** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetProfiles** AND
- Device response on the **GetProfiles** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetProfilesResponse**.

FAIL -

- The Client failed PASS criteria.

5.11 Get Stream Uri Using Media2 Test Cases

5.11.1 Feature Level Normative Reference:

Validated Feature: Get Stream URI Using Media2 (Media2_GetStreamURI)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.11.2 Expected Scenarios Under Test:

1. Client connects to Device to retrive stream uri.
2. Client is considered as supporting Get Stream Uri if the following conditions are met:
   - Client is able to get the stream URI for the selected media profile using **GetStreamURI** operation (Media2 Service).
3. Client is considered as NOT supporting Get Stream Uri if ANY of the following is TRUE:
   - No valid response to **GetStreamURI** request (Media2 Service).

5.11.3 GET STREAM URI USING MEDIA2

Test Label: GetStreamUri

Test Case ID: MEDIA2_GETSTREAMURI-1

Feature Under Test: Get Stream URI Using Media2 (Media2_GetStreamURI_Media2_GetStreamURIRequest)

Test Purpose: To verify that stream URI provided by Device is received by Client using the **GetStreamUri** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message to retrieve a stream URI from the Device.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.

Test Result:

PASS -

- Client **GetStreamUri** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetStreamUri** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetStreamUri** AND

- Device response on the **GetStreamUri** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetStreamUriResponse**.

FAIL -

- The Client failed PASS criteria.

5.12 Media Streaming Using Media2 Test Cases

5.12.1 Feature Level Requirement:

Validated Feature: Media Streaming Using Media2 (Media2_MediaStreaming)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.12.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Media Streaming.
2. Client is considered as supporting Media Streaming if the following conditions are met:
   - Device supports Media2_GetProfiles_Media2_GetProfilesRequest feature (please see MEDIA2_GETPROFILES-1 GET PROFILES USING MEDIA2 section) AND
   - Device supports Media2_GetStreamURI_Media2_GetStreamURIRequest feature (please see MEDIA2_GETSTREAMURI-1 GET STREAM URI USING MEDIA2 section) AND
   - Stream was successfully established by Client using UDP protocol OR HTTP protocol.
   - Stream was successfully established by Client using RTSP protocol (if supported).
3. Client is considered as NOT supporting Media Streaming if the following is TRUE:
   - Device does not Media2_GetProfiles_Media2_GetProfilesRequest feature (please see MEDIA2_GETPROFILES-1 GET PROFILES USING MEDIA2 section) OR

- Device does not support Media2_GetStreamURI_Media2_GetStreamURIRequest feature (please see MEDIA2_GETSTREAMURI-1 GET STREAM URI USING MEDIA2 section) OR
- Client is unable to establish stream using UDP protocol OR HTTP protocol OR
- Client is unable to establish stream using RTSP protocol if detected.

5.12.3 GET PROFILES USING MEDIA2

Test Label: GetProfiles

Test Case ID: MEDIA2_GETPROFILES-1

Feature Under Test: Get Profiles Using Media2 (Media2_GetProfiles_Media2_GetProfilesRequest)

Test Purpose: To verify that media profiles provided by Device are received by Client using the **GetProfiles** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetProfiles** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetProfiles** request message to retrieve a media profile or a list of media profiles from the Device.
2. Device responds with code HTTP 200 OK and **GetProfilesResponse** message.

Test Result:

PASS -

- Client **GetProfiles** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetProfiles** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetProfiles** AND
- Device response on the **GetProfiles** request fulfills the following requirements:

- [S2] It has HTTP 200 response code AND
- [S3] **soapenv:Body** element has child element **tr2:GetProfilesResponse**.

FAIL -

- The Client failed PASS criteria.

5.12.4 GET STREAM URI USING MEDIA2

Test Label: GetStreamUri

Test Case ID: MEDIA2_GETSTREAMURI-1

Feature Under Test: Get Stream URI Using Media2 (Media2_GetStreamURI_Media2_GetStreamURIRequest)

Test Purpose: To verify that stream URI provided by Device is received by Client using the **GetStreamUri** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message to retrieve a stream URI from the Device.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.

Test Result:

PASS -

- Client **GetStreamUri** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetStreamUri** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetStreamUri** AND
- Device response on the **GetStreamUri** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND

- [S3] **soapenv:Body** element has child element **tr2:GetStreamUriResponse**.

FAIL -

- The Client failed PASS criteria.

5.12.5 STREAMING OVER RTSP USING MEDIA2

Test Label: Media Streaming - RTSP

Test Case ID: MEDIA2_MEDIASTREAMING-1

Feature Under Test: Streaming Over RTSP Using Media2 (Media2_MediaStreaming_Media2_RTSP)

Test Purpose: To verify that stream over RTSP protocol was successfully established between Client and Device using RTSP commands and then successfully stopped.

Pre-Requisite:

- Device supports RTSP streaming for Media2 Service (Media2_RTPRTSPTCP).
- The Network Trace Capture files contains at least one Conversation between Client and Device with RTSP SETUP request with transport parameter as "RTP/AVP/TCP" and which does not contain Require header with "onvif-replay" value and which is not tunneled in HTTP present.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service with **rt2:Protocol** element value equals to RTSP.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile with Transport element with "RTSP" value.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK.
5. Client invokes **RTSP SETUP** request with **Transport** tag in RTSP header that contains "RTP/AVP/TCP" and without "onvif-replay" Require header to set media session parameters.

6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

PASS -

- Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S1] It contains **Transport** request header field with value is equal to "RTP/AVP/TCP" (transport=RTP, profile=AVP, lower-transport=TCP) (see [RFC 2326]) AND
  - [S2] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
  - [S3] It is not tunneled in HTTP AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S4] It has RTSP 200 response code AND
- There is Client **RTSP DESCRIBE** request in Test Procedure fulfills the following requirements:
  - [S5] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S6] It invoked before the Client **RTSP SETUP** request AND
  - [S7] It is not tunneled in HTTP AND
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S8] SDP packet contains media type with Control URL that was used to send **RTSP SETUP** (see [RFC 2326, C.1.1 Control URL]) AND
  - [S9] It has RTSP 200 response code AND
- There is a Client **GetStreamUri** request in Test Procedure fulfills the following requirements:
  - [S10] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S11] It invoked before the Client **RTSP DESCRIBE** request AND

- [S12] **tr2:GetStreamUri/tr2:Protocol** element value is equal to "RTSP"

- Device response on the **GetStreamUri** request to Media2 Service fulfills the following requirements:
  - [S13] It has HTTP 200 response code AND
  - [S14] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S15] It is invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S16] It invoked after the Client **RTSP SETUP** request AND
  - [S17] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S18] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
  - [S19] It is not tunneled in HTTP AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S20] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S21] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S22] It invoked after the Client **RTSP PLAY** request AND
  - [S23] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S24] It is not tunneled in HTTP AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S25] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.12.6 STREAMING OVER UDP USING MEDIA2

Test Label: Media Streaming - UDP

Test Case ID: MEDIA2_MEDIASTREAMING-2

Feature Under Test: Streaming Over UDP Using Media2 (Media2_MediaStreaming_Media2_UDP)

Test Purpose: To verify that stream over UDP protocol was successfully established between Client and Device using RTSP commands and then successfully stopped.

Pre-Requisite:

- Device supports Media2 Real Time Streaming (Media2_RealTimeStreaming).
- The Network Trace Capture files contains at least one Conversation between Client and Device with RTSP SETUP request with transport parameter as "RTP/AVP/UDP" or "RTP/AVP" and which does not contain Require header with "onvif-replay" value present.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service with **rt2:Protocol** element value equals to "RtspUnicast" or "RtspMulticast".

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile with Transport element with "RtspUnicast" value or "RtspMulticast" value.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK.
5. Client invokes **RTSP SETUP** request with **Transport** tag in RTSP header that contains "RTP/AVP/UDP" or "RTP/AVP" and without "onvif-replay" Require header to set media session parameters.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.

8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

PASS -

- Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S1] It contains **Transport** request header field with value is equal to "RTP/AVP/UDP" OR "RTP/AVP" (transport=RTP, profile=AVP, lower-transport=TCP or skipped) (see [RFC 2326]) AND
  - [S2] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S3] It has RTSP 200 response code AND
- There is Client **RTSP DESCRIBE** request in Test Procedure fulfills the following requirements:
  - [S4] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S5] It invoked before the Client **RTSP SETUP** request AND
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S6] SDP packet contains media type with Control URL that was used to send **RTSP SETUP** (see [RFC 2326, C.1.1 Control URL]) AND
  - [S7] It has RTSP 200 response code AND
- There is a Device **GetStreamUri** request in Test Procedure fulfills the following requirements:
  - [S8] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S9] It invoked before the Client **RTSP DESCRIBE** request AND
  - [S10] **tr2:GetStreamUri/tr2:Protocol** element value is equal to "RtspUnicast" or "RtspMulticast"
- Device response on the **GetStreamUri** request to Media2 Service fulfills the following requirements:

- [S11] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- [S12] It has HTTP 200 response code AND

- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S13] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S14] It invoked after the Client **RTSP SETUP** request AND
  - [S15] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S16] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S17] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S18] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S19] It invoked after the Client **RTSP PLAY** request AND
  - [S20] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S21] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.12.7 STREAMING OVER HTTP USING MEDIA2

Test Label: Media Streaming - HTTP

Test Case ID: MEDIA2_MEDIASTREAMING-3

Feature Under Test: Streaming Over HTTP Using Media2 (Media2_MediaStreaming_Media2_HTTP)

Test Purpose: To verify that stream over HTTP protocol was successfully established between Client and Device using RTSP commands and then successfully stopped.

Pre-Requisite:

- Device supports HTTP streaming for Media2 Service (Media2_RTPRTSPHTTP).
- The Network Trace Capture files contains at least one Conversation between Client and Device with RTSP SETUP request with transport parameter as "RTP/AVP/TCP" and which does not contain Require header with "onvif-replay" value and which is tunneled in HTTP present.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service with **rt2:Protocol** element value equals to RtspOverHttp.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile with Protocol element with "RtspOverHttp" value.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request in HTTP tunnel to retrieve media stream description.
4. Device responds with code RTSP 200 OK.
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header in HTTP tunnel with **Transport** tag in RTSP header that contains "RTP/AVP/TCP" to set media session parameters.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header in HTTP tunnel to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request in HTTP tunnel to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

PASS -

- Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S1] It contains **Transport** request header field with value is equal to "RTP/AVP/TCP" (transport=RTP, profile=AVP, lower-transport=TCP) (see [RFC 2326]) AND
  - [S2] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
  - [S3] It is tunneled in HTTP AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S4] It has RTSP 200 response code AND
- There is Client **RTSP DESCRIBE** request in Test Procedure fulfills the following requirements:
  - [S5] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S6] It invoked before the Client **RTSP SETUP** request AND
  - [S7] It is tunneled in HTTP AND
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S8] SDP packet contains media type with Control URL that was used to send **RTSP SETUP** (see [RFC 2326, C.1.1 Control URL]) AND
  - [S9] It has RTSP 200 response code AND
- There is a Device **GetStreamUri** request in Test Procedure fulfills the following requirements:
  - [S10] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S11] It invoked before the Client **RTSP DESCRIBE** request AND
  - [S12] **tr2:GetStreamUri/tr2:Protocol** element value is equal to "RtspOverHttp"
- Device response on the **GetStreamUri** request to Media2 Service fulfills the following requirements:
  - [S13] It has HTTP 200 response code AND
  - [S14] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND

- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S15] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S16] It invoked after the Client **RTSP SETUP** request AND
  - [S17] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S18] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
  - [S19] It is tunneled in HTTP AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S20] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S21] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S22] It invoked after the Client **RTSP PLAY** request AND
  - [S23] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S24] It is tunneled in HTTP AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S25] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.13 H264 Video Streaming Using Media2 Test Cases

5.13.1 Feature Level Normative Reference:

Validated Feature: H264 Video Streaming Using Media2 (Media2_VideoStreaming_H264)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and H264 (Media2 Service) are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.13.2 Expected Scenarios Under Test:

1. Client connects to Device to setup and control of H264 video streaming.
2. Client is considered as supporting H264 Video Streaming if the following conditions are met:
   - Client is able to receive a stream using Media2 and decode H.264 video using the selected Media Profile over RTSP.
3. Client is considered as NOT supporting H264 Video Streaming if ANY of the following is TRUE:
   - Client is unable to initiate and retrieve video stream using Media2 with H.264 encoding type.

5.13.3 H264 VIDEO STREAMING USING MEDIA2

Test Label: Video Streaming - H264

Test Case ID: MEDIA2_VIDEOSTREAMING_H264-1

Feature Under Test: H264 Video Streaming Using Media2 (Media2_VideoStreaming_H264_Media2_H264VideoStreaming)

Test Purpose: To verify that the Client is able to initiate and retrieve a video stream with H264 encoding type.

Pre-Requisite:

- Device supports H264 encoding for Video Streaming for Media2 Service (Media2_H264).
- The Network Trace Capture files contains at least one Conversation between Client and Device with Video Streaming of H264 encoding type.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports H264 encoding for Video Streaming using Media2.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile that contains Video Source Configuration and Video Encoder Configuration with H264 Encoding value. GetStreamUri request is set for RtspUnicast OR RtspMulticast OR RTSP OR RtspOverHttp transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "video" and with encoding name "H264".
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for H264 video streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "video" (m=video) with sessions attribute "rtpmap" with encoding name "H264" (see [RFC 3984], item 8.2.1. Mapping of MIME Parameters to SDP) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S3] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND

- [S4] It invoked after the Client **RTSP DESCRIBE** request AND
- [S5] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- [S6] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND
- There is a Device response on the **GetStreamUri** request for Media2 Service in Test Procedure fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S10] It received before the Client **RTSP DESCRIBE** request AND
  - [S11] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S12] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S13] It invoked after the Client **RTSP SETUP** request AND
  - [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S16] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S17] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S18] It invoked after the Client **RTSP PLAY** request AND

- [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND

- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:

  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.14 H265 Video Streaming Using Media2 Test Cases

5.14.1 Feature Level Normative Reference:

Validated Feature: H265 Video Streaming Using Media2 (Media2_VideoStreaming_H265)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and H265 (Media2 Service) are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.14.2 Expected Scenarios Under Test:

1. Client connects to Device to setup and control of H265 video streaming.
2. Client is considered as supporting H265 Video Streaming if the following conditions are met:
   - Client is able to receive a stream using Media2 and decode H.265 video using the selected Media Profile over RTSP.
3. Client is considered as NOT supporting H265 Video Streaming if ANY of the following is TRUE:
   - Client is unable to initiate and retrieve video stream using Media2 with H.265 encoding type.

5.14.3 H265 VIDEO STREAMING USING MEDIA2

Test Label: Video Streaming - H265

Test Case ID: MEDIA2_VIDEOSTREAMING_H265-1

Feature Under Test: H265 Video Streaming Using Media2 (Media2_VideoStreaming_H265_Media2_H265VideoStreaming)

Test Purpose: To verify that the Client is able to initiate and retrieve a video stream with H265 encoding type.

Pre-Requisite:

- Device supports H265 encoding for Video Streaming for Media2 Service (Media2_H265).
- The Network Trace Capture files contains at least one Conversation between Client and Device with Video Streaming of H265 encoding type.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports H265 encoding for Video Streaming using Media2.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile that contains Video Source Configuration and Video Encoder Configuration with H265 Encoding value. GetStreamUri request is set for RtspUnicast OR RtspMulticast OR RTSP OR RtspOverHttp transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "video" and with encoding name "H265".
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for H265 video streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "video" (m=video) with sessions attribute "rtpmap" with encoding name "H265" (see [RFC 3984], item 8.2.1. Mapping of MIME Parameters to SDP) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S3] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S4] It invoked after the Client **RTSP DESCRIBE** request AND
  - [S5] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S6] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND
- There is a Device response on the **GetStreamUri** request for Media2 Service in Test Procedure fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S10] It received before the Client **RTSP DESCRIBE** request AND
  - [S11] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S12] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND

- [S13] It invoked after the Client **RTSP SETUP** request AND
- [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S16] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S17] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S18] It invoked after the Client **RTSP PLAY** request AND
  - [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

5.15 Video Encoder Configuration Using Media2 Test Cases

5.15.1 Feature Level Normative Reference:

Validated Feature: Video Encoder Configuration Using Media2 (Media2_VideoEncoderConfiguration)

Check Condition based on Device Features: Video (Media2 Service) is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.15.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Video Encoder Configuration.
2. Client is considered as supporting Video Encoder Configuration if the following conditions are met:
   - Client is able to retrieve video encoder configuratoin options using **GetVideoEncoderConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify video encoder configuratoin using **SetVideoEncoderConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Video Encoder Configuration if ANY of the following is TRUE:
   - No valid response to **GetVideoEncoderConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetVideoEncoderConfiguration** request (Media2 Service) OR

5.15.3 GET VIDEO ENCODER CONFIGURATION OPTIONS USING MEDIA2

Test Label: Video Encoder Configuration - Get Video Encoder Configuration Options

Test Case ID: MEDIA2_VIDEOENCODERCONFIGURATION-1

Feature Under Test: Get Video Encoder Configuration Options Using Media2 (Media2_VideoEncoderConfiguration_Media2_GetVideoEncoderConfigurationOptions)

Test Purpose: To verify that video encoder configuration options provided by Device is received by Client using the **GetVideoEncoderConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoEncoderConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoEncoderConfigurationOptions** request message to retrieve a video encoder configuration options from the Device.

2. Device responds with code HTTP 200 OK and **GetVideoEncoderConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetVideoEncoderConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoEncoderConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoEncoderConfigurationOptions** AND
- Device response on the **GetVideoEncoderConfigurationOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetVideoEncoderConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

5.15.4 SET VIDEO ENCODER CONFIGURATION USING MEDIA2

Test Label: Video Encoder Configuration - Set Video Encoder Configuration

Test Case ID: MEDIA2_VIDEOENCODERCONFIGURATION-2

Feature Under Test: Set Video Encoder Configuration Using Media2 (Media2_VideoEncoderConfiguration_Media2_SetVideoEncoderConfigurations)

Test Purpose: To verify that Client is able to change video encoder configuration provided by Device using the **SetVideoEncoderConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetVideoEncoderConfiguration** operation for Media2 Service present.

- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetVideoEncoderConfiguration** request message to change a video encoder configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetVideoEncoderConfigurationResponse** message.

Test Result:

PASS -

- Client **SetVideoEncoderConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetVideoEncoderConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetVideoEncoderConfiguration** AND
- Device response on the **SetVideoEncoderConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetVideoEncoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.16 Get Imaging Settings Test Cases

5.16.1 Feature Level Requirement:

Validated Feature: Get Imaging Settings (GetImagingSettings)

Check Condition based on Device Features: Imaging Service is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.16.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a current imaging settings.
2. Client is considered as supporting Get Imaging Settings if the following conditions are met:
   - Client is able to retrieve a current imaging settings using **GetImagingSettings** operation.
3. Client is considered as NOT supporting Get Imaging Settings if ANY of the following is TRUE:
   - No valid responses for **GetImagingSettings** request.

5.16.3 GET IMAGING SETTINGS

Test Label: Get Imaging Settings - Get Imaging Settings

Test Case ID: GETIMAGINGSETTINGS-1

Feature Under Test: Get Imaging Settings (GetImagingSettings_GetImgSettings)

Test Purpose: To verify that imaging settings for Device is received by Client using the **GetImagingSettings** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetImagingSettings** operation present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetImagingSettings** request message to retrieve imaging settings for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetImagingSettingsResponse** message.

Test Result:

PASS -

- Client **GetImagingSettings** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetImagingSettings** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:GetImagingSettings** AND

- Device response on the **GetImagingSettings** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **timg:GetImagingSettingsResponse**.

FAIL -

- The Client failed PASS criteria.

5.17 Imaging Settings Configuration Test Cases

5.17.1 Feature Level Requirement:

Validated Feature: Imaging Settings Configuration (SetImagingSettings)

Check Condition based on Device Features: Imaging Service is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.17.2 Expected Scenarios Under Test:

1. Client connects to Device to change imaging settings.
2. Client is considered as supporting Imaging Settings Configuration if the following conditions are met:
   - Client is able to retrieve a imaging options using **GetOptions** operation AND
   - Client is able to change a imaging settings using **SetImagingSettings** operation.
3. Client is considered as NOT supporting Imaging Settings Configuration if ANY of the following is TRUE:
   - No valid responses for **GetOptions** request OR
   - No valid responses for **SetImagingSettings** request OR
   - There is no **GetOptions** request for the same video source token as used in **SetImagingSettings** request.

5.17.3 GET OPTIONS

Test Label: Get Imaging Settings - Get Options

Test Case ID: SETIMAGINGSETTINGS-1

Feature Under Test: Get Options (SetImagingSettings_GetOptions)

Test Purpose: To verify that imaging options for Device is received by Client using the **GetOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetOptions** operation present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetOptions** request message to retrieve imaging options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetOptionsResponse** message.

Test Result:

PASS -

- Client **GetOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:GetOptions** AND
- Device response on the **GetOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **timg:GetOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

5.17.4 SET IMAGING SETTINGS

Test Label: Set Imaging Settings - Set Imaging Settings

Test Case ID: SETIMAGINGSETTINGS-2

Feature Under Test: Set Imaging Settings (SetImagingSettings_SetImagingSettingsRequest)

Test Purpose: To verify that Client is able to change imaging settings on Device using the **SetImagingSettings** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetImagingSettings** operation present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetOptions** request message to retrieve imaging options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetOptionsResponse** message.
3. Client invokes **SetImagingSettings** request message to change imaging settings for specified video source which correspond to the retrieved options on the Device.
4. Device responds with code HTTP 200 OK and **SetImagingSettingsResponse** message.

Test Result:

PASS -

- Client **SetImagingSettings** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetImagingSettings** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:SetImagingSettings** AND
- Device response on the **SetImagingSettings** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **timg:SetImagingSettingsResponse** AND
- There is a Client **GetOptions** request in Test Procedure that fulfills the following requirements:
  - [S4] **timg:VideoSourceToken** element value is equal to **timg:VideoSourceToken** element from the **SetImagingSettings** request AND
  - [S5] It is invoked before the Client **SetImagingSettings** request AND
- Device response on the **GetOptions** request fulfills the following requirements:
  - [S6] It has HTTP 200 response code AND
  - [S7] **soapenv:Body** element has child element **timg:GetOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

5.18 Motion Alarm Test Cases

5.18.1 Feature Level Normative Reference:

Validated Feature: Motion Alarm Event (MotionAlarm)

Check Condition based on Device Features: Motion Alarm is supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.18.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get motion alarm notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Motion Alarm if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to retrieve tns1:VideoSource/MotionAlarm notifications if Device supports Motion Alarm feature.
4. Client is considered as NOT supporting Motion Alarm if ANY of the following is TRUE:
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client is not able to retrieve tns1:VideoSource/MotionAlarm.

5.18.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Test Purpose: To verify that the Client is able to retrieve events using Pull Point.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Pull Point event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes PullMessages command with Timeout and MessageLimit elements.
4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.19 PTZ Pan Tilt Continuous Positioning Test Cases

5.19.1 Feature Level Requirement:

Validated Feature: Continuous Move (PtzPanTiltContinuousPositioning)

Check Condition based on Device Features: PTZ Continuous Pan Tilt movement is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Mandatory

5.19.2 Expected Scenarios Under Test:

1. Client connects to Device to control PTZ Pan Tilt position using continuous move.
2. Client is considered as supporting PTZ Pan Tilt Continuous Positioning if the following conditions are met:
   - Client is able to move PTZ Device using the ContinuousMove operation with specified PanTilt element AND
   - Client is able to stop PTZ Pan Tilt Device movement using the Stop operation OR using ContinuousMove operation with zero values in PanTilt element.
3. Client is considered as NOT supporting PTZ Pan Tilt Continuous Positioning if ANY of the following is TRUE:
   - Client is unable to move a PTZ device using the ContinuousMove operation with specified PanTilt element OR
   - Client is unable to stop PTZ Pan Tilt movement using EITHER Stop operation OR using ContinuousMove operation OR
   - No Valid Device Response to **Stop** request if detected OR
   - No Valid Device Response to **ContinuousMove** request with zero "x" and "y" attributes values in PanTilt element if detected.

5.19.3 PTZ CONTINUOUS MOVE PAN/TILT

Test Label: PTZ Continuous Positioning - ContinuousMove PanTilt

Test Case ID: PTZPANTILTCONTINUOUSPOSITIONING-1

Feature Under Test: Pan Tilt Continuous Move (PtzPanTiltContinuousPositioning_ContinuousMovePanTilt)

Test Purpose: To verify that Client is able to move a PTZ Device using the ContinuousMove operation with specified PanTilt element.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.
- Device supports PTZContinuousPanTilt.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes ContinuousMove request message to start move of PTZ Device using specific value of PanTilt element.
2. Device responds with code HTTP 200 OK and ContinuousMoveResponse message.

Test Result:

PASS -

- Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<ContinuousMove>" tag after the "<Body>" tag AND
  - [S2] "<ContinuousMove>" includes tag: "<ProfileToken>" with non-empty string value of specific token AND
  - [S4] "<Velocity>" includes tag: "<PanTilt>" AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<ContinuousMoveResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.19.4 PTZ PAN TILT STOP

Test Label: PTZ Pan Tilt Continuous Positioning - Stop

Test Case ID: PTZPANTILTCONTINUOUSPOSITIONING-2

Feature Under Test: Stop Pan Tilt Movement (PtzPanTiltContinuousPositioning_PanTiltStop)

Test Purpose: To verify that Client is able to stop a PTZ Pan Tilt Device movement using the Stop operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with Stop operation with skipped PanTilt element or with PanTilt = true present

Test Procedure (expected to be reflected in network trace file):

1. Client invokes Stop request message to stop ongoing pan tilt movements of PTZ Device.
2. Device responds with code HTTP 200 OK and StopResponse message.

Test Result:

PASS -

- Client **Stop** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Stop** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:Stop** AND
  - [S2] **tptz:Stop/tptz:ProfileToken** element has non-empty string value of specific token AND
  - [S3] If **tptz:Stop** contains **tptz:PanTilt** element then **tptz:Stop/tptz:PanTilt** = true AND
- Device response on the **Stop** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:StopResponse**.

FAIL -

- The Client failed PASS criteria.

5.19.5 STOP PAN TILT MOVEMENT USING PTZ CONTINUOUS MOVE

Test Label: PTZ Continuous Positioning - Stop Pan Tilt Movement using ContinuousMove

Test Case ID: PTZPANTILTCONTINUOUSPOSITIONING-3

Feature Under Test: Stop Pan Tilt Movement using Continuous Move (PtzPanTiltContinuousPositioning_PanTiltStopUsingPTZContinuousMove)

Test Purpose: To verify that Client is able to stop a PTZ Pan Tilt Device movement using ContinuousMove operation with zero values in PanTilt element.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes ContinuousMove request message with zero "x" and "y" attributes values in PanTilt element.
2. Device responds with code HTTP 200 OK and ContinuousMoveResponse message.

Test Result:

NOTE: In case Client does not send ContinuousMove request message with zero "x" and "y" attributes values in PanTilt element then the test shall be deemed as "NOT DETECTED".

PASS -

- There is client **ContinuousMove** request messages which corresponds to the following requirements (else skip the check):
  - Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
  - Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
    - [S1] **soapenv:Body** element has child element **tptz:ContinuousMove** AND
    - [S2] **tptz:ContinuousMove/tptz:ProfileToken** element has non-empty string value of specific token AND
    - [S3] **tptz:ContinuousMove/tptz:Velocity** containt tag **tt:PanTilt** AND
    - [S4] **tptz:ContinuousMove/tptz:Velocity/tt:PanTilt/@x** attribute value is equal to 0 AND
    - [S5] **tptz:ContinuousMove/tptz:Velocity/tt:PanTilt/@y** attribute value is equal to 0 AND

- Device response on the **ContinuousMove** request fulfills the following requirements:
  - [S6] It has HTTP 200 response code AND
  - [S7] **soapenv:Body** element has child element **tptz:ContinuousMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.20 PTZ Zoom Continuous Positioning Test Cases

5.20.1 Feature Level Requirement:

Validated Feature: Zoom Continuous Move (PtzZoomContinuousPositioning)

Check Condition based on Device Features: PTZ Continuous Zoom movement is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Mandatory

5.20.2 Expected Scenarios Under Test:

1. Client connects to Device to control PTZ Zoom position using continuous move.
2. Client is considered as supporting PTZ Zoom Continuous Positioning if the following conditions are met:
   - Client is able to move PTZ Device using the ContinuousMove operation with specified Zoom element AND
   - Client is able to stop PTZ Zoom Device movement using the Stop operation OR using ContinuousMove operation with zero values in Zoom element.
3. Client is considered as NOT supporting PTZ Zoom Continuous Positioning if ANY of the following is TRUE:
   - Client is unable to move a PTZ device using the ContinuousMove operation with specified Zoom element OR

- Client is unable to stop PTZ Zoom movement using EITHER Stop operation OR using ContinuousMove operation OR
- No Valid Device Response to **Stop** request if detected OR
- No Valid Device Response to **ContinuousMove** request with zero "x" attributes values in Zoom element if detected.

5.20.3 PTZ CONTINUOUS MOVE ZOOM

Test Label: PTZ Continuous Positioning - ContinuousMove Zoom

Test Case ID: PTZZOOMCONTINUOUSPOSITIONING-1

Feature Under Test: Zoom Continuous Move (PtzZoomContinuousPositioning_ContinuousMoveZoom)

Test Purpose: To verify that Client is able to change zoom of PTZ Device using the ContinuousMove operation with specified Zoom element.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.
- Device supports PTZContinuousZoom.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes ContinuousMove request message to change zoom of PTZ Device using specific value of Zoom element.
2. Device responds with code HTTP 200 OK and ContinuousMoveResponse message.

Test Result:

PASS -

- Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<ContinuousMove>" tag after the "<Body>" tag AND
  - [S2] "<ContinuousMove>" includes tag: "<ProfileToken>" with non-empty string value of specific token AND

- [S4] "<Velocity>" includes tag: "<Zoom>" AND
- [S6] Device response contains "HTTP/* 200 OK" AND
- [S7] Device response contains "<ContinuousMoveResponse>" tag.

FAIL -

- The Client failed PASS criteria.

5.20.4 PTZ ZOOM STOP

Test Label: PTZ Zoom Continuous Positioning - Stop

Test Case ID: PTZZOOMCONTINUOUSPOSITIONING-2

Feature Under Test: Stop Zoom Movement (PtzZoomContinuousPositioning_ZoomStop)

Test Purpose: To verify that Client is able to stop a PTZ Zoom Device movement using the Stop operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with Stop operation with skipped Zoom element or with Zoom = true present

Test Procedure (expected to be reflected in network trace file):

1. Client invokes Stop request message to stop ongoing zoom movements of PTZ Device.
2. Device responds with code HTTP 200 OK and StopResponse message.

Test Result:

PASS -

- Client **Stop** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Stop** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:Stop** AND
  - [S2] **tptz:Stop/tptz:ProfileToken** element has non-empty string value of specific token AND
  - [S3] If **tptz:Stop** contains **tptz:Zoom** element then **tptz:Stop/tptz:Zoom** = true AND

- Device response on the **Stop** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:StopResponse**.

FAIL -

- The Client failed PASS criteria.

5.20.5 STOP ZOOM MOVEMENT USING PTZ CONTINUOUS MOVE

Test Label: PTZ Continuous Positioning - Stop Zoom Movement using ContinuousMove

Test Case ID: PTZZOOMCONTINUOUSPOSITIONING-3

Feature Under Test: Stop Zoom Movement using Continuous Move (PtzZoomContinuousPositioning_ZoomStopUsingPTZContinuousMove)

Test Purpose: To verify that Client is able to stop a PTZ Zoom Device movement using ContinuousMove operation with zero values in Zoom element.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes ContinuousMove request message with zero "x" attribute value in Zoom element.
2. Device responds with code HTTP 200 OK and ContinuousMoveResponse message.

Test Result:

NOTE: In case Client does not send ContinuousMove request message with zero "x" attribute value in Zoom element if device supports PTZContinuousZoom then the test shall be deemed as "NOT DETECTED".

PASS -

- There is client **ContinuousMove** request messages which corresponds to the following requirements (else skip the check):

- Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:ContinuousMove** AND
  - [S2] **tptz:ContinuousMove/tptz:ProfileToken** element has non-empty string value of specific token AND
  - [S3] **tptz:ContinuousMove/tptz:Velocity** containt tag **tt:Zoom** AND
  - [S4] **tptz:ContinuousMove/tptz:Velocity/tt:Zoom/@x** attribute value is equal to 0.
- Device response on the **ContinuousMove** request fulfills the following requirements:
  - [S5] It has HTTP 200 response code AND
  - [S6] **soapenv:Body** element has child element **tptz:ContinuousMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.21 PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees Test Cases

5.21.1 Feature Level Requirement:

Validated Feature: Absolute Positioning using Media2 - Spherical Position Space Degrees Media2_PanTiltSpaces_SphericalPositionSpaceDegrees

Check Condition based on Device Features: Profile T, PTZ Absolute Move and PTZ Spherical Coordinate Spaces are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.21.2 Expected Scenarios Under Test:

1. Client connects to Device control the position using absolute positioning with SphericalPositionSpaceDegrees PTZ space.

2. Client is considered as supporting PTZ Using Media2 Absolute Positioning with Spherical Position Space Degrees if the following conditions are met:

   - Client is able to move PTZ Device using the **AbsoluteMove** operation with specified PanTilt element EITHER using space attribute in PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** value OR using Media Profile with PTZConfiguration with DefaultAbsolutePantTiltPositionSpace value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees**

3. Client is considered as NOT supporting PTZ Using Media2 Absolute Positioning with Spherical Position Space Degrees if ANY of the following is TRUE:

   - No valid response to **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** PTZ space if detected AND
   - No valid response to **AbsoluteMove** request with skipped PTZ space attribute for **AbsoluteMove** operations which use media profile with DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** if detected OR
   - No **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** PTZ space is detected AND no **AbsoluteMove** request with skipped PTZ space attribute which use media profile with DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** is detected

5.21.3 PTZ ABSOLUTE MOVE PAN/TILT SPHERICAL POSITION SPACE DEGREES

Test Label: PTZ Using Media2 Absolute Positioning - AbsoluteMove PanTilt with SphericalPositionSpace

Test Case ID: MEDIA2_PANTILTSPACES_SPHERICALPOSITIONSPACEDEGREES-1

Feature Under Test: Absolute PanTilt Move with SphericalPositionSpace (Media2_PanTiltSpaces_SphericalPositionSpaceDegrees_PTZAbsolutePositioningPanTiltSphericalPositionSpace

Test Purpose: To verify that Client is able to move a PTZ Device using the AbsoluteMove operation with specified PanTilt element using **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** PTZ space.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with AbsoluteMove operation present.
- Device supports PTZ Service.
- Device supports Absolute Pan/Tilt movement (PTZAbsolutePanTilt).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **AbsoluteMove** request message to move of PTZ Device using specific value of PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** space attribute value OR
2. Client find or configure media profile to contain PTZConfiguration with DefaultAbsolutePantTiltPositionSpace with value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** and invokes **AbsoluteMove** request message using specific value of PanTilt element with no space attribute.
3. Device responds with code HTTP 200 OK and **AbsoluteMoveResponse** message.

Test Result:

- Client **AbsoluteMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AbsoluteMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:AbsoluteMove** AND
  - [S2] It contains **tptz:Position/tt:PanTilt** element AND
  - [S3] If it contains **tptz:Position/tt:PanTilt/@space** attribute, THEN **tptz:Position/tt:PanTilt/@space** element value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees** AND
  - [S4] ELSE PTZConfiguration that corresponding to media profile used in **AbsoluteMove** request (PTZ Move operation) has DefaultAbsolutePantTiltPositionSpace (Default space element name to get) value is equal to http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpaceDegrees (see Annex A.1 HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove)
- Device response on the **AbsoluteMove** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND

- [S5] **soapenv:Body** element has child element **tptz:AbsoluteMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.22 PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space Test Cases

5.22.1 Feature Level Requirement:

Validated Feature: Absolute Move Using Media2 - Pan Tilt Position Generic Space (Media2_PanTiltSpaces_PositionGenericSpace)

Check Condition based on Device Features: PTZ Absolute Pan/Tilt Move and PTZ Generic Coordinate Spaces are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.22.2 Expected Scenarios Under Test:

1. Client connects to Device control the position using absolute positioning with Pan/Tilt PositionGenericSpace PTZ space.
2. Client is considered as supporting PTZ Using Media2 Absolute Positioning with Pan/Tilt PositionGenericSpace PTZ space if the following conditions are met:
   - Client is able to move PTZ Device using the **AbsoluteMove** operation with specified PanTilt element EITHER using space attribute in PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** value OR using Media Profile with PTZConfiguration with DefaultAbsolutePantTiltPositionSpace value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace**
3. Client is considered as NOT supporting PTZ Using Media2 Absolute Positioning with Pan/Tilt PositionGenericSpace PTZ space if ANY of the following is TRUE:
   - No valid response to **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** PTZ space if detected AND
   - No valid response to **AbsoluteMove** request with skipped PTZ space attribute for **AbsoluteMove** operations which use media profile with

DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** if detected OR

- No **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** PTZ space is detected AND no **AbsoluteMove** request with skipped PTZ space attribute which use media profile with DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** is detected

5.22.3 PTZ ABSOLUTE MOVE PAN/TILT POSITION GENERIC SPACE

Test Label: PTZ Using Media2 Absolute Positioning - AbsoluteMove PanTilt with PositionGeneric Space

Test Case ID: MEDIA2_PANTILTSPACES_POSITIONGENERICSPACE-1

Feature Under Test: Absolute Move PanTilt with PositionGeneric Space (Media2_PanTiltSpaces_PositionGenericSpace_PTZAbsolutePositioningPanTiltPositionGenericSpace)

Test Purpose: To verify that Client is able to move a PTZ Device using the AbsoluteMove operation with specified PanTilt element using **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** PTZ space.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with AbsoluteMove operation present.
- Device supports PTZ Service.
- Device supports Absolute Pan/Tilt movement (PTZAbsolutePanTilt).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **AbsoluteMove** request message to move of PTZ Device using specific value of PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** space attribute value OR
2. Client find or configure media profile to contain PTZConfiguration with DefaultAbsolutePantTiltPositionSpace with value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** and invokes **AbsoluteMove** request message using specific value of PanTilt element with no space attribute.

3. Device responds with code HTTP 200 OK and **AbsoluteMoveResponse** message.

Test Result:

- Client **AbsoluteMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AbsoluteMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:AbsoluteMove** AND
  - [S2] It contains **tptz:Position/tt:PanTilt** element AND
  - [S3] If it contains **tptz:Position/tt:PanTilt/@space** attribute, THEN **tptz:Position/tt:PanTilt/@space** element value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace** AND
  - [S4] ELSE PTZConfiguration that corresponding to media profile used in **AbsoluteMove** request (PTZ Move operation) has DefaultAbsolutePantTiltPositionSpace (Default space element name to get) value is equal to http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace (see Annex A.1 HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove)
- Device response on the **AbsoluteMove** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:AbsoluteMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.23 PTZ Using Media2 Absolute Positioning - Zoom Position Generic Space Test Cases

5.23.1 Feature Level Requirement:

Validated Feature: Absolute Positioning Using Media2 - Zoom Position Generic Space (Media2_ZoomSpaces_PositionGenericSpace)

Check Condition based on Device Features: PTZ Absolute Zoom Move and PTZ Generic Coordinate Spaces are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.23.2 Expected Scenarios Under Test:

1. Client connects to Device control the position using absolute positioning with Zoom PositionGenericSpace PTZ space.

2. Client is considered as supporting PTZ Using Media2 Absolute Positioning with Zoom PositionGenericSpace PTZ space if the following conditions are met:

   - Client is able to move PTZ Device using the **AbsoluteMove** operation with specified Zoom element EITHER using space attribute in Zoom element with **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** value OR using Media Profile with PTZConfiguration with DefaultAbsolutePantTiltPositionSpace value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace**

3. Client is considered as NOT supporting PTZ Using Media2 Absolute Positioning with Zoom PositionGenericSpace PTZ space if ANY of the following is TRUE:

   - No valid response to **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** PTZ space if detected AND

   - No valid response to **AbsoluteMove** request with skipped PTZ space attribute for **AbsoluteMove** operations which use media profile with DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** if detected OR

   - No **AbsoluteMove** request with **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** PTZ space is detected AND no **AbsoluteMove** request with skipped PTZ space attribute which use media profile with DefaultAbsolutePantTiltPositionSpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** is detected

5.23.3 PTZ ABSOLUTE MOVE ZOOM POSITION GENERIC SPACE

Test Label: PTZ Using Media2 Absolute Positioning - AbsoluteMove Zoom with PositionGeneric Space

Test Case ID: MEDIA2_ZOOMSPACES_POSITIONGENERICSPACE-1

Feature Under Test: Absolute Positioning Using Media2 - AbsoluteMove Zoom with PositionGeneric Space (Media2_ZoomSpaces_PositionGenericSpace_PTZAbsolutePositioningZoomPositionGenericSpace)

Test Purpose: To verify that Client is able to move a PTZ Device using the AbsoluteMove operation with specified Zoom element using **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** PTZ space.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with AbsoluteMove operation present.
- Device supports PTZ Service.
- Device supports Absolute Zoom movement (PTZAbsoluteZoom).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **AbsoluteMove** request message to move of PTZ Device using specific value of Zoom element with **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** space attribute value OR
2. Client find or configure media profile to contain PTZConfiguration with DefaultAbsolutePantTiltPositionSpace with value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** and invokes **AbsoluteMove** request message using specific value of Zoom element with no space attribute.
3. Device responds with code HTTP 200 OK and **AbsoluteMoveResponse** message.

Test Result:

- Client **AbsoluteMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AbsoluteMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:AbsoluteMove** AND
  - [S2] It contains **tptz:Position/tt:Zoom** element AND
  - [S3] If it contains **tptz:Position/tt:Zoom/@space** attribute, THEN **tptz:Position/tt:Zoom/@space** element value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace** AND
  - [S4] ELSE PTZConfiguration that corresponding to media profile used in **AbsoluteMove** request (PTZ Move operation) has DefaultAbsolutePantTiltPositionSpace

(Default space element name to get) value is equal to http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace (see Annex A.1 HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove)

- Device response on the **AbsoluteMove** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:AbsoluteMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.24 PTZ Using Media2 Pan Tilt Continuous Positioning Test Cases

5.24.1 Feature Level Requirement:

Validated Feature: PTZ Using Media2 Pan Tilt Continuous Positioning (Media2_PanTiltSpaces_VelocityGenericSpace)

Check Condition based on Device Features: PTZ Continuous PanTilt and Media2 Service are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.24.2 Expected Scenarios Under Test:

1. Client connects to Device control the pan tilt position using absolute positioning with VelocityGenericSpace PTZ spaces.
2. Client is considered as supporting PTZ Using Media2 Pan Tilt Continuous Positioning if the following conditions are met:
   - Client is able to move PTZ Device using the **ContinuousMove** operation with specified PanTilt element EITHER using space attribute in PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** value OR using Media Profile with PTZConfiguration with DefaultContinuousPanTiltVelocitySpace value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** AND

3. Client is considered as NOT supporting PTZ Using Media2 Pan Tilt Continuous Positioning if ANY of the following is TRUE:

   - No valid response to **ContinuousMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** PTZ space if detected AND
   - No valid response to **ContinuousMove** request with skipped PTZ space attribute for **ContinuousMove** operations which use media profile with DefaultContinuousPanTiltVelocitySpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** if detected OR
   - No **ContinuousMove** request with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** PTZ space is detected AND no **ContinuousMove** request with skipped PTZ space attribute which use media profile with DefaultContinuousPanTiltVelocitySpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** is detected AND

5.24.3 PTZ CONTINUOUS MOVE PAN/TILT VELOCITY GENERIC SPACE

Test Label: PTZ Using Media2 Continuous Positioning - ContinuousMove PanTilt

Test Case ID: MEDIA2_PANTILTSPACES_VELOCITYGENERICSPACE-1

Feature Under Test: Pan Tilt Continuous Move Using Media2 - Velocity Generic Space (Media2_PanTiltSpaces_VelocityGenericSpace_PTZContinuousPositioningPanTiltVelocityGenericSpace)

Test Purpose: To verify that Client is able to move a PTZ Device using the ContinuousMove operation with specified PanTilt element using **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** PTZ space.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.
- Device supports PTZ Service.
- Device supports Continuous Pan/Tilt movement (PTZContinuousPanTilt).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **ContinuousMove** request message to move of PTZ Device using specific value of PanTilt element with **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** space attribute value OR

2. Client find or configure media profile to contain PTZConfiguration with DefaultContinuousPanTiltVelocitySpace with value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** and invokes **ContinuousMove** request message using specific value of PanTilt element with no space attribute.
3. Device responds with code HTTP 200 OK and **ContinuousMoveResponse** message.

Test Result:

- Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:ContinuousMove** AND
  - [S2] It contains **tptz:Velocity/tt:PanTilt** element AND
  - [S3] If it contains **tptz:Velocity/tt:PanTilt/@space** attribute, THEN **tptz:Velocity/tt:PanTilt/@space** element value is equal to **http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace** AND
  - [S4] ELSE PTZConfiguration that corresponding to media profile used in **ContinuousMove** request (PTZ Move operation) has DefaultContinuousPanTiltVelocitySpace (Default space element name to get) value is equal to http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace (see Annex A.1 HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove)
- Device response on the **ContinuousMove** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:ContinuousMoveResponse**.

FAIL -

- The Client failed PASS criteria.

5.25 PTZ Using Media2 Zoom Continuous Positioning Test Cases

5.25.1 Feature Level Requirement:

Validated Feature: PTZ Using Media2 Zoom Continuous Positioning (Media2_ZoomSpaces_VelocityGenericSpace)

Check Condition based on Device Features: PTZ Continuous Zoom and Medis2 Service are supported by Device.

Required Number of Devices: 3

Profile T Requirement: Mandatory

5.25.2 Expected Scenarios Under Test:

1. Client connects to Device control the zoom position using absolute positioning with VelocityGenericSpace PTZ spaces.
2. Client is considered as supporting PTZ Using Media2 Zoom Continuous Positioning if the following conditions are met:
   - Client is able to move PTZ Device using the **ContinuousMove** operation with specified Zoom element EITHER using space attribute in Zoom element with **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** value OR using Media Profile with PTZConfiguration with DefaultContinuousZoomVelocitySpace value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** AND
3. Client is considered as NOT supporting PTZ Using Media2 Zoom Continuous Positioning if ANY of the following is TRUE:
   - No valid response to **ContinuousMove** request with **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** PTZ space if detected AND
   - No valid response to **ContinuousMove** request with skipped PTZ space attribute for **ContinuousMove** operations which use media profile with DefaultContinuousZoomVelocitySpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** if detected OR
   - No **ContinuousMove** request with **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** PTZ space is detected AND no **ContinuousMove** request with skipped PTZ space attribute which use media profile with DefaultContinuousZoomVelocitySpace value in PTZConfiguration is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** is detected AND

5.25.3 PTZ CONTINUOUS MOVE ZOOM VELOCITY GENERIC SPACE

Test Label: PTZ Using Media2 Continuous Positioning - ContinuousMove Zoom

Test Case ID: MEDIA2_ZOOMSPACES_VELOCITYGENERICSPACE-1

Feature Under Test: Zoom Continuous Move Using Media2 - Velocity Generic Space (Media2_ZoomSpaces_VelocityGenericSpace_PTZContinuousPositioningZoomVelocityGenericSpace)

Test Purpose: To verify that Client is able to move a PTZ Device using the ContinuousMove operation with specified Zoom element using **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** PTZ space.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with ContinuousMove operation present.
- Device supports PTZ Service.
- Device supports Continuous Zoom movement (PTZContinuousZoom).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **ContinuousMove** request message to move of PTZ Device using specific value of Zoom element with **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** space attribute value OR
2. Client find or configure media profile to contain PTZConfiguration with DefaultContinuousZoomVelocitySpace with value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** and invokes **ContinuousMove** request message using specific value of Zoom element with no space attribute.
3. Device responds with code HTTP 200 OK and **ContinuousMoveResponse** message.

Test Result:

- Client **ContinuousMove** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **ContinuousMove** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:ContinuousMove** AND
  - [S2] It contains **tptz:Velocity/tt:Zoom** element AND
  - [S3] If it contains **tptz:Velocity/tt:Zoom/@space** attribute, THEN **tptz:Velocity/tt:Zoom/@space** element value is equal to **http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace** AND
  - [S4] ELSE PTZConfiguration that corresponding to media profile used in **ContinuousMove** request (PTZ Move operation) has DefaultContinuousZoomVelocitySpace

(Default space element name to get) value is equal to http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace (see Annex A.1 HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove)

- Device response on the **ContinuousMove** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tptz:ContinuousMoveResponse**.

FAIL -

- The Client failed PASS criteria.

6 Test Cases for Profile Conditional Features

6.1 System Test Cases

6.1.1 Feature Level Requirement:

Validated Feature: System (System)

Check Condition based on Device Features: None

Required Number of Devices: 3

Profile A Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile S Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

Profile M Requirement: Conditional

6.1.2 Expected Scenarios Under Test:

1. Client connects to Device to get information, such as manufacturer, model, firmware version and etc.
2. Client is considered as supporting System if the following conditions are met:
   - Client is able to list Device information using the GetDeviceInformation operation.
3. Client is considered as NOT supporting System if ANY of the following is TRUE:
   - No Valid Device Response to GetDeviceInformation request.

6.1.3 GET DEVICE INFORMATION

Test Label: System - Get Device Information

Test Case ID: SYSTEM-1

Feature Under Test: Get Device Information (System_GetDeviceInformation)

Test Purpose: To verify that Client is able to list Device information using the GetDeviceInformation operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetDeviceInformation operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetDeviceInformation request message to list Device information.
2. Device responds with code HTTP 200 OK and GetDeviceInformationResponse message.

Test Result:

PASS -

- Client **GetDeviceInformation** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetDeviceInformation** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetDeviceInformation>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetDeviceInformationResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.2 User Handling Test Cases

6.2.1 Feature Level Requirement:

Validated Feature: User Handling (UserHandling)

Check Condition based on Device Features: User Configuration

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile S Requirement: Conditional

Profile C Requirement: Conditional

Profile G Requirement: Conditional

Profile T Requirement: Conditional

Profile D Requirement: Conditional

6.2.2 Expected Scenarios Under Test:

1. Client connects to Device to create, list, modify and delete users.
2. Client is considered as supporting User Handling if the following conditions are met:
   - Client is able to create users on Device using the CreateUsers operation AND
   - Client is able to list existing users of Device using the GetUsers operation AND
   - Client is able to modify users on Device using the SetUser operation AND
   - Client is able to delete users from Device using the DeleteUsers operation.
3. Client is considered as NOT supporting System if ANY of the following is TRUE:
   - No Valid Device Response to CreateUsers request (except SOAP fault: **soapenv:Receiver/ter:Action/ter:TooManyUsers**) OR
   - No Valid Device Response to GetUsers request OR
   - No Valid Device Response to SetUser request (except SOAP fault: **soapenv:Sender/ter:InvalidArgVal/ter:FixedUser**) OR
   - No Valid Device Response to DeleteUsers request (except SOAP fault: **soapenv:Sender/ter:InvalidArgVal/ter:FixedUser**).

6.2.3 CREATE USERS

Test Label: User Handling - CreateUsers

Test Case ID: USERHANDLING-1

Feature Under Test: Create Users (UserHandling_CreateUsers)

Test Purpose: To verify that Client is able to create users on Device using the CreateUsers operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with CreateUsers operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreateUsers request message to create new users and corresponding credentials on Device.
2. Device responds with code HTTP 200 OK and CreateUsersResponse message.

Test Result:

PASS -

- Client **CreateUsers** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateUsers** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreateUsers>" tag after the "<Body>" tag AND
  - [S2] "<CreateUsers>" includes tag: "<User>" AND
  - [S3] "<User>" includes tag: "<Username>" with non-empty string value AND
  - [S4] "<User>" includes tag: "<Password>" with non-empty string value AND
  - [S5] If Device response contains "HTTP/* 200 OK" THEN it contains "<CreateUsersResponse>" tag, ELSE it contains **soapenv:Fault** with **soapenv:Receiver/ter:Action/ter:TooManyUsers** fault code.

FAIL -

- The Client failed PASS criteria.

6.2.4 GET USERS

Test Label: User Handling - GetUsers

Test Case ID: USERHANDLING-2

Feature Under Test: Get Users (UserHandling_GetUsers)

Test Purpose: To verify that Client is able to list existing users of Device using the GetUsers operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetUsers operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetUsers request message to list registered users and their user levels.
2. Device responds with code HTTP 200 OK and GetUsersResponse message.

Test Result:

PASS -

- Client **GetUsers** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetUsers** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetUsers>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetUsersResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.2.5 SET USER

Test Label: User Handling - SetUser

Test Case ID: USERHANDLING-3

Feature Under Test: Set User (UserHandling_SetUser)

Test Purpose: To verify that Client is able to modify users on Device using the SetUser operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with SetUser operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetUser request message to update the authentication settings on Device.

2. Device responds with code HTTP 200 OK and SetUserResponse message.

Test Result:

PASS -

- Client **SetUser** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetUser** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetUser>" tag after the "<Body>" tag AND
  - [S2] "<SetUser>" includes tag: "<User>" AND
  - [S3] "<User>" includes tag: "<Username>" with non-empty string value AND
  - [S4] If Device response contains "HTTP/* 200 OK" THEN it contains "<SetUserResponse>" tag, ELSE it contains **soapenv:Fault** with **soapenv:Sender/ter:InvalidArgVal/ter:FixedUser** fault code.

FAIL -

- The Client failed PASS criteria.

6.2.6 DELETE USERS

Test Label: User Handling - DeleteUsers

Test Case ID: USERHANDLING-4

Feature Under Test: Delete Users (UserHandling_DeleteUsers)

Test Purpose: To verify that Client is able to delete users from Device using the DeleteUsers operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with DeleteUsers operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes DeleteUsers request message to delete specific users from Device.
2. Device responds with code HTTP 200 OK and DeleteUsersResponse message.

Test Result:

PASS -

- Client **DeleteUsers** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **DeleteUsers** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<DeleteUsers>" tag after the "<Body>" tag AND
  - [S2] "<DeleteUsers>" includes tag: "<Username>" with non-empty string value AND
  - [S3] If Device response contains "HTTP/* 200 OK" THEN it contains "<DeleteUsersResponse>" tag, ELSE it contains **soapenv:Fault** with **soapenv:Sender/ter:InvalidArgVal/ter:FixedUser** fault code.

FAIL -

- The Client failed PASS criteria.

6.3 NTP Test Cases

6.3.1 Feature Level Requirement:

Validated Feature: NTP (NTP)

Check Condition based on Device Features: NTP is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Conditional

6.3.2 Expected Scenarios Under Test:

1. Client connects to Device to configure synchronization of time using NTP servers on Device.
2. Client is considered as supporting NTP if the following conditions are met:
   - Client is able to get the NTP settings from Device using the GetNTP operation AND
   - Client is able to set the NTP settings on Device using the SetNTP operation.
3. Client is considered as NOT supporting NTP if ANY of the following is TRUE:
   - No Valid Device Response to GetNTP request OR

- No Valid Device Response to SetNTP request.

6.3.3 GET NTP

Test Label: NTP - GetNTP

Test Case ID: NTP-1

Feature Under Test: Get NTP (NTP_GetNTP)

Test Purpose: To verify that Client is able to get the NTP settings from Device using the GetNTP operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with GetNTP operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetNTP request message to get current settings of NTP servers on Device.
2. Device responds with code HTTP 200 OK and GetNTPResponse message.

Test Result:

PASS -

- Client **GetNTP** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetNTP** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNTP>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetNTPResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.3.4 SET NTP

Test Label: NTP - SetNTP

Test Case ID: NTP-2

Feature Under Test: Set NTP (NTP_SetNTP)

Test Purpose: To verify that Client is able to set the NTP settings on Device using the SetNTP operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with SetNTP operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetNTP request message to set the NTP servers settings on Device.
2. Device responds with code HTTP 200 OK and SetNTPResponse message.

Test Result:

PASS -

- Client **SetNTP** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetNTP** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<SetNTP>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<SetNTPResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.4 Auxiliary Commands (Device Management Service) Test Cases

6.4.1 Feature Level Requirement:

Validated Feature: Auxiliary Commands (AuxiliaryCommands)

Check Condition based on Device Features: None (ONVIF Profile T Simulator is used as device).

Required Number of Devices: 1

Profile S Requirement: None

Profile G Requirement: None

Profile A Requirement: None

Profile C Requirement: None

Profile T Requirement: Conditional

6.4.2 Expected Scenarios Under Test:

1. Client connects to ONVIF Profile T Simulator to manage auxiliary commands supported by the Device.
2. Client is considered as supporting Auxiliary Commands (Device Management Service) if the following conditions are met:
   - ONVIF Profile T Simulator detects **SendAuxiliaryCommand** request with at least one of the following auxiliary commands:
     - tt:Wiper|On
     - tt:Wiper|Off
     - tt:Washer|On
     - tt:Washer|Off
     - tt:WashingProcedure|On
     - tt:WashingProcedure|Off
     - tt:IRLamp|On
     - tt:IRLamp|Off
     - tt:IRLamp|Auto
3. Client is considered as NOT supporting Auxiliary Commands (Device Management Service) if ANY of the following is TRUE:
   - • ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:Wiper|On auxiliary command AND
     - ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:Wiper|Off auxiliary command AND

- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:Washer|On auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:Washer|Off auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:WashingProcedure|On auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:WashingProcedure|Off auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:IRLamp|On auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:IRLamp|Off auxiliary command AND
- ONVIF Profile T Simulator does not detect **SendAuxiliaryCommand** request with tt:IRLamp|Auto auxiliary command.

Recommendation: Clients should support all auxiliary operations listed in the scenario.

6.4.3 WIPER ON

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-1

Feature Under Test: tt:Wiper|On (AuxiliaryCommands_WiperOn)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:Wiper|On"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:Wiper|On"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.4 WIPER OFF

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-2

Feature Under Test: tt:Wiper|Off (AuxiliaryCommands_WiperOff)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:Wiper|Off"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:Wiper|Off"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.5 WASHER ON

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-3

Feature Under Test: tt:Washer|On (AuxiliaryCommands_WasherOn)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:Washer|On"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:Washer|On"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.6 WASHER OFF

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-4

Feature Under Test: tt:Washer|Off (AuxiliaryCommands_WasherOff)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:Washer|Off"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:Washer|Off"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.7 WASHINGPROCEDURE ON

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-5

Feature Under Test: tt:WashingProcedure|On (AuxiliaryCommands_WashingProcedureOn)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:WashingProcedure|On"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:WashingProcedure|On"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.8 WASHINGPROCEDURE OFF

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-6

Feature Under Test: tt:WashingProcedure|Off (AuxiliaryCommands_WashingProcedureOff)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:WashingProcedure|Off"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:WashingProcedure|Off"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.9 IRLAMP ON

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-7

Feature Under Test: tt:IRLamp|On (AuxiliaryCommands_IRLampOn)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:IRLamp|On"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:IRLamp|On"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.10 IRLAMP OFF

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-8

Feature Under Test: tt:IRLamp|Off (AuxiliaryCommands_IRLampOff)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:IRLamp|Off"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:IRLamp|Off"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.4.11 IRLAMP AUTO

Test Label: Auxiliary Commands

Test Case ID: AUXILIARYCOMMANDS-9

Feature Under Test: tt:IRLamp|Auto (AuxiliaryCommands_IRLampAuto)

Test Purpose: To verify that the Client supports the SendAuxiliaryCommand operation.

Pre-Requisite:

- The .osc file of conversation with ONVIF Profile T Simulator is present (please, refer to 'Profile T Simulator' section in Help) .

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SendAuxiliaryCommand** request message with **AuxiliaryCommand** = **"tt:IRLamp|Auto"** to the ONVIF Profile T Simulator (please, refer to 'Profile T Simulator' section in Help).
2. ONVIF Profile T Simulator responds with **SendAuxiliaryCommandResponse** message and 200 OK HTTP code.

Test Result:

PASS -

- Client **SendAuxiliaryCommand** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SendAuxiliaryCommand** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SendAuxiliaryCommand** AND
  - [S2] **tds:SendAuxiliaryCommand/tds:AuxiliaryCommand** is equal to **"tt:IRLamp|Auto"** AND
- ONVIF Profile T Simulator response on the **SendAuxiliaryCommand** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SendAuxiliaryCommandResponse**.

FAIL -

- None.

6.5 Metadata Profile Configuration Using Media2 Test Cases

6.5.1 Feature Level Requirement:

Validated Feature: Metadata Media2 Profile Configuration (Media2_MetadataProfileConfiguration)

Check Condition based on Device Features: Media2 Service is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

Profile M Requirement: Conditional

6.5.2 Expected Scenarios Under Test:

1. Client connects to Device to add compatible Metadata configuration to a Media Profile.
2. Client is considered as supporting Metadata Profile Configuration if the following conditions are met:

- Client is able to retrieve Metadata configurations compatible with media profile using **GetMetadataConfigurations** operation with specified ProfileToken element.
- Client is able to add an Metadata configuration using **AddConfiguration** operation with Type element value is equal to Metadata.

3. Client is considered as NOT supporting Metadata Profile Configuration if ANY of the following is TRUE:
   - No valid responses for **GetMetadataConfigurations** request with ProfileToken element OR
   - No valid responses for **AddConfiguration** request with Type element value is equal to Metadata is detected.

6.5.3 GET METADATA CONFIGURATIONS COMPATIBLE WITH PROFILE USING MEDIA2

Test Label: Get Metadata Configurations

Test Case ID: MEDIA2_METADATAPROFILECONFIGURATION-1

Feature Under Test: Get Metadata Configurations Compatible With Media2 Profile (Media2_MetadataProfileConfiguration_Media2_GetCompatibleMetadataProfileConfigurations)

Test Purpose: To verify that list of metadata configurations compatible with a media profile is received by Client using the **GetMetadataConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetMetadataConfigurations** operation with specified **ProfileToken** element present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMetadataConfigurations** request message with **ProfileToken** element to retrieve a list of metadata configurations compatible with requested media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetMetadataConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetMetadataConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetMetadataConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetMetadataConfigurations** AND
  - [S2] It has **tr2:ProfileToken** element AND
- Device response to the **GetMetadataConfigurations** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:GetMetadataConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.5.4 ADD METADATA CONFIGURATION USING MEDIA2

Test Label: Add Metadata Configuration

Test Case ID: MEDIA2_METADATAPROFILECONFIGURATION-2

Feature Under Test: Add Metadata Configuration To Media2 Profile (Media2_MetadataProfileConfiguration_Media2_AddMetadataProfileConfiguration)

Test Purpose: To verify that Client is able to add an metadata configuration to a media profile using the **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type value is equal to "Metadata" present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMetadataConfigurations** request message with specified **ProfileToken** to retrieve compatible metadata configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetMetadataConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to Metadata and with Configuration token that was recieved in **GetMetadataConfigurationsResponse** message for the same media profile to add an metadata configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "Metadata" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.
- There is Client **GetMetadataConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetMetadataConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetMetadataConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND

- [S9] **soapenv:Body** element has child element **tr2:GetMetadataConfigurationsResponse** AND
- [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to Metadata from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.6 Metadata Configuration Using Media2 Test Cases

6.6.1 Feature Level Normative Reference:

Validated Feature: Metadata Configuration Using Media2 (Media2_MetadataConfiguration)

Check Condition based on Device Features: Media2 Service is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

Profile M Requirement: Mandatory

6.6.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Metadata Configuration.
2. Client is considered as supporting Metadata Configuration if the following conditions are met:
   - Client is able to retrieve metadata configuratoins using **GetMetadataConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve metadata configuratoin options using **GetMetadataConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify metadata configuratoin using **SetMetadataConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Metadata Configuration if ANY of the following is TRUE:

- No valid response to **GetMetadataConfigurations** request (Media2 Service) OR
- No valid response to **GetMetadataConfigurationOptions** request (Media2 Service) OR
- No valid response to **SetMetadataConfiguration** request (Media2 Service) OR

6.6.3 GET METADATA CONFIGURATIONS USING MEDIA2

Test Label: Metadata Configuration - Get Metadata Configurations

Test Case ID: MEDIA2_METADATACONFIGURATION-1

Feature Under Test: Get Metadata Configurations Using Media2 (Media2_MetadataConfiguration_Media2_GetMetadataConfigurations)

Test Purpose: To verify that metadata configuration provided by Device is received by Client using the **GetMetadataConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetMetadataConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMetadataConfigurations** request message to retrieve an metadata configuration or a list of metadata configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetMetadataConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetMetadataConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetMetadataConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetMetadataConfigurations** AND

- Device response on the **GetMetadataConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetMetadataConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.6.4 GET METADATA CONFIGURATION OPTIONS USING MEDIA2

Test Label: Metadata Configuration - Get Metadata Configuration Options

Test Case ID: MEDIA2_METADATACONFIGURATION-2

Feature Under Test: Get Metadata Configuration Options Using Media2 (Media2_MetadataConfiguration_Media2_GetMetadataConfigurationOptions)

Test Purpose: To verify that metadata configuration options provided by Device is received by Client using the **GetMetadataConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetMetadataConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMetadataConfigurationOptions** request message to retrieve an metadata configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetMetadataConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetMetadataConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **GetMetadataConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetMetadataConfigurationOptions** AND
- Device response on the **GetMetadataConfigurationOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetMetadataConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.6.5 SET METADATA CONFIGURATION USING MEDIA2

Test Label: Metadata Configuration - Set Metadata Configuration

Test Case ID: MEDIA2_METADATACONFIGURATION-3

Feature Under Test: Set Metadata Configuration Using Media2 (Media2_MetadataConfiguration_Media2_SetMetadataConfiguration)

Test Purpose: To verify that Client is able to change metadata configuration provided by Device using the **SetMetadataConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetMetadataConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetMetadataConfiguration** request message to change an metadata configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetMetadataConfigurationResponse** message.

Test Result:

PASS -

- Client **SetMetadataConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetMetadataConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetMetadataConfiguration** AND
- Device response on the **SetMetadataConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetMetadataConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

6.7 Metadata Streaming Using Media2 Test Cases

6.7.1 Feature Level Normative Reference:

Validated Feature: Metadata Streaming Using Media2 (Media2_MetadataStreaming)

Check Condition based on Device Features: Media2 Service is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.7.2 Expected Scenarios Under Test:

1. Client connects to Device to receive metadata stream using Media2 Service.
2. Client is considered as supporting Metadata Streaming Using Media2 if the following conditions are met:
   - Client supports Media2_GetProfiles_Media2_GetProfilesRequest feature (please see MEDIA2_GETPROFILES-1 GET PROFILES USING MEDIA2 section) AND

- Client supports Media2_GetStreamURI_Media2_GetStreamURIRequest feature (please see MEDIA2_GETSTREAMURI-1 GET STREAM URI USING MEDIA2 section) AND
- Metadata stream was successfully established by a Client.

3. Client is considered as NOT supporting Metadata Streaming Using Media2 if the following is TRUE:
   - Client does not support Media2_GetProfiles_Media2_GetProfilesRequest feature (please see MEDIA2_GETPROFILES-1 GET PROFILES USING MEDIA2 section) OR
   - Client does not support Media2_GetStreamURI_Media2_GetStreamURIRequest feature (please see MEDIA2_GETSTREAMURI-1 GET STREAM URI USING MEDIA2 section) OR
   - Client is unable to establish metadata stream.

6.7.3 METADATA STREAMING USING MEDIA2

Test Label: Metadata Streaming Using Media2

Test Case ID: MEDIA2_METADATASTREAMING-1

Feature Under Test: Metadata Streaming (Media2_MetadataStreaming_MetadataStreamingUsingMedia2)

Test Purpose: To verify that the Client is able to retrieve the Metadata Streaming.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Metadata Streaming using Media2 Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for Media2 service for media profile that contains Metadata Configuration. GetStreamUri request is set for RtspUnicast OR RtspMulticast OR RTSP OR RtspOverHttp transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "application" and with encoding name "vnd.onvif.metadata" or "vnd.onvif.metadata.gzip" or "vnd.onvif.metadata.exi.onvif" or "vnd.onvif.metadata.exi.ext".

5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for metadata streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RtspOverHttp transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "application" (m=application) with sessions attribute "rtpmap" with encoding name "vnd.onvif.metadata" OR "vnd.onvif.metadata.gzip" OR "vnd.onvif.metadata.exi.onvif" OR "vnd.onvif.metadata.exi.ext" (see ONVIF Streaming Spec) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S3] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S4] It invoked after the Client **RTSP DESCRIBE** request AND
  - [S5] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S6] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND

- There is a Device response on the **GetStreamUri** request invoked for Media2 Service in Test Procedure fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S10] It received before the Client **RTSP DESCRIBE** request AND
  - [S11] It contains **tr2:GetStreamUriResponse\tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S12] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S13] It invoked after the Client **RTSP SETUP** request AND
  - [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S16] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S17] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S18] It invoked after the Client **RTSP PLAY** request AND
  - [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.8 Multicast Streaming Using Media2 Test Cases

6.8.1 Feature Level Requirement:

Validated Feature: Multicast Streaming Using Media2 (Media2_MulticastStreaming)

Check Condition based on Device Features: RTP-Multicast/UDP (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.8.2 Expected Scenarios Under Test:

1. Client connects to Device and initiates Multicast Streaming using RTSP.
2. Client is considered as supporting Multicast Streaming Using Media2 if the following conditions are met:
   - Client is able to start and stop a multicast stream by using RTSP.
3. Client is considered as NOT supporting Multicast Streaming Using Media2 if ANY of the following is TRUE:
   - Client is not able to start and stop a multicast stream by using RTSP.

6.8.3 MULTICAST STREAMING OVER RTSP USING MEDIA2

Test Label: Multicast Streaming Using Media2 - RTSP multicast setup

Test Case ID: MEDIA2_MULTICASTSTREAMING-1

Feature Under Test: Multicast Streaming Using RTSP Using Media2 (Media2_MulticastStreaming_Media2_MulticastRTSP)

Test Purpose: To verify that the Client is able to setup and initiate a multicast stream with RTSP commands for stream control.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with RTSP SETUP request with transport parameter as "RTP/AVP/UDP;multicast" or "RTP/AVP;multicast" and without "onvif-replay" Require header present.

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service with **rt2:Protocol** element with "RtspMulticast" value.
- Device supports RTPMulticastUDP feature.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile with Protocol element with "RtspMulticast" value.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message. message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK.
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with **Transport** tag in RTSP header that contains "RTP/AVP/UDP;multicast" or "RTP/AVP;multicast" to set media session parameters.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

PASS -

- Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S1] It contains **Transport** request header field with value is equal to "RTP/AVP/UDP" OR "RTP/AVP" and with "multicast" parameter value (transport=RTP, profile=AVP, lower-transport=TCP or skipped, parameter=multicast) (see [RFC 2326]) AND
  - [S2] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S3] It has RTSP 200 response code AND
- There is Client **RTSP DESCRIBE** request in Test Procedure that fulfills the following requirements:
  - [S4] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S5] It invoked before the Client **RTSP SETUP** request AND
  - [S6] SDP packet contains media type with Control URL that was used to send **RTSP SETUP** (see [RFC 2326, C.1.1 Control URL]) AND
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND
- There is a Device **GetStreamUri** request in Test Procedure that fulfills the following requirements:
  - [S8] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S9] It invoked before the Client **RTSP DESCRIBE** request AND
  - [S10] **tr2:GetStreamUri/tr2:Protocol** element value is equal to "RtspMulticast"
- Device response on the **GetStreamUri** request fulfills the following requirements:
  - [S11] It has HTTP 200 response code AND
  - [S12] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure that fulfills the following requirements:
  - [S13] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S14] It invoked after the Client **RTSP SETUP** request AND
  - [S15] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S16] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:

- [S17] It has RTSP 200 response code AND

- There is Client **RTSP TEARDOWN** request in Test Procedure that fulfills the following requirements:
  - [S18] It invoked for the same RTSP session as the Client **RTSP SETUP** request AND
  - [S19] It invoked after the Client **RTSP PLAY** request AND
  - [S20] RTSP address that was used to send it is correspond to any media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S21] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.9 Video Profile Configuration Using Media2 Test Cases

6.9.1 Feature Level Requirement:

Validated Feature: Video Profile Configuration Using Media2 (Media2_VideoProfileConfiguration)

Check Condition based on Device Features: Video (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.9.2 Expected Scenarios Under Test:

1. Client connects to Device to add video encoder configuration to a Media Profile.
2. Client is considered as supporting Video Profile Configuration if the following conditions are met:
   - Client is able to add an video encoder configuration using **AddConfiguration** operation with Type element value is equal to VideoEncoder.
3. Client is considered as NOT supporting Video Profile Configuration if ANY of the following is TRUE:

- No valid responses for **AddConfiguration** request with Type element value is equal to VideoEncoder is detected.

6.9.3 ADD VIDEO ENCODER CONFIGURATION USING MEDIA2

Test Label: Add Video Encoder Configuration

Test Case ID: MEDIA2_VIDEOPROFILECONFIGURATION-1

Feature Under Test: Add Video Encoder Configuration To Media2 Profile (Media2_VideoProfileConfiguration_Media2_AddVideoEncoderConfiguration)

Test Purpose: To verify that Client is able to add an video encoder configuration to a media profile using the **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type value is equal to "VideoEncoder" present.
- Device supports Media2 Video feature (Media2_Video).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoEncoderConfigurations** request message with specified **ProfileToken** to retrieve compatible video encoder configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoEncoderConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to VideoEncoder and with Configuration token that was recieved in **GetVideoEncoderConfigurationsResponse** message for the same media profile to add an video encoder configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "VideoEncoder" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.
- There is Client **GetVideoEncoderConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetVideoEncoderConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetVideoEncoderConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tr2:GetVideoEncoderConfigurationsResponse** AND
  - [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to VideoEncoder from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.10 Audio Encoder Configuration Using Media2 Test Cases

6.10.1 Feature Level Normative Reference:

Validated Feature: Audio Encoder Configuration Using Media2 (Media2_AudioEncoderConfiguration)

Check Condition based on Device Features: Audio (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.10.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Audio Encoder Configuration.
2. Client is considered as supporting Audio Encoder Configuration if the following conditions are met:
   - Client is able to retrieve audio encoder configuratoins using **GetAudioEncoderConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve audio encoder configuratoin options using **GetAudioEncoderConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify audio encoder configuratoin using **SetAudioEncoderConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Audio Encoder Configuration if ANY of the following is TRUE:
   - No valid response to **GetAudioEncoderConfigurations** request (Media2 Service) OR
   - No valid response to **GetAudioEncoderConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetAudioEncoderConfiguration** request (Media2 Service) OR

6.10.3 GET AUDIO ENCODER CONFIGURATIONS USING MEDIA2

Test Label: Audio Encoder Configuration - Get Audio Encoder Configurations

Test Case ID: MEDIA2_AUDIOENCODERCONFIGURATION-1

Feature Under Test: Get Audio Encoder Configurations Using Media2 (Media2_AudioEncoderConfiguration_Media2_GetAudioEncoderConfigurations)

Test Purpose: To verify that audio encoder configuration provided by Device is received by Client using the **GetAudioEncoderConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioEncoderConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioEncoderConfigurations** request message to retrieve an audio encoder configuration or a list of audio encoder configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioEncoderConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioEncoderConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioEncoderConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurations** AND
- Device response on the **GetAudioEncoderConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.10.4 GET AUDIO ENCODER CONFIGURATION OPTIONS USING MEDIA2

Test Label: Audio Encoder Configuration - Get Audio Encoder Configuration Options

Test Case ID: MEDIA2_AUDIOENCODERCONFIGURATION-2

Feature Under Test: Get Audio Encoder Configuration Options Using Media2 (Media2_AudioEncoderConfiguration_Media2_GetAudioEncoderConfigurationOptions)

Test Purpose: To verify that audio encoder configuration options provided by Device is received by Client using the **GetAudioEncoderConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioEncoderConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioEncoderConfigurationOptions** request message to retrieve an audio encoder configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioEncoderConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetAudioEncoderConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioEncoderConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurationOptions** AND
- Device response on the **GetAudioEncoderConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.10.5 SET AUDIO ENCODER CONFIGURATION USING MEDIA2

Test Label: Audio Encoder Configuration - Set Audio Encoder Configuration

Test Case ID: MEDIA2_AUDIOENCODERCONFIGURATION-3

Feature Under Test: Set Audio Encoder Configuration Using Media2 (Media2_AudioEncoderConfiguration_Media2_SetAudioEncoderConfigurations)

Test Purpose: To verify that Client is able to change audio encoder configuration provided by Device using the **SetAudioEncoderConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetAudioEncoderConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetAudioEncoderConfiguration** request message to change an audio encoder configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetAudioEncoderConfigurationResponse** message.

Test Result:

PASS -

- Client **SetAudioEncoderConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetAudioEncoderConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetAudioEncoderConfiguration** AND
- Device response on the **SetAudioEncoderConfiguration** request fulfills the following requirements:

- [S2] It has HTTP 200 response code AND
- [S3] **soapenv:Body** element has child element **tr2:SetAudioEncoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

6.11 G.711 Audio Streaming Using Media2 Test Cases

6.11.1 Feature Level Requirement:

Validated Feature: G.711 Audio Streaming Using Media2 (Media2_AudioStreaming_G711)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and Audio G711 (Media2 Service) are supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.11.2 Expected Scenarios Under Test:

1. Client connects to Device to configure a media profile and initiate Audio Streaming with G.711 encoding type.
2. Client is considered as supporting G711 Audio Streaming if the following conditions are met:
   - Client is able to initiate and retrieve audio stream using Media2 with G.711 encoding type.
3. Client is considered as NOT supporting G711 Audio Streaming if ANY of the following is TRUE:
   - Client is unable to initiate and retrieve audio stream using Media2 with G.711 encoding type.

6.11.3 G.711 AUDIO STREAMING USING MEDIA2

Test Label: Audio Streaming using Media2 - G.711

Test Case ID: MEDIA2_AUDIOSTREAMING_G711-1

Feature Under Test: G.711 Audio Streaming Using Media2 (Media2_AudioStreaming_G711_Media2_G711AudioStreaming)

Test Purpose: To verify that the Client is able to initiate and retrieve audio stream with G.711 encoding type.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with Audio Streaming of G.711 encoding type.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports G.711 encoding for Audio streaming using Media2 (Media2_G711).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile that contains Audio Source Configuration and Audio Encoder Configuration with G711 Encoding value. GetStreamUri request is set for RTP-Unicast/UDP OR RTP-Multicast/UDP OR RTP/RTSP/TCP OR RTP-Unicast/RTSP/HTTP/TCP transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "audio" and with encoding name "PCMU" or with payload type number "0".
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to set media session parameters for G711 audio streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] IF SDP packet contains media type "audio" (m=audio) without session attribute "sendonly" (a=sendonly) AND with sessions attribute "rtpmap" THEN encoding name is "PCMU"
  - [S3] ELSE IF SDP packet contains media type "audio" (m=audio) without session attribute "sendonly" (a=sendonly) AND without sessions attribute "rtpmap" THEN payload type number is "0" (see [RFC 3551]) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S4] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S5] It invoked after the Client **RTSP DESCRIBE** request AND
  - [S6] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S7] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S8] It has RTSP 200 response code AND
- There is a Device response on the **GetStreamUri** request to Media2 Service in Test Procedure fulfills the following requirements:
  - [S9] It has HTTP 200 response code AND
  - [S10] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S11] It received before the Client **RTSP DESCRIBE** request AND
  - [S12] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S13] It invoked for the same Device as for the Client **RTSP SETUP** request AND

- [S14] It invoked after the Client **RTSP SETUP** request AND
- [S15] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- [S16] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S17] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S18] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S19] It invoked after the Client **RTSP PLAY** request AND
  - [S20] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S21] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.12 AAC Audio Streaming Using Media2 Test Cases

6.12.1 Feature Level Requirement:

Validated Feature: AAC Audio Streaming Using Media2 (Media2_AudioStreaming_AAC)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and Audio AAC (Media2 Service) are supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.12.2 Expected Scenarios Under Test:

1. Client connects to Device to configure a media profile and initiate Audio Streaming with AAC encoding type.
2. Client is considered as supporting AAC Audio Streaming if the following conditions are met:
   - Client is able to initiate and retrieve audio stream using Media2 with AAC encoding type.
3. Client is considered as NOT supporting AAC Audio Streaming if ANY of the following is TRUE:
   - Client is unable to initiate and retrieve audio stream using Media2 with AAC encoding type.

6.12.3 AAC AUDIO STREAMING USING MEDIA2

Test Label: Audio Streaming using Media2 - AAC

Test Case ID: MEDIA2_AUDIOSTREAMING_AAC-1

Feature Under Test: AAC Audio Streaming Using Media2 (Media2_AudioStreaming_AAC_Media2_AudioStreaming_AAC)

Test Purpose: To verify that the Client is able to initiate and retrieve audio stream with AAC encoding type.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Audio Streaming of AAC encoding type.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports AAC encoding for Audio streaming using Media2 (Media2_AAC).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message for media profile that contains Audio Source Configuration and Audio Encoder Configuration with AAC Encoding value. GetStreamUri request is set for RTP-Unicast/UDP OR RTP-Multicast/UDP OR RTP/RTSP/TCP OR RTP-Unicast/RTSP/HTTP/TCP transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.

3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description.
4. Device responds with code RTSP 200 OK and SDP information with Media Type: "audio" and with encoding name "MPEG4-GENERIC" or "MP4A-LATM".
5. Client invokes **RTSP SETUP** request without "onvif-replay" Require header and with transport parameter element to to set media session parameters for AAC audio streaming.
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request without "onvif-replay" Require header to start media stream.
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session.
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Test Result:

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

PASS -

- There is Client **RTSP DESCRIBE** request in Test Procedure
- Device response on the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S1] It has RTSP 200 response code AND
  - [S2] SDP packet contains media type "audio" (m=audio) without session attribute "sendonly" (a=sendonly) AND with sessions attribute "rtpmap" with encoding name "MPEG4-GENERIC" or "MP4A-LATM"(see [RFC 3640]) AND
- There is Client **RTSP SETUP** request in Test Procedure fulfills the following requirements:
  - [S3] It invoked for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S4] It invoked after the Client **RTSP DESCRIBE** request AND
  - [S5] RTSP address that was used to send **RTSP SETUP** is correspond to corresponding media Control URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S6] It does not contain **Require** request header field with value is equal to "onvif-replay" AND

- Device response on the **RTSP SETUP** request fulfills the following requirements:
  - [S7] It has RTSP 200 response code AND
- There is a Device response on the **GetStreamUri** request to Media2 Service in Test Procedure fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] It received for the same Device as for the Client **RTSP DESCRIBE** request AND
  - [S10] It received before the Client **RTSP DESCRIBE** request AND
  - [S11] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure fulfills the following requirements:
  - [S12] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S13] It invoked after the Client **RTSP SETUP** request AND
  - [S14] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
  - [S15] It does not contain **Require** request header field with value is equal to "onvif-replay" AND
- Device response on the **RTSP PLAY** request fulfills the following requirements:
  - [S16] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure fulfills the following requirements:
  - [S17] It invoked for the same Device as for the Client **RTSP SETUP** request AND
  - [S18] It invoked after the Client **RTSP PLAY** request AND
  - [S19] RTSP address that was used to send it is correspond to corresponding media Control URL or session Control URL or Content-Base URL from SDP packet (see [RFC 2326, C.1.1 Control URL]) AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S20] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.13 Audio Profile Configuration Using Media2 Test Cases

6.13.1 Feature Level Requirement:

Validated Feature: Audio Profile Configuration Using Media2 (Media2_AudioProfileConfiguration)

Check Condition based on Device Features: Audio (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.13.2 Expected Scenarios Under Test:

1. Client connects to Device to add compatible audio source configuration and audio encoder configuration to a Media Profile.
2. Client is considered as supporting Audio Profile Configuration if the following conditions are met:
   - Client is able to add an audio source configuration to profile using EITHER **AddConfiguration** operation with **Type** element value is equal to **AudioSource** OR **CreateProfile** operation with **Type** element value is equal to **AudioSource**.
   - Client is able to retrieve audio encoder configurations compatible with media profile using **GetAudioEncoderConfigurations** operation with specified ProfileToken element.
   - Client is able to add an audio encoder configuration using **AddConfiguration** operation with Type element value is equal to AudioEncoder.
3. Client is considered as NOT supporting Audio Profile Configuration if ANY of the following is TRUE:
   - Client unable to add an audio source configuration to profile using **AddConfiguration** operation and **CreateProfile** operation OR
   - No valid responses for **CreateProfile** request **Type** element value is equal to **AudioSource** if deteced OR
   - No valid responses for **GetAudioEncoderConfigurations** request with ProfileToken element OR

- No valid responses for **AddConfiguration** request with Type element value is equal to AudioEncoder OR
- No valid responses for **AddConfiguration** request with Type element value is equal to AudioSource is detected.

6.13.3 ADD AUDIO SOURCE CONFIGURATION USING MEDIA2

Test Label: Add Audio Source Configuration

Test Case ID: MEDIA2_AUDIOPROFILECONFIGURATION-1

Feature Under Test: Add Audio Source Configuration using Media2 (Media2_AudioProfileConfiguration_Media2_AddAudioSourceConfiguration)

Test Purpose: To verify that Client is able to add an audio source configuration to a media profile using the **GetAudioSourceConfigurations** and **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type = **AudioSource** present.
- Device supports Media2 Audio feature (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioSourceConfigurations** request message with specified **ProfileToken** to retrieve compatible audio source configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioSourceConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to **AudioSource** and with Configuration token that was recieved in **GetAudioSourceConfigurationsResponse** message for the same media profile to add an audio source configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "AudioSource" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.
- There is Client **GetAudioSourceConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetAudioSourceConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetAudioSourceConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tr2:GetAudioSourceConfigurationsResponse** AND
  - [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to AudioSource from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.13.4 CREATE MEDIA PROFILE WITH AUDIO SOURCE CONFIGURATION USING MEDIA2

Test Label: Create Media2 Profile with Audio Source Configuration

Test Case ID: MEDIA2_AUDIOPROFILECONFIGURATION-2

Feature Under Test: Create Media2 Profile with Audio Source Configuration (Media2_AudioProfileConfiguration_Media2_CreateMediaProfileWithAudioSourceConfiguration)

Test Purpose: To verify that Client is able to create media profile with audio source configuration using the **CreateProfile** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateProfile** operation with Type = **AudioSource** present.
- Device supports Media2 Audio feature (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateProfile** request message with **tr2:Configuration\tr2:Type** = **AudioSource** and with specified **tr2:Token** element for this Configuration to create profile with audio source configuration on the Device.
2. Device responds with code HTTP 200 OK and **CreateProfileResponse** message.

Test Result:

PASS -

- Client **CreateProfile** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateProfile** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateProfile** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **"AudioSource"** AND
  - [S3] **tr2:Configuration** element with **tr2:Type** value is equal to **"AudioSource"** has **tr2:Token** element AND
- Device response to the **CreateProfile** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:CreateProfileResponse**.

FAIL -

- The Client failed PASS criteria.

6.13.5 GET AUDIO ENCODER CONFIGURATIONS COMPATIBLE WITH PROFILE USING MEDIA2

Test Label: Get Audio Encoder Configurations

Test Case ID: MEDIA2_AUDIOPROFILECONFIGURATION-3

Feature Under Test: Get Audio Encoder Configurations Compatible With Media2 Profile (Media2_AudioProfileConfiguration_Media2_GetCompatibleAudioEncoderConfigurations)

Test Purpose: To verify that list of audio encoder configurations compatible with a media profile is received by Client using the **GetAudioEncoderConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioEncoderConfigurations** operation with specified **ProfileToken** element present.
- Device supports Media2 Audio feature (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioEncoderConfigurations** request message with **ProfileToken** element to retrieve a list of audio encoder configurations compatible with requested media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioEncoderConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioEncoderConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioEncoderConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurations** AND
  - [S2] It has **tr2:ProfileToken** element AND

- Device response to the **GetAudioEncoderConfigurations** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.13.6 ADD AUDIO ENCODER CONFIGURATION USING MEDIA2

Test Label: Add Audio Encoder Configuration

Test Case ID: MEDIA2_AUDIOPROFILECONFIGURATION-4

Feature Under Test: Add Audio Encoder Configuration Using Media2 (Media2_AudioProfileConfiguration_Media2_AddAudioEncoderConfiguration)

Test Purpose: To verify that Client is able to add an audio encoder configuration to a media profile using the **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type value is equal to "VideoEncoder" present.
- Device supports Media2 Audio feature (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioEncoderConfigurations** request message with specified **ProfileToken** to retrieve compatible audio encoder configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioEncoderConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to AudioEncoder and with Configuration token that was recieved in **GetAudioEncoderConfigurationsResponse** message for the same media profile to add an audio encoder configuration to specified media profile on the Device.

4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "AudioEncoder" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.
- There is Client **GetAudioEncoderConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetAudioEncoderConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetAudioEncoderConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tr2:GetAudioEncoderConfigurationsResponse** AND
  - [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to AudioEncoder from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.14 Audio Backchannel Streaming Using Media2 Test Cases

6.14.1 Feature Level Requirement:

Validated Feature: Audio Backchannel Streaming Using Media2 (Media2_AudioBackchannelStreaming)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and Audio Output (Media2 Service) are supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.14.2 Expected Scenarios Under Test:

1. Client connects to Device to stream audio for backchannel using Media2.
2. Client is considered as supporting Audio Backchannel Streaming if the following conditions are met:
   - Client is able to stream unicast audio for backchannel using **G.711** AND
   - Client is able to stream unicast audio for backchannel using **AAC** if supported.
3. Client is considered as NOT supporting Audio Backchannel Streaming if ANY of the following is TRUE:
   - No unicast G.711 Audio Backchannel Streaming attempts were found OR
   - Unicast G.711 Audio Backchannel Streaming attempts have failed OR
   - Detected unicast AAC Audio Backchannel Streaming attempts have failed.

6.14.3 G.711 AUDIO BACKCHANNEL STREAMING USING MEDIA2

Test Label: Audio Backchannel Streaming Using Media2 - G.711

Test Case ID: MEDIA2_AUDIOBACKCHANNELSTREAMING-1

Feature Under Test: G.711 Audio Backchannel Streaming Using Media2 (Media2_AudioBackchannelStreaming_Media2_G711AudioBackchannelStreaming)

Test Purpose: To verify that unicast audio backchannel streaming to Device was successfully started by Client.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with unicast audio backchannel streaming using Media2 with G.711 encoding.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports G.711 decoding for Media2 Audio Outputs (Media2_AudioOutputG711).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message to Media2 Service for media profile that contains Audio Output Configuration and Audio Decoder Configuration with RTP-Unicast/UDP OR RTP/RTSP/TCP OR RTP-Unicast/RTSP/HTTP/TCP transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
4. Device responds with code RTSP 200 OK with SDP that contains media type "audio" with session attribute "sendonly".
5. Client invokes **RTSP SETUP** request with transport parameter element to set media session parameters for audio backchannel with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request to start media stream with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

Test Result:

PASS -

- Client **RTSP DESCRIBE** request in Test Procedure fulfills the following requirements:
  - [S1] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S2] It has RTSP 200 response code AND
  - [S3] SDP packet contains media type "audio" (m=audio) with session attribute "sendonly" (a=sendonly) and sessions attribute "rtpmap" with encoding name "PCMU" AND
- There is Client **RTSP SETUP** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked after the Client **RTSP DESCRIBE** request AND
  - [S6] RTSP address that was used to send **RTSP SETUP** is corresponds to media type "audio" with session attribute "sendonly" depending on media session attribute, general session attribute and address that was used for the **RTSP DESCRIBE** request (see [RFC 2326]) AND
  - [S7] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP SETUP** request fulfills the following requirements:
  - [S8] It has RTSP 200 response code AND
- There is **GetStreamUri** request to Media2 Service in Test Procedure that fulfills the following requirements:
  - [S9] **tr2:Protocol** element value is equal EITHER RtspUnicast OR RTSP OR RtspOverHttp AND
- There is a Device response to the **GetStreamUri** request to Media2 Service in Test Procedure that fulfills the following requirements:
  - [S10] It has HTTP 200 response code AND
  - [S11] It is received from the same Device as the response for **RTSP DESCRIBE** request AND
  - [S12] It is received before the Client **RTSP DESCRIBE** request AND
  - [S13] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure that fulfills the following requirements:

- [S14] It is invoked for the same Device as the response for **RTSP SETUP** request AND
- [S15] It is invoked after the Client **RTSP SETUP** request AND
- [S16] RTSP address that was used to send it should be equal to address that was used for the **RTSP DESCRIBE** request AND
- [S17] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND

- Device response to the **RTSP PLAY** request fulfills the following requirements:
  - [S18] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure that fulfills the following requirements:
  - [S19] It is invoked for the same Device as the response for **RTSP SETUP** request AND
  - [S20] It is invoked after the Client **RTSP PLAY** request AND
  - [S21] RTSP address that was used to send it should be equal to address that was used for the **RTSP DESCRIBE** request AND
  - [S22] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S23] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.14.4 AAC AUDIO BACKCHANNEL STREAMING USING MEDIA2

Test Label: Audio Backchannel Streaming - AAC

Test Case ID: MEDIA2_AUDIOBACKCHANNELSTREAMING-2

Feature Under Test: AAC Audio Backchannel Streaming Using Media2 (Media2_AudioBackchannelStreaming_Media2_AACAudioBackchannelStreaming)

Test Purpose: To verify that audio backchannel streaming to Device was successfully started by Client.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with unicast audio backchannel streaming using Media2 with AAC encoding.
- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetStreamUri** for Media2 Service.
- Device supports AAC encoding for Media2 Audio Outputs (Media2_AudioOutputAAC).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetStreamUri** request message to Media2 Service for media profile that contains Audio Output Configuration and Audio Decoder Configuration with RTP-Unicast/UDP OR RTP/RTSP/TCP OR RTP-Unicast/RTSP/HTTP/TCP transport.
2. Device responds with code HTTP 200 OK and **GetStreamUriResponse** message.
3. Client invokes **RTSP DESCRIBE** request to retrieve media stream description with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
4. Device responds with code RTSP 200 OK with SDP that contains media type "audio" with session attribute "sendonly".
5. Client invokes **RTSP SETUP** request with transport parameter element to set media session parameters for audio backchannel with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
6. Device responds with code RTSP 200 OK.
7. Client invokes **RTSP PLAY** request to start media stream with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
8. Device responds with code RTSP 200 OK.
9. Client invokes **RTSP TEARDOWN** request to terminate the RTSP session with **Require** tag in RTSP header that contains "www.onvif.org/ver20/backchannel".
10. If Device sends response to RTSP TEARDOWN, it has code RTSP 200 OK or RTSP 454.

Note: RTSP requests and RTSP response could be tunneled in HTTP if RTP-Unicast/RTSP/HTTP/TCP transport is used.

Test Result:

PASS -

- Client **RTSP DESCRIBE** request in Test Procedure fulfills the following requirements:
  - [S1] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP DESCRIBE** request fulfills the following requirements:
  - [S2] It has RTSP 200 response code AND
  - [S3] SDP packet contains media type "audio" (m=audio) with session attribute "sendonly" (a=sendonly) and sessions attribute "rtpmap" with encoding name "mpeg4-generic" or "MP4A-LATM" AND
- There is Client **RTSP SETUP** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked after the Client **RTSP DESCRIBE** request AND
  - [S6] RTSP address that was used to send **RTSP SETUP** is corresponds to media type "audio" with session attribute "sendonly" depending on media session attribute, general session attribute and address that was used for the **RTSP DESCRIBE** request (see [RFC 2326]) AND
  - [S7] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP SETUP** request fulfills the following requirements:
  - [S8] It has RTSP 200 response code AND
- There is **GetStreamUri** request to Media2 Service in Test Procedure that fulfills the following requirements:
  - [S9] **tr2:Protocol** element value is equal EITHER RtspUnicast OR RTSP OR RtspOverHttp AND
- There is a Device response to the **GetStreamUri** request to Media2 Service in Test Procedure that fulfills the following requirements:
  - [S10] It has HTTP 200 response code AND
  - [S11] It is received from the same Device the response for **RTSP DESCRIBE** request AND
  - [S12] It is received before the Client **RTSP DESCRIBE** request AND
  - [S13] It contains **tr2:Uri** element which value is equal to RTSP address that was used to send the **RTSP DESCRIBE** request AND
- There is Client **RTSP PLAY** request in Test Procedure that fulfills the following requirements:
  - [S14] It is invoked for the same Device as the response for **RTSP SETUP** request AND

- [S15] It is invoked after the Client **RTSP SETUP** request AND
- [S16] RTSP address that was used to send it should be equal to address that was used for the **RTSP DESCRIBE** request AND
- [S17] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND

- Device response to the **RTSP PLAY** request fulfills the following requirements:
  - [S18] It has RTSP 200 response code AND
- There is Client **RTSP TEARDOWN** request in Test Procedure that fulfills the following requirements:
  - [S19] It is invoked for the same Device the response for **RTSP SETUP** request AND
  - [S20] It is invoked after the Client **RTSP PLAY** request AND
  - [S21] RTSP address that was used to send it should be equal to address that was used for the **RTSP DESCRIBE** request AND
  - [S22] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- If there is Device response on the **RTSP TEARDOWN** request then it fulfills the following requirements:
  - [S23] It has RTSP 200 response code.

FAIL -

- The Client failed PASS criteria.

6.15 Audio Output Profile Configuration Using Media2 Test Cases

6.15.1 Feature Level Requirement:

Validated Feature: Audio Output Media2 Profile Configuration (Media2_AudioOutputProfileConfiguration)

Check Condition based on Device Features: Audio Output (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.15.2 Expected Scenarios Under Test:

1. Client connects to Device to add compatible audio output configuration and decoder configuration to a Media Profile.
2. Client is considered as supporting Audio Output Profile Configuration if the following conditions are met:
   - Client is able to add an audio output configuration to profile using EITHER **AddConfiguration** operation with **Type** element value is equal to **AudioOutput** OR **CreateProfile** operation with **Type** element value is equal to **AudioOutput**.
   - Client is able to retrieve audio decoder configurations compatible with media profile using **GetAudioDecoderConfigurations** operation with specified ProfileToken element.
   - Client is able to add an audio decoder configurations using **AddConfiguration** operation with Type element value is equal to AudioDecoder.
3. Client is considered as NOT supporting Audio Output Profile Configuration if ANY of the following is TRUE:
   - Client unable to add an audio output configuration to profile using **AddConfiguration** operation and **CreateProfile** operation OR
   - No valid responses for **CreateProfile** request **Type** element value is equal to **AudioOutput** if deteced OR
   - No valid responses for **GetAudioDecoderConfigurations** request with ProfileToken element OR
   - No valid responses for **AddConfiguration** request with Type element value is equal to AudioDecoder OR
   - No valid responses for **AddConfiguration** request with Type element value is equal to AudioOutput is detected.

6.15.3 ADD AUDIO OUTPUT CONFIGURATION USING MEDIA2

Test Label: Add Audio Output Configuration

Test Case ID: MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-1

Feature Under Test: Add Audio Output Configuration Using Media2 (Media2_AudioOutputProfileConfiguration_Media2_AddAudioOutputConfiguration)

Test Purpose: To verify that Client is able to add an audio output configuration to a media profile using the **GetAudioOutputConfigurations** and **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type = **AudioOutput** present.
- Device supports Media2 Audio Output feature (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioOutputConfigurations** request message with specified **ProfileToken** to retrieve compatible audio output configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioOutputConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to **AudioOutput** and with Configuration token that was recieved in **GetAudioOutputConfigurationsResponse** message for the same media profile to add an audio output configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "AudioOutput" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.

- There is Client **GetAudioOutputConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetAudioOutputConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetAudioOutputConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tr2:GetAudioOutputConfigurationsResponse** AND
  - [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to AudioOutput from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.15.4 CREATE MEDIA PROFILE WITH AUDIO OUTPUT CONFIGURATION USING MEDIA2

Test Label: Create Media2 Profile with Audio Output Configuration

Test Case ID: MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-2

Feature Under Test: Create Media2 Profile With Audio Output Configuration (Media2_AudioOutputProfileConfiguration_Media2_CreateMediaProfileWithAudioOutputConfiguration)

Test Purpose: To verify that Client is able to create media profile with audio output configuration using the **CreateProfile** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateProfile** operation with Type = **AudioOutput** present.
- Device supports Media2 Audio Output feature (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateProfile** request message with **tr2:Configuration\tr2:Type** = **AudioOutput** and with specified **tr2:Token** element for this Configuration to create profile with audio output configuration on the Device.
2. Device responds with code HTTP 200 OK and **CreateProfileResponse** message.

Test Result:

PASS -

- Client **CreateProfile** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateProfile** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateProfile** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **"AudioOutput"** AND
  - [S3] **tr2:Configuration** element with **tr2:Type** value is equal to **"AudioOutput"** has **tr2:Token** element AND
- Device response to the **CreateProfile** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:CreateProfileResponse**.

FAIL -

- The Client failed PASS criteria.

6.15.5 GET AUDIO DECODER CONFIGURATIONS COMPATIBLE WITH PROFILE USING MEDIA2

Test Label: Get Audio Decoder Configurations

Test Case ID: MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-3

Feature Under Test: Get Audio Decoder Configurations Compatible With Media2 Profile (Media2_AudioOutputProfileConfiguration_Media2_GetCompatibleAudioDecoderConfigurations)

Test Purpose: To verify that list of audio decoder configurations compatible with a media profile is received by Client using the **GetAudioDecoderConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioDecoderConfigurations** operation present.
- Device supports Media2 Audio Output feature (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioDecoderConfigurations** request message with **ProfileToken** element to retrieve a list of audio decoder configurations compatible with requested media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioDecoderConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioDecoderConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioDecoderConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurations** AND
  - [S2] It has **tr2:ProfileToken** element AND
- Device response to the **GetAudioDecoderConfigurations** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.15.6 ADD AUDIO DECODER CONFIGURATION USING MEDIA2

Test Label: Add Audio Decoder Configuration

Test Case ID: MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-4

Feature Under Test: Add Audio Decoder Configuration Using Media2 (Media2_AudioOutputProfileConfiguration_Media2_AddAudioDecoderConfiguration)

Test Purpose: To verify that Client is able to add an audio decoder configuration to a media profile using the **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation present.
- Device supports Media2 Audio Output feature (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioDecoderConfigurations** request message with specified **ProfileToken** to retrieve compatible audio decoder configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioDecoderConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to AudioDecoder and with Configuration token that was recieved in **GetAudioDecoderConfigurationsResponse** message for the same media profile to add an audio decoder configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to "AudioDecoder" AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND

- [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.

- There is Client **GetAudioDecoderConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetAudioDecoderConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetAudioDecoderConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurationsResponse** AND
  - [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to AudioDecoder from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.16 Analytics Profile Configuration Using Media2 Test Cases

6.16.1 Feature Level Requirement:

Validated Feature: Analytics Media2 Profile Configuration (Media2_AnalyticsProfileConfiguration)

Check Condition based on Device Features: Analytics (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.16.2 Expected Scenarios Under Test:

1. Client connects to Device to add compatible analytics configuration to a Media Profile.

2. Client is considered as supporting Analytics Profile Configuration if the following conditions are met:
   - Client is able to retrieve analytics configurations compatible with media profile using **GetAnalyticsConfigurations** operation with specified ProfileToken element.
   - Client is able to add an analytics configuration using **AddConfiguration** operation with Type element value is equal to **Analytics**.
3. Client is considered as NOT supporting Analytics Profile Configuration if ANY of the following is TRUE:
   - No valid responses for **GetAnalyticsConfigurations** request with ProfileToken element OR
   - No valid responses for **AddConfiguration** request with Type element value is equal to **Analytics**.

6.16.3 GET ANALYTICS CONFIGURATIONS COMPATIBLE WITH PROFILE USING MEDIA2

Test Label: Get Analytics Configurations

Test Case ID: MEDIA2_ANALYTICSPROFILECONFIGURATION-1

Feature Under Test: Get Analytics Configurations Compatible With Media2 Profile (Media2_AnalyticsProfileConfiguration_Media2_GetCompatibleAnalyticsConfigurations)

Test Purpose: To verify that list of analytics configurations compatible with a media profile is received by Client using the **GetAnalyticsConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAnalyticsConfigurations** operation with specified **ProfileToken** element present.
- Device supports Media2 Analytics feature (Media2_Analytics).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAnalyticsConfigurations** request message with **ProfileToken** element to retrieve a list of analytics configurations compatible with requested media profile from the Device.

2. Device responds with code HTTP 200 OK and **GetAnalyticsConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAnalyticsConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAnalyticsConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAnalyticsConfigurations** AND
  - [S2] It has **tr2:ProfileToken** element AND
- Device response to the **GetAnalyticsConfigurations** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:GetAnalyticsConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.16.4 ADD ANALYTICS CONFIGURATION USING MEDIA2

Test Label: Add Analytics Configuration

Test Case ID: MEDIA2_ANALYTICSPROFILECONFIGURATION-2

Feature Under Test: Add Analytics Configuration To Media2 Profile (Media2_AnalyticsProfileConfiguration_Media2_AddAnalyticsConfiguration)

Test Purpose: To verify that Client is able to add an analytics configuration to a media profile using the **GetAnalyticsConfigurations** and **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type = **Analytics** present.
- Device supports Media2 Analytics feature (Media2_Analytics).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAnalyticsConfigurations** request message with specified **ProfileToken** to retrieve compatible analytics configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetAnalyticsConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to **Analytics** and with Configuration token that was recieved in **GetAnalyticsConfigurationsResponse** message for the same media profile to add an analytics configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **Analytics** AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.
- There is Client **GetAnalyticsConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tr2:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetAnalyticsConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetAnalyticsConfigurations** request fulfills the following requirements:

- [S8] It has HTTP 200 response code AND
- [S9] **soapenv:Body** element has child element **tr2:GetAnalyticsConfigurationsResponse** AND
- [S10] It contains **tr2:Configurations** element with **@token** attribute value equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to **Analytics** from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.17 Video Source Mode Test Cases

6.17.1 Feature Level Normative Reference:

Validated Feature: Video Source Mode (VideoSourceMode)

Check Condition based on Device Features: Video Source Mode (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.17.2 Expected Scenarios Under Test:

1. Client connects to Device to request the information for current video source mode and settable video source modes of specified video source using **GetVideoSourceModes** operation.
2. Client changes current video source mode using the **SetVideoSourceMode** operation.
3. Client is considered as supporting Video Source Mode if the following conditions are met:
   - Client is able to retrieve current video source mode and settable video source modes using **GetVideoSourceModes** operation AND
   - Client is able to modify source mode using **SetVideoSourceMode** operation.
4. Client is considered as NOT supporting Video Source Mode if ANY of the following is TRUE:
   - No valid response to **GetVideoSourceModes** request OR

- No valid response to **SetVideoSourceMode** request.

6.17.3 GET VIDEO SOURCE MODES

Test Label: Get Video Source Modes

Test Case ID: VIDEOSOURCEMODE-1

Feature Under Test: Get Video Source Modes (VideoSourceMode_GetVideoSourceModes)

Test Purpose: To verify that video source modes provided by Device is received by Client using the **GetVideoSourceModes** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceModes** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video Source Mode feature (Media2_VideoSourceMode).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceModes** request message to retrieve a video source modes from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceModesResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceModes** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoSourceModes** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceModes** AND
- Device response on the **GetVideoSourceModes** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetVideoSourceModesResponse**.

FAIL -

- The Client failed PASS criteria.

6.17.4 SET VIDEO SOURCE MODE

Test Label: Set Video Source Mode

Test Case ID: VIDEOSOURCEMODE-2

Feature Under Test: Set Video Source Mode (VideoSourceMode_SetVideoSourceMode)

Test Purpose: To verify that current video source mode can be changed by Client using the **SetVideoSourceMode** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetVideoSourceMode** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video Source Mode feature (Media2_VideoSourceMode).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetVideoSourceMode** request message to change a video source mode on the Device.
2. Device responds with code HTTP 200 OK and **SetVideoSourceModeResponse** message.

Test Result:

PASS -

- Client **SetVideoSourceMode** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetVideoSourceMode** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetVideoSourceMode** AND
- Device response on the **SetVideoSourceMode** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetVideoSourceModeResponse**.

FAIL -

- The Client failed PASS criteria.

6.18 Video Source Configuration Using Media2 Test Cases

6.18.1 Feature Level Normative Reference:

Validated Feature: Video Source Configuration Using Media2 (Media2_VideoSourceConfiguration)

Check Condition based on Device Features: Video (Media2 Service) or VideoSource (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.18.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Video Source Configuration.
2. Client is considered as supporting Video Source Configuration if the following conditions are met:
   - Client is able to retrieve video source configuratoins using **GetVideoSourceConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve video source configuratoin options using **GetVideoSourceConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify video source configuratoin using **SetVideoSourceConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Video Source Configuration if ANY of the following is TRUE:
   - No valid response to **GetVideoSourceConfigurations** request (Media2 Service) OR
   - No valid response to **GetVideoSourceConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetVideoSourceConfiguration** request (Media2 Service) OR

6.18.3 GET VIDEO SOURCE CONFIGURATIONS USING MEDIA2

Test Label: Video Source Configuration - Get Video Source Configurations

Test Case ID: MEDIA2_VIDEOSOURCECONFIGURATION-1

Feature Under Test: Get Video Source Configurations Using Media2 (Media2_VideoSourceConfiguration_Media2_GetVideoSourceConfigurations)

Test Purpose: To verify that video source configuration provided by Device is received by Client using the **GetVideoSourceConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video feature for Media2 Service (Media2_Video) OR Device supports Video Source feature for Media2 Service (Media2_VideoSource).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceConfigurations** request message to retrieve an video source configuration or a list of video source configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoSourceConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurations** AND
- Device response on the **GetVideoSourceConfigurations** request fulfills the following requirements:

- [S2] It has HTTP 200 response code AND
- [S3] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.18.4 GET VIDEO SOURCE CONFIGURATION OPTIONS USING MEDIA2

Test Label: Video Source Configuration - Get Video Source Configuration Options

Test Case ID: MEDIA2_VIDEOSOURCECONFIGURATION-2

Feature Under Test: Get Video Source Configuration Options Using Media2 (Media2_VideoSourceConfiguration_Media2_GetVideoSourceConfigurationOptions)

Test Purpose: To verify that video source configuration options provided by Device is received by Client using the **GetVideoSourceConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video feature for Media2 Service (Media2_Video) OR Device supports Video Source feature for Media2 Service (Media2_VideoSource).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceConfigurationOptions** request message to retrieve an video source configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **GetVideoSourceConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationOptions** AND
- Device response on the **GetVideoSourceConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.18.5 SET VIDEO SOURCE CONFIGURATION USING MEDIA2

Test Label: Video Source Configuration - Set Video Source Configuration

Test Case ID: MEDIA2_VIDEOSOURCECONFIGURATION-3

Feature Under Test: Set Video Source Configuration Using Media2 (Media2_VideoSourceConfiguration_Media2_SetVideoSourceConfiguration)

Test Purpose: To verify that Client is able to change video source configuration provided by Device using the **SetVideoSourceConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetVideoSourceConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video feature for Media2 Service (Media2_Video) OR Device supports Video Source feature for Media2 Service (Media2_VideoSource).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetVideoSourceConfiguration** request message to change an video source configuration on the Device.

2. Device responds with code HTTP 200 OK and **SetVideoSourceConfigurationResponse** message.

Test Result:

PASS -

- Client **SetVideoSourceConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetVideoSourceConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetVideoSourceConfiguration** AND
- Device response on the **SetVideoSourceConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetVideoSourceConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

6.19 List Video Source Configurations Using Media2 Test Cases

6.19.1 Feature Level Normative Reference:

Validated Feature: List Video Source Configurations Using Media2 (Media2_ListVideoSourceConfigurations)

Check Condition based on Device Features: Video (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.19.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve all Video Source Configurations availible on device.

2. Client is considered as supporting List Video Source Configurations if the following conditions are met:
   - Client is able to retrieve full list of video source configuratoins using **GetVideoSourceConfigurations** operation with skipped ConfigurationToken and ProfileToken in request (Media2 Service).
3. Client is considered as NOT supporting List Video Source Configurations if ANY of the following is TRUE:
   - No valid response to **GetVideoSourceConfigurations** request that does not contain ConfigurationToken and ProfileToken elements (Media2 Service) OR

6.19.3 LIST VIDEO SOURCE CONFIGURATIONS USING MEDIA2

Test Label: Video Source Configuration - Get Video Source Configurations

Test Case ID: MEDIA2_LISTVIDEOSOURCECONFIGURATIONS-1

Feature Under Test: Get Video Source Configurations Using Media2 (Media2_ListVideoSourceConfigurations_Media2_GetVideoSourceConfigurationsRequest)

Test Purpose: To verify that video source configurations provided by Device is received by Client using the **GetVideoSourceConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceConfigurations** operation with skipped ConfigurationToken element and with skipped ProfileToken element present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceConfigurations** request message with skipped ConfigurationToken element and with skipped ProfileToken element to retrieve a list of video source configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoSourceConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurations** AND
  - [S2] **tr2:GetVideoSourceConfigurations** element does not have child element **tr2:ConfigurationToken** AND
  - [S3] **tr2:GetVideoSourceConfigurations** element does not have child element **tr2:ProfileToken** AND
- Device response on the **GetVideoSourceConfigurations** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.20 OSD Configuration Using Media2 Test Cases

6.20.1 Feature Level Normative Reference:

Validated Feature: OSD Configuration (Media2_OSDConfiguration)

Check Condition based on Device Features: OSD (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.20.2 Expected Scenarios Under Test:

1. Client connects to Device to list OSDs, create OSD, and modify OSD on the device.
2. Client is considered as supporting OSD Configuration if the following conditions are met:
   - Client supports **Media2_ListVideoSourceConfigurations_Media2_GetVideoSourceConfigurationsRequest**

feature (please see MEDIA2_LISTVIDEOSOURCECONFIGURATIONS-1 LIST VIDEO SOURCE CONFIGURATIONS USING MEDIA2 section) AND

- Client is able to retrieve OSD configuratoins using **GetOSDs** operation (Media2 Service) AND
- Client is able to create OSD text configurations using **CreateOSD** operation with Type = **Text** (Media2 Service) AND
- If supported, Client is able to create OSD image configurations using **CreateOSD** operation with Type = **Image** (Media2 Service) AND
- Client is able to retrieve OSD options using **GetOSDOptions** operation (Media2 Service).
- Client is able to modify OSD using **SetOSD** operation (Media2 Service).

3. Client is considered as NOT supporting OSD Configuration if ANY of the following is TRUE:
   - Client does not support **Media2_ListVideoSourceConfigurations_Media2_GetVideoSourceConfigurationsRequest** feature (please see MEDIA2_LISTVIDEOSOURCECONFIGURATIONS-1 LIST VIDEO SOURCE CONFIGURATIONS USING MEDIA2 section) OR
   - No valid response to **GetOSDs** request (Media2 Service) OR
   - No valid response to **CreateOSD** operation with Type = **Text** (Media2 Service) OR
   - No valid response to **CreateOSD** operation with Type = **Image** if detected (Media2 Service) OR
   - No valid response to **GetOSDOptions** operation (Media2 Service) OR
   - No valid response to **SetOSD** operation (Media2 Service).

6.20.3 LIST VIDEO SOURCE CONFIGURATIONS USING MEDIA2

Test Label: Video Source Configuration - Get Video Source Configurations

Test Case ID: MEDIA2_LISTVIDEOSOURCECONFIGURATIONS-1

Feature Under Test: Get Video Source Configurations Using Media2 (Media2_ListVideoSourceConfigurations_Media2_GetVideoSourceConfigurationsRequest)

Test Purpose: To verify that video source configurations provided by Device is received by Client using the **GetVideoSourceConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceConfigurations** operation with skipped ConfigurationToken element and with skipped ProfileToken element present.
- Device supports Media2 Service (Media2Service).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceConfigurations** request message with skipped ConfigurationToken element and with skipped ProfileToken element to retrieve a list of video source configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoSourceConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurations** AND
  - [S2] **tr2:GetVideoSourceConfigurations** element does not have child element **tr2:ConfigurationToken** AND
  - [S3] **tr2:GetVideoSourceConfigurations** element does not have child element **tr2:ProfileToken** AND
- Device response on the **GetVideoSourceConfigurations** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND
  - [S5] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.20.4 GET OSD CONFIGURATIONS USING MEDIA2

Test Label: OSD Configuration - Get OSDs

Test Case ID: MEDIA2_OSDCONFIGURATION-1

Feature Under Test: Get OSDs (Media2_OSDConfiguration_Media2_GetOSDConfigurations)

Test Purpose: To verify that existing OSD configurations is received by Client using the **GetOSDs** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetOSDs** operation with skipped **OSDToken** element for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Media2 OSD feature (Media2_OSD).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetOSDs** request message to retrieve OSD configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetOSDsResponse** message.

Test Result:

PASS -

- Client **GetOSDs** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetOSDs** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetOSDs** AND
  - [S2] **tr2:GetOSDs** element does not contain child element **tr2:OSDToken** AND
- Device response on the **GetOSDs** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:GetOSDsResponse**.

FAIL -

- The Client failed PASS criteria.

6.20.5 CREATE TEXT OSD USING MEDIA2

Test Label: OSD Configuration - Create OSD

Test Case ID: MEDIA2_OSDCONFIGURATION-2

Feature Under Test: Create Text OSD (Media2_OSDConfiguration_Media2_CreateTextOSD)

Test Purpose: To verify that Client is able to create text OSD using the **CreateOSD** operation with Type value is equal to **Text**

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateOSD** operation for Media2 Service with Type value is equal to **Text** present.
- Device supports Media2 Service (Media2Service).
- Device supports Media2 OSD feature (Media2_OSD).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateOSD** with Type = **Text** request message to create text OSD on the Device.
2. Device responds with code HTTP 200 OK and **CreateOSDResponse** message.

Test Result:

PASS -

- Client **CreateOSD** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateOSD** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateOSD** AND
  - [S2] **tr2:CreateOSD/tr2:OSD/tt:Type** element value is equal to **Text** AND
- Device response on the **CreateOSD** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:CreateOSDResponse**.

FAIL -

- The Client failed PASS criteria.

6.20.6 CREATE IMAGE OSD USING MEDIA2

Test Label: OSD Configuration - Create OSD

Test Case ID: MEDIA2_OSDCONFIGURATION-3

Feature Under Test: Create Image OSD (Media2_OSDConfiguration_Media2_CreateImageOSD)

Test Purpose: To verify that Client is able to create image OSD using the **CreateOSD** operation with Type value is equal to **Image**

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateOSD** operation for Media2 Service with Type value is equal to **Image** present.
- Device supports Media2 Service (Media2Service).
- Device supports Media2 OSD feature (Media2_OSD).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateOSD** with Type = **Image** request message to create image OSD on the Device.
2. Device responds with code HTTP 200 OK and **CreateOSDResponse** message.

Test Result:

PASS -

- Client **CreateOSD** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateOSD** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateOSD** AND
  - [S2] **tr2:CreateOSD/tr2:OSD/tt:Type** element value is equal to **Image** AND
- Device response on the **CreateOSD** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND

- [S4] **soapenv:Body** element has child element **tr2:CreateOSDResponse**.

FAIL -

- The Client failed PASS criteria.

6.20.7 GET OSD OPTIONS USING MEDIA2

Test Label: OSD Configuration - Get OSD Options

Test Case ID: MEDIA2_OSDCONFIGURATION-4

Feature Under Test: Get OSD Options (Media2_OSDConfiguration_Media2_GetOSDOptions)

Test Purpose: To verify that OSD options provided by Device is received by Client using the **GetOSDOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetOSDOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Media2 OSD feature (Media2_OSD).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetOSDOptions** request message to retrieve an OSD options from the Device.
2. Device responds with code HTTP 200 OK and **GetOSDOptionsResponse** message.

Test Result:

PASS -

- Client **GetOSDOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetOSDOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetOSDOptions** AND
- Device response on the **GetOSDOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND

- [S3] **soapenv:Body** element has child element **tr2:GetOSDOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.20.8 SET OSD USING MEDIA2

Test Label: OSD Configuration - Set OSD Configuration

Test Case ID: MEDIA2_OSDCONFIGURATION-5

Feature Under Test: Set OSD (Media2_OSDConfiguration_Media2_SetOSDConfiguration)

Test Purpose: To verify that Client is able to change OSD configuration provided by Device using the **SetOSD** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetOSD** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Media2 OSD feature (Media2_OSD).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetOSD** request message to change an OSD configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetOSDResponse** message.

Test Result:

PASS -

- Client **SetOSD** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetOSD** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetOSD** AND
- Device response on the **SetOSD** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetOSDResponse**.

FAIL -

- The Client failed PASS criteria.

6.21 Media Profile Management Test Cases

6.21.1 Feature Level Requirement:

Validated Feature: Media2 Profile Management (Media2_MediaProfileManagement)

Check Condition based on Device Features: Real Time Streaming (Media2 Service) and Video (Media2 Service) are supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.21.2 Expected Scenarios Under Test:

1. Client connects to Device to create media profile and query the maximum number of concurrent streams using the GetVideoSourceConfigurations and GetVideoEncoderInstances operations.

2. Client is considered as supporting Media Profile Management if the following conditions are met:

   - Client is able to create media profile using **CreateProfile** operation with **Type** element value is equal to EITHER **VideoSource** (please see MEDIA2_MEDIAPROFILEMANAGEMENT-1 CREATE MEDIA PROFILE WITH VIDEO SOURCE CONFIGURATION USING MEDIA2 section) OR **AudioSource** (please see MEDIA2_AUDIOPROFILECONFIGURATION-2 CREATE MEDIA PROFILE WITH AUDIO SOURCE CONFIGURATION USING MEDIA2 section) OR **AudioOutput** (please see MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-2 CREATE MEDIA PROFILE WITH AUDIO OUTPUT CONFIGURATION USING MEDIA2 section) AND

   - Client is able to retrieve minimum number of guaranteed video encoder instances for video source configuration using **GetVideoEncoderInstances** operation.

   - Client supports **Media2_VideoSourceConfiguration_Media2_GetVideoSourceConfigurations** feature (please see MEDIA2_VIDEOSOURCECONFIGURATION-1 GET VIDEO SOURCE CONFIGURATIONS USING MEDIA2 section).

3. Client is considered as NOT supporting Video Profile Configuration if ANY of the following is TRUE:

   - No valid responses for **CreateProfile** request **Type** element value is equal to **VideoSource** if deteced OR
   - Client does not support the following features: **Media2_MediaProfileManagement_Media2_CreateMediaProfileWithVideoSourceConfiguration** (please see MEDIA2_MEDIAPROFILEMANAGEMENT-1 CREATE MEDIA PROFILE WITH VIDEO SOURCE CONFIGURATION USING MEDIA2 section) AND **Media2_AudioProfileConfiguration_Media2_CreateMediaProfileWithAudioSourceConfiguration** (please see MEDIA2_AUDIOPROFILECONFIGURATION-2 CREATE MEDIA PROFILE WITH AUDIO SOURCE CONFIGURATION USING MEDIA2 section) AND **Media2_AudioOutputProfileConfiguration_Media2_CreateMediaProfileWithAudioOutputConfigur** (please see MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-2 CREATE MEDIA PROFILE WITH AUDIO OUTPUT CONFIGURATION USING MEDIA2 section) OR
   - No valid responses for **GetVideoEncoderInstances** request OR
   - Client does not support **Media2_VideoSourceConfiguration_Media2_GetVideoSourceConfigurations** feature (please see MEDIA2_VIDEOSOURCECONFIGURATION-1 GET VIDEO SOURCE CONFIGURATIONS USING MEDIA2 section).

6.21.3 CREATE MEDIA PROFILE WITH AUDIO SOURCE CONFIGURATION USING MEDIA2

Test Label: Create Media2 Profile with Audio Source Configuration

Test Case ID: MEDIA2_AUDIOPROFILECONFIGURATION-2

Feature Under Test: Create Media2 Profile with Audio Source Configuration (Media2_AudioProfileConfiguration_Media2_CreateMediaProfileWithAudioSourceConfiguration)

Test Purpose: To verify that Client is able to create media profile with audio source configuration using the **CreateProfile** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateProfile** operation with Type = **AudioSource** present.
- Device supports Media2 Audio feature (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateProfile** request message with **tr2:Configuration\tr2:Type** = **AudioSource** and with specified **tr2:Token** element for this Configuration to create profile with audio source configuration on the Device.
2. Device responds with code HTTP 200 OK and **CreateProfileResponse** message.

Test Result:

PASS -

- Client **CreateProfile** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateProfile** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateProfile** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **"AudioSource"** AND
  - [S3] **tr2:Configuration** element with **tr2:Type** value is equal to **"AudioSource"** has **tr2:Token** element AND
- Device response to the **CreateProfile** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:CreateProfileResponse**.

FAIL -

- The Client failed PASS criteria.

6.21.4 CREATE MEDIA PROFILE WITH AUDIO OUTPUT CONFIGURATION USING MEDIA2

Test Label: Create Media2 Profile with Audio Output Configuration

Test Case ID: MEDIA2_AUDIOOUTPUTPROFILECONFIGURATION-2

Feature Under Test: Create Media2 Profile With Audio Output Configuration (Media2_AudioOutputProfileConfiguration_Media2_CreateMediaProfileWithAudioOutputConfiguration)

Test Purpose: To verify that Client is able to create media profile with audio output configuration using the **CreateProfile** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateProfile** operation with Type = **AudioOutput** present.
- Device supports Media2 Audio Output feature (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateProfile** request message with **tr2:Configuration\tr2:Type** = **AudioOutput** and with specified **tr2:Token** element for this Configuration to create profile with audio output configuration on the Device.
2. Device responds with code HTTP 200 OK and **CreateProfileResponse** message.

Test Result:

PASS -

- Client **CreateProfile** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateProfile** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateProfile** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **"AudioOutput"** AND
  - [S3] **tr2:Configuration** element with **tr2:Type** value is equal to **"AudioOutput"** has **tr2:Token** element AND
- Device response to the **CreateProfile** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:CreateProfileResponse**.

FAIL -

- The Client failed PASS criteria.

6.21.5 GET VIDEO SOURCE CONFIGURATIONS USING MEDIA2

Test Label: Video Source Configuration - Get Video Source Configurations

Test Case ID: MEDIA2_VIDEOSOURCECONFIGURATION-1

Feature Under Test: Get Video Source Configurations Using Media2 (Media2_VideoSourceConfiguration_Media2_GetVideoSourceConfigurations)

Test Purpose: To verify that video source configuration provided by Device is received by Client using the **GetVideoSourceConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoSourceConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Video feature for Media2 Service (Media2_Video) OR Device supports Video Source feature for Media2 Service (Media2_VideoSource).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoSourceConfigurations** request message to retrieve an video source configuration or a list of video source configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoSourceConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetVideoSourceConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoSourceConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurations** AND
- Device response on the **GetVideoSourceConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetVideoSourceConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.21.6 CREATE MEDIA PROFILE WITH VIDEO SOURCE CONFIGURATION USING MEDIA2

Test Label: Create Media2 Profile with Video Source Configuration

Test Case ID: MEDIA2_MEDIAPROFILEMANAGEMENT-1

Feature Under Test: Create Media2 Profile with Video Source Configuration (Media2_MediaProfileManagement_Media2_CreateMediaProfileWithVideoSourceConfiguration)

Test Purpose: To verify that Client is able to create media profile with video source configuration using the **CreateProfile** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateProfile** operation with Type = **VideoSource** present.
- Device supports Media2 Video feature (Media2_Video).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateProfile** request message with **tr2:Configuration\tr2:Type** = **VideoSource** and with specified **tr2:Token** element for this Configuration to create profile with video source configuration on the Device.
2. Device responds with code HTTP 200 OK and **CreateProfileResponse** message.

Test Result:

PASS -

- Client **CreateProfile** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateProfile** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:CreateProfile** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **"VideoSource"** AND
  - [S3] **tr2:Configuration** element with **tr2:Type** value is equal to **"VideoSource"** has **tr2:Token** element AND
- Device response to the **CreateProfile** request fulfills the following requirements:
  - [S4] It has HTTP 200 response code AND

- [S5] **soapenv:Body** element has child element **tr2:CreateProfileResponse**.

FAIL -

- The Client failed PASS criteria.

6.21.7 GET VIDEO ENCODER INSTANCES USING MEDIA2

Test Label: Get Video Encoder Instances

Test Case ID: MEDIA2_MEDIAPROFILEMANAGEMENT-2

Feature Under Test: Get Video Encoder Instances (Media2_MediaProfileManagement_Media2_GetVideoEncoderInstances)

Test Purpose: To verify that list of video encoder instances is received by Client using the **GetVideoEncoderInstances** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetVideoEncoderInstances** operation present.
- Device supports Media2 Video feature (Media2_Video).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetVideoEncoderInstances** request message to retrieve a list of video encoder instances from the Device.
2. Device responds with code HTTP 200 OK and **GetVideoEncoderInstancesResponse** message.

Test Result:

PASS -

- Client **GetVideoEncoderInstances** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetVideoEncoderInstances** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetVideoEncoderInstances** AND
- Device response to the **GetVideoEncoderInstances** request fulfills the following requirements:

- [S2] It has HTTP 200 response code AND
- [S3] **soapenv:Body** element has child element **tr2:GetVideoEncoderInstancesResponse**.

FAIL -

- The Client failed PASS criteria.

6.22 HTTPS Streaming Using Media2 Test Cases

6.22.1 Feature Level Requirement:

Validated Feature: HTTPS Streaming Using Media2 (Media2_HTTPSStreaming)

Check Condition based on Device Features: No (ONVIF Profile T Simulator is used as device).

Required Number of Devices: 1

Profile T Requirement: Conditional

6.22.2 Expected Scenarios Under Test:

1. Client connects to ONVIF Profile T Simulator to initiate HTTPS H264 video Streaming.
2. Client is considered as supporting HTTPS Streaming if the following conditions are met:
   - ONVIF Profile T Simulator detects **Streaming over RTP/RTSP/HTTPS/TCP** feature as supported.
3. Client is considered as NOT supporting HTTPS Streaming if the following is TRUE:
   - ONVIF Profile T Simulator detects **Streaming over RTP/RTSP/HTTPS/TCP** feature as not supported.

6.23 Focus Move Capabilities Test Cases

6.23.1 Feature Level Requirement:

Validated Feature: Focus Move Capabilities (GetMoveOptions)

Check Condition based on Device Features: Imaging Service is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.23.2 Expected Scenarios Under Test:

1. Client connects to Device to get focus move capabilities.
2. Client is considered as supporting Focus Move Capabilities if the following conditions are met:
   - Client is able to retrieve a focus move options using **GetMoveOptions** operation AND
3. Client is considered as NOT supporting Focus Move Capabilities if ANY of the following is TRUE:
   - No valid responses for **GetMoveOptions** request OR

6.23.3 GET FOCUS MOVE OPTIONS

Test Label: Get Move Options

Test Case ID: GETMOVEOPTIONS-1

Feature Under Test: Get Move Options (GetMoveOptions_GetFocusMoveOptions)

Test Purpose: To verify that Client is able retrive focus move capabilities from Device using the **GetMoveOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetMoveOptions** operation present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMoveOptions** request message to retrieve focus move options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetMoveOptionsResponse** message.

Test Result:

PASS -

- Client **GetMoveOptions** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **GetMoveOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:GetMoveOptions** AND
- Device response on the **GetMoveOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **timg:GetMoveOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.24 Focus Control Test Cases

6.24.1 Feature Level Requirement:

Validated Feature: Focus Control (FocusControl)

Check Condition based on Device Features: Focus Control is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.24.2 Expected Scenarios Under Test:

1. Client connects to Device to control focus.
2. Client is considered as supporting Focus Control if the following conditions are met:
   - Client supports get_move_options feature AND
   - Client is able to invoke Absolute OR Relative OR Continuous focus move using **Move** operation AND
   - If Client is able to invoke Continuous focus move Client is able to invoke stop focus move using **Stop** operation.
3. Client is considered as NOT supporting Focus Control if ANY of the following is TRUE:
   - Client does not support get_move_options feature OR
   - No valid responses for **Move** request OR
   - **Move** request contains settings which does not correspong to **GetMoveOptions** message for the same video source token OR

- No valid responses for **Stop** request if **Stop** request is supported by the Client OR
- **Stop** request is not supported, in the case Continuous focus move is supported by the Client.

6.24.3 ABSOLUTE FOCUS MOVE

Test Label: Focus Control - Absolute Focus Move

Test Case ID: FOCUSCONTROL-1

Feature Under Test: Absolute Focus Move (FocusControl_AbsoluteFocusMove)

Test Purpose: To verify that Client is able retrive absolute focus move on Device using the **Move** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Move** operation with **tt:Absolute** element present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMoveOptions** request message to retrieve focus move options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetMoveOptionsResponse** message.
3. If **GetMoveOptionsResponse** message contains **tt:Absolute** element Client invokes **Move** request message for specified video source with **tt:Absolute** element with parameters which are correspond to the resieved focus move options to start absolute focus movement on the Device.
4. Device responds with code HTTP 200 OK and **MoveResponse** message.

Test Result:

PASS -

- Client **Move** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Move** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:Move** AND

- [S2] It contains **timg:Focus/tt:Absolute** element AND

- Device response on the **Move** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **timg:MoveResponse** AND

- There is a Client **GetMoveOptions** request in Test Procedure fulfills the following requirements:
  - [S5] It invoked for the same Device as for the Client **Move** request AND
  - [S6] It invoked before the Client **Move** request AND
  - [S7] **timg:VideoSourceToken** element value is equal to **timg:VideoSourceToken** element from the **Move** request AND

- Device response on the **GetMoveOptions** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **timg:GetMoveOptionsResponse** AND
  - [S10] It contains **timg:MoveOptions\tt:Absolute** element AND

- Settings from the **Move** request corresponds options recieved in the **GetMoveOptionsResponse** message:
  - [S11] **timg:Focus/tt:Absolute/tt:Position** element value from the **Move** request is less or equal to **timg:MoveOptions/tt:Absolute/tt:Position/tt:Max** from the the **GetMoveOptionsResponse** message AND
  - [S12] **timg:Focus/tt:Absolute/tt:Position** element value from the **Move** request is greater or equal to **timg:MoveOptions/tt:Absolute/tt:Position/tt:Min** from the the **GetMoveOptionsResponse** message AND
  - [S13] IF the **Move** request contains **timg:Focus/tt:Absolute/tt:Speed** element THEN:
    - The **GetMoveOptionsResponse** message contains **timg:MoveOptions/tt:Absolute/tt:Speed** element AND
    - **timg:Focus/tt:Absolute/tt:Speed** element value from the **Move** request is less or equal to **timg:MoveOptions/tt:Absolute/tt:Speed/tt:Max** from the the **GetMoveOptionsResponse** message AND
    - **timg:Focus/tt:Absolute/tt:Speed** element value from the **Move** request is greater or equal to **timg:MoveOptions/tt:Absolute/tt:Speed/tt:Min** from the the **GetMoveOptionsResponse** message.

FAIL -

- The Client failed PASS criteria.

6.24.4 RELATIVE FOCUS MOVE

Test Label: Focus Control - Relative Focus Move

Test Case ID: FOCUSCONTROL-2

Feature Under Test: Relative Focus Move (FocusControl_RelativeFocusMove)

Test Purpose: To verify that Client is able retrive relative focus move on Device using the **Move** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Move** operation with **tt:Relative** element present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMoveOptions** request message to retrieve focus move options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetMoveOptionsResponse** message.
3. If **GetMoveOptionsResponse** message contains **tt:Relative** element Client invokes **Move** request message for specified video source with **tt:Relative** element with parameters which are correspond to the resieved focus move options to start relative focus movement on the Device.
4. Device responds with code HTTP 200 OK and **MoveResponse** message.

Test Result:

PASS -

- Client **Move** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Move** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:Move** AND
  - [S2] It contains **timg:Focus/tt:Relative** element AND
- Device response on the **Move** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **timg:MoveResponse** AND
- There is a Client **GetMoveOptions** request in Test Procedure fulfills the following requirements:
  - [S5] It invoked for the same Device as for the Client **Move** request AND

- [S6] It invoked before the Client **Move** request AND
- [S7] **timg:VideoSourceToken** element value is equal to **timg:VideoSourceToken** element from the **Move** request AND

- Device response on the **GetMoveOptions** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **timg:GetMoveOptionsResponse** AND
  - [S10] It contains **timg:MoveOptions\tt:Relative** element AND

- Settings from the **Move** request corresponds options recieved in the **GetMoveOptionsResponse** message:
  - [S11] **timg:Focus/tt:Relative/tt:Distance** element value from the **Move** request is less or equal to **timg:MoveOptions/tt:Relative/tt:Distance/tt:Max** from the the **GetMoveOptionsResponse** message AND
  - [S12] **timg:Focus/tt:Relative/tt:Distance** element value from the **Move** request is greater or equal to **timg:MoveOptions/tt:Relative/tt:Distance/tt:Min** from the the **GetMoveOptionsResponse** message AND
  - [S13] IF the **Move** request contains **timg:Focus/tt:Relative/tt:Speed** element THEN:
    - The **GetMoveOptionsResponse** message contains **timg:MoveOptions/tt:Relative/tt:Speed** element AND
    - **timg:Focus/tt:Relative/tt:Speed** element value from the **Move** request is less or equal to **timg:MoveOptions/tt:Relative/tt:Speed/tt:Max** from the the **GetMoveOptionsResponse** message AND
    - **timg:Focus/tt:Relative/tt:Speed** element value from the **Move** request is greater or equal to **timg:MoveOptions/tt:Relative/tt:Speed/tt:Min** from the the **GetMoveOptionsResponse** message.

FAIL -

- The Client failed PASS criteria.

6.24.5 CONTINUOUS FOCUS MOVE

Test Label: Focus Control - Continuous Focus Move

Test Case ID: FOCUSCONTROL-3

Feature Under Test: Continuous Focus Move (FocusControl_ContinuousFocusMove)

Test Purpose: To verify that Client is able retrive continuous focus move on Device using the **Move** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Move** operation with **tt:Continuous** element present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetMoveOptions** request message to retrieve focus move options for specified video source from the Device.
2. Device responds with code HTTP 200 OK and **GetMoveOptionsResponse** message.
3. If **GetMoveOptionsResponse** message contains **tt:Continuous** element Client invokes **Move** request message for specified video source with **tt:Relative** element with parameters which are correspond to the resieved focus move options to start continuous focus movement on the Device.
4. Device responds with code HTTP 200 OK and **MoveResponse** message.

Test Result:

PASS -

- Client **Move** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Move** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:Move** AND
  - [S2] It contains **timg:Focus/tt:Continuous** element AND
- Device response on the **Move** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **timg:MoveResponse** AND
- There is a Client **GetMoveOptions** request in Test Procedure fulfills the following requirements:
  - [S5] It invoked for the same Device as for the Client **Move** request AND
  - [S6] It invoked before the Client **Move** request AND
  - [S7] **timg:VideoSourceToken** element value is equal to **timg:VideoSourceToken** element from the **Move** request AND
- Device response on the **GetMoveOptions** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **timg:GetMoveOptionsResponse** AND
  - [S10] It contains **timg:MoveOptions\tt:Continuous** element AND

- Settings from the **Move** request corresponds options recieved in the **GetMoveOptionsResponse** message:
  - [S11] **timg:Focus/tt:Continuous/tt:Speed** element value from the **Move** request is less or equal to **timg:MoveOptions/tt:Continuous/tt:Speed/tt:Max** from the the **GetMoveOptionsResponse** message AND
  - [S12] **timg:Focus/tt:Continuous/tt:Speed** element value from the **Move** request is greater or equal to **timg:MoveOptions/tt:Continuous/tt:Speed/tt:Min** from the the **GetMoveOptionsResponse** message.

FAIL -

- The Client failed PASS criteria.

6.24.6 STOP

Test Label: Focus Control - Stop

Test Case ID: FOCUSCONTROL-4

Feature Under Test: Stop (FocusControl_FocusStop)

Test Purpose: To verify that Client is able retrivefocus move options fromon Device using the **Stop** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Stop** operation present.
- Device supports Imaging Service.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **Stop** request message to stop focus move for specified video source for the Device.
2. Device responds with code HTTP 200 OK and **StopResponse** message.

Test Result:

PASS -

- Client **Stop** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Stop** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **timg:Stop** AND

- Device response on the **Stop** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **timg:StopResponse**.

FAIL -

- The Client failed PASS criteria.

6.25 Tampering Test Cases

6.25.1 Feature Level Normative Reference:

Validated Feature: Tampering Events (Tampering)

Check Condition based on Device Features: None (ONVIF Profile T Simulator is used as device).

Required Number of Devices: 1

Profile T Requirement: Conditional

6.25.2 Expected Scenarios Under Test:

1. Client subscribes to ONVIF Profile T Simulator using **CreatePullPointSubscription** operation to get tampering notifications.
2. Client uses Pull Point event mechanism to retrieve the following notification events from ONVIF Profile T Simulator:
   - tns1:VideoSource/ImageTooBlurry/AnalyticsService
   - tns1:VideoSource/ImageTooBlurry/ImagingService
   - tns1:VideoSource/ImageTooDark/AnalyticsService
   - tns1:VideoSource/ImageTooDark/ImagingService
   - tns1:VideoSource/ImageTooBright/AnalyticsService
   - tns1:VideoSource/ImageTooBright/ImagingService
   - tns1:VideoSource/GlobalSceneChange/AnalyticsService
   - tns1:VideoSource/GlobalSceneChange/ImagingService
3. Client is considered as supporting Tampering if the following conditions are met:

- ONVIF Profile T Simulator detects **Tampering** feature as supported.

4. Client is considered as NOT supporting Tampering if ANY of the following is TRUE:
   - ONVIF Profile T Simulator detects **Tampering** feature as not supported.

6.26 PTZ - Listing Test Cases

6.26.1 Feature Level Requirement:

Validated Feature: PTZ Listing (PtzListing)

Check Condition based on Device Features: PTZ Service is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Conditional

6.26.2 Expected Scenarios Under Test:

1. Client connects to Device to read PTZ capabilities.
2. Client is considered as supporting PTZ - Listing if the following conditions are met:
   - Client is able to read PTZ capabilities from PTZ Node using EITHER GetNodes OR GetNode operations.
3. Client is considered as NOT supporting PTZ - Listing if ANY of the following is TRUE:
   - No Valid Device Response to GetNodes request AND
   - No Valid Device Response to GetNode request.

6.26.3 GET NODES

Test Label: PTZ Listing - GetNodes

Test Case ID: PTZLISTING-1

Feature Under Test: Get Nodes (PtzListing_GetNodes)

Test Purpose: To verify that list of all existing PTZ capabilities from Device is received by Client using the GetNodes operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetNodes operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetNodes request message to retrieve complete PTZ capabilities list from Device.
2. Device responds with code HTTP 200 OK and GetNodesResponse message.

Test Result:

PASS -

- Client **GetNodes** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetNodes** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNodes>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<GetNodesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.26.4 GET NODE

Test Label: PTZ Listing - GetNode

Test Case ID: PTZLISTING-2

Feature Under Test: Get Node (PtzListing_GetNode)

Test Purpose: To verify that Client is able to retrieve a specific PTZ capability properties from Device using the GetNode operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetNode operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetNode request message to retrieve a specific PTZ capability properties from Device.
2. Device responds with code HTTP 200 OK and GetNodeResponse message.

Test Result:

PASS -

- Client **GetNode** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetNode** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetNode>" tag after the "<Body>" tag AND
  - [S2] "<GetNode>" includes tag: "<NodeToken>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<GetNodeResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.27 PTZ Presets Test Cases

6.27.1 Feature Level Requirement:

Validated Feature: PTZ Presets (PtzPresets)

Check Condition based on Device Features: PTZ Presets is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Conditional

6.27.2 Expected Scenarios Under Test:

1. Client connects to Device to manage the presets of a PTZ Node.
2. Client is considered as supporting PTZ Presets if the following conditions are met:

- Client is able to list the presets using the GetPresets operation AND
- Client is able to move a PTZ Device to a specific preset using the GotoPreset operation.

3. Client is considered as NOT supporting PTZ Presets if ANY of the following is TRUE:
   - No Valid Device Response to GetPresets request OR
   - No Valid Device Response to GotoPreset request.

6.27.3 PTZ GET PRESETS

Test Label: PTZ Presets - GetPresets

Test Case ID: PTZPRESETS-1

Feature Under Test: Get Presets (PtzPresets_PtzGetPresets)

Test Purpose: To verify that Client is able to list the presets using the GetPresets operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetPresets operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetPresets request message to list the available presets from Device.
2. Device responds with code HTTP 200 OK and GetPresetsResponse message.

Test Result:

PASS -

- Client **GetPresets** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetPresets** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GetPresets>" tag after the "<Body>" tag AND
  - [S2] "<GetPresets>" includes tag: "<ProfileToken>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<GetPresetsResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.27.4 PTZ GOTO PRESET

Test Label: PTZ Presets - GotoPreset

Test Case ID: PTZPRESETS-2

Feature Under Test: Goto Preset (PtzPresets_PtzGotoPreset)

Test Purpose: To verify that Client is able to move a PTZ Device to a specific preset using the GotoPreset operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GotoPreset operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GotoPreset request message to move PTZ Device to specific preset.
2. Device responds with code HTTP 200 OK and GotoPresetResponse message.

Test Result:

PASS -

- Client **GotoPreset** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GotoPreset** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GotoPreset>" tag after the "<Body>" tag AND
  - [S2] "<GotoPreset>" includes tag: "<ProfileToken>" with non-empty string value of specific token AND
  - [S3] "<GotoPreset>" includes tag: "<PresetToken>" with non-empty string value of specific token AND
  - [S4] Device response contains "HTTP/* 200 OK" AND
  - [S5] Device response contains "<GotoPresetResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.28 PTZ Home Position Test Cases

6.28.1 Feature Level Requirement:

Validated Feature: PTZ Home Position (PtzHomePosition)

Check Condition based on Device Features: PTZ Home Position is supported by Device.

Required Number of Devices: 1

Profile S Requirement: Conditional

Profile T Requirement: Conditional

6.28.2 Expected Scenarios Under Test:

1. Client connects to Device to manage the home position of a PTZ Node.
2. Client is considered as supporting PTZ Home Position if the following conditions are met:
   - Client is able to move PTZ Device to its home position using the GotoHomePosition operation
3. Client is considered as NOT supporting PTZ Home Position if ANY of the following is TRUE:
   - No Valid Device Response to GotoHomePosition request.

6.28.3 PTZ HOME POSITION

Test Label: PTZ Presets - GotoHomePosition

Test Case ID: PTZHOMEPOSITION-1

Feature Under Test: Goto Home Position (PtzHomePosition_PtzGotoHomePosition)

Test Purpose: To verify that Client is able to move PTZ Device to its home position using the GotoHomePosition operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GotoHomePosition operation present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GotoHomePosition request message to move PTZ Device to its home position.
2. Device responds with code HTTP 200 OK and GotoHomeResponse message.

Test Result:

PASS -

- Client **GotoHomePosition** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GotoHomePosition** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<GotoHomePosition>" tag after the "<Body>" tag AND
  - [S2] "<GotoHomePosition>" includes tag: "<ProfileToken>" with non-empty string value of specific token AND
  - [S3] Device response contains "HTTP/* 200 OK" AND
  - [S4] Device response contains "<GotoHomePositionResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.29 PTZ - Set Preset Test Cases

6.29.1 Feature Level Requirement:

Validated Feature: PTZ - Set Preset (PtzSetPreset)

Check Condition based on Device Features: PTZ Presets is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.29.2 Expected Scenarios Under Test:

1. Client connects to Device to store a preset using the **SetPreset** operation.
2. Client is considered as supporting PTZ - SetPreset if the following conditions are met:
   - Client is able to store a preset using the **SetPreset** operation.

3. Client is considered as NOT supporting PTZ - SetPreset if ANY of the following is TRUE:
    - No Valid Device Response to **SetPreset** request.

6.29.3 PTZ SET PRESET

Test Label: PTZ SetPreset

Test Case ID: PTZSETPRESET-1

Feature Under Test: Set Preset (PtzSetPreset_PtzSetPresetRequest)

Test Purpose: To verify that Client is able to to store a preset using the **SetPreset** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with SetPreset operation present.
- Device supports PTZ Presets (PTZPresets).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetPreset request message to store a preset on the Device.
2. Device responds with code HTTP 200 OK and SetPresetResponse message.

Test Result:

PASS -

- Client **SetPreset** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetPreset** request in Test Procedure fulfills the following requirements:
    - [S1] **soapenv:Body** element has child element **tptz:SetPreset** AND
- Device response on the **SetPreset** request fulfills the following requirements:
    - [S2] It has HTTP 200 response code AND
    - [S3] **soapenv:Body** element has child element **tptz:SetPresetResponse**.

FAIL -

- The Client failed PASS criteria.

6.30 PTZ Get Compatible Configurations Test Cases

6.30.1 Feature Level Requirement:

Validated Feature: Get Compatible Configurations (PtzGetCompatibleConfigurations)

Check Condition based on Device Features: GetCompatibleConfigurations (PTZ Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.30.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve PTZ configurations compatible with profile using the **GetCompatibleConfigurations** operation.
2. Client is considered as supporting PTZ Get Compatible Configurations if the following conditions are met:
   - Client is able to retrieve PTZ configurations compatible with profile using the **GetCompatibleConfigurations** operation.
3. Client is considered as NOT supporting PTZ Get Compatible Configurations if ANY of the following is TRUE:
   - No Valid Device Response to **GetCompatibleConfigurations** request.

6.30.3 PTZ GET COMPATIBLE CONFIGURATIONS

Test Label: PTZ GetCompatibleConfigurations

Test Case ID: PTZGETCOMPATIBLECONFIGURATIONS-1

Feature Under Test: Get Compatible Configurations (PtzGetCompatibleConfigurations_PTZGetCompatibleConfigurationsRequest)

Test Purpose: To verify that Client is able to retrieve PTZ configurations compatible with profile using the **GetCompatibleConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetCompatibleConfigurations operation present.

- Device supports Get Compatible Configurations feature (PTZGetCompatibleConfigurations).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetCompatibleConfigurations request message to get TPZ configurations compatible with profile.
2. Device responds with code HTTP 200 OK and GetCompatibleConfigurationsResponse message.

Test Result:

PASS -

- Client **GetCompatibleConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetCompatibleConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurations** AND
- Device response on the **GetCompatibleConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.31 PTZ Media2 Profile Configuration Test Cases

6.31.1 Feature Level Requirement:

Validated Feature: PTZ Using Media2 Profile Configuration (PTZUsingMedia2ProfileConfiguration)

Check Condition based on Device Features: PTZ Service and Media2 Service are supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.31.2 Expected Scenarios Under Test:

1. Client connects to Device to add compatible ptz configuration to a Media Profile.
2. Client is considered as supporting PTZ Media2 Profile Configuration if the following conditions are met:
   - Client supports **PtzGetCompatibleConfigurations_PTZGetCompatibleConfigurationsRequest** feature (please see PTZGETCOMPATIBLECONFIGURATIONS-1 PTZ GET COMPATIBLE CONFIGURATIONS section) AND
   - Client is able to add compatible ptz configuration using **GetCompatibleConfigurations** operation and **AddConfiguration** operation with Type element value is equal to **PTZ**.
3. Client is considered as NOT supporting PTZ Media2 Profile Configuration if ANY of the following is TRUE:
   - Client does not support **PtzGetCompatibleConfigurations_PTZGetCompatibleConfigurationsRequest** feature (please see PTZGETCOMPATIBLECONFIGURATIONS-1 PTZ GET COMPATIBLE CONFIGURATIONS section) OR
   - Client is unable to add an ptz configuration compatible with profile using **GetCompatibleConfigurations** operation and **AddConfiguration** operation OR
   - No valid responses for **GetCompatibleConfigurations** request OR
   - No valid responses for **AddConfiguration** request with Type element value is equal to **PTZ**.

6.31.3 PTZ GET COMPATIBLE CONFIGURATIONS

Test Label: PTZ GetCompatibleConfigurations

Test Case ID: PTZGETCOMPATIBLECONFIGURATIONS-1

Feature Under Test: Get Compatible Configurations (PtzGetCompatibleConfigurations_PTZGetCompatibleConfigurationsRequest)

Test Purpose: To verify that Client is able to retrieve PTZ configurations compatible with profile using the **GetCompatibleConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetCompatibleConfigurations operation present.
- Device supports Get Compatible Configurations feature (PTZGetCompatibleConfigurations).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes GetCompatibleConfigurations request message to get TPZ configurations compatible with profile.
2. Device responds with code HTTP 200 OK and GetCompatibleConfigurationsResponse message.

Test Result:

PASS -

- Client **GetCompatibleConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetCompatibleConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurations** AND
- Device response on the **GetCompatibleConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

6.31.4 ADD PTZ CONFIGURATION USING MEDIA2

Test Label: Add PTZ Configuration to Media2 Profile

Test Case ID: PTZUSINGMEDIA2PROFILECONFIGURATION-1

Feature Under Test: Add PTZ Configuration in Media2 Profile (PTZUsingMedia2ProfileConfiguration_AddPTZConfigurationToMedia2Profile)

Test Purpose: To verify that Client is able to add an ptz configuration to a media profile using the **AddConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddConfiguration** operation with Type value is equal to **PTZ** present.
- Device supports Media2 Service (Media2Service).
- Device supports PTZ service (PTZService).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetCompatibleConfigurations** request message with specified **ProfileToken** to retrieve compatible ptz configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetCompatibleConfigurationsResponse** message.
3. Client invokes **AddConfiguration** request message with Type element value is equal to PTZ and with Configuration token that was recieved in **GetCompatibleConfigurationsResponse** message for the same media profile to add an ptz configuration to specified media profile on the Device.
4. Device responds with code HTTP 200 OK and **AddConfigurationResponse** message.

Test Result:

PASS -

- Client **AddConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **AddConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:AddConfiguration** AND
  - [S2] It has **tr2:Configuration/tr2:Type** element with value is equal to **PTZ** AND
- Device response to the **AddConfiguration** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tr2:AddConfigurationResponse**.

- There is Client **GetCompatibleConfigurations** request in Test Procedure that fulfills the following requirements:
  - [S5] It is invoked before the Client **AddConfiguration** request AND
  - [S6] It has **tptz:ProfileToken** element with value is equal to **tr2:ProfileToken** element value from the **AddConfiguration** request AND
  - [S7] It is the last **GetCompatibleConfigurations** request which corresponds to [S5], AND [S6] AND
- Device response to the **GetCompatibleConfigurations** request fulfills the following requirements:
  - [S8] It has HTTP 200 response code AND
  - [S9] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurationsResponse** AND
  - [S10] It contains **tptz:PTZConfiguration** element with **@token** attribute value is equal to **tr2:Configuration/tr2:Token** value for Configuration with **tr2:Configuration/tr2:Type** value is equal to PTZ from the **AddConfiguration** request message.

FAIL -

- The Client failed PASS criteria.

6.32 PTZ Set Configuration Test Cases

6.32.1 Feature Level Requirement:

Validated Feature: Set Configuration (PtzSetConfiguration)

Check Condition based on Device Features: PTZ Service is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.32.2 Expected Scenarios Under Test:

1. Client connects to Device to modify a PTZ configuration using the **SetConfiguration** operation.
2. Client is considered as supporting PTZ Set Configuration if the following conditions are met:

- Client is able to modify a PTZ configuration using the **SetConfiguration** operation.

3. Client is considered as NOT supporting PTZ Set Configuration if ANY of the following is TRUE:
   - No Valid Device Response to **SetConfiguration** request.

6.32.3 PTZ SET CONFIGURATION

Test Label: PTZ Set Configuration

Test Case ID: PTZSETCONFIGURATION-1

Feature Under Test: Set Configuration (PtzSetConfiguration_PtzSetConfigurationRequest)

Test Purpose: To verify that Client is able to to modify a PTZ configuration using the **SetConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with SetConfiguration operation present.
- Device supports PTZ Service (PTZService).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes SetConfiguration request message to modify a PTZ configuration on the Device.
2. Device responds with code HTTP 200 OK and SetConfigurationResponse message.

Test Result:

PASS -

- Client **SetConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tptz:SetConfiguration** AND
- Device response on the **SetConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND

- [S3] **soapenv:Body** element has child element **tptz:SetConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

6.33 Relay Outputs Using Device IO Test Cases

6.33.1 Feature Level Normative Reference:

Validated Feature: Relay Outputs (RelayOutputsUsingDeviceIO)

Check Condition based on Device Features: Relay Outputs (Device IO Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.33.2 Expected Scenarios Under Test:

1. Client connects to Device to control of Relay Outputs connected to a device.
2. Client is considered as supporting Relay Outputs if the following conditions are met:
   - Client is able to retrieve available Relay Outputs using **GetRelayOutputs** operation (Device IO Service) AND
   - Client is able to control Relay Output state using **SetRelayOutputState** operation (Device IO Service).
3. Client is considered as NOT supporting Relay Outputs if ANY of the following is TRUE:
   - No valid response to **GetRelayOutputs** request (Device IO Service) OR
   - No valid response to **SetRelayOutputState** request (Device IO Service).

6.33.3 GET RELAY OUTPUTS USING DEVICE IO

Test Label: Relay Output - Get Relay Outputs

Test Case ID: RELAYOUTPUTSUSINGDEVICEIO-1

Feature Under Test: Get Relay Outputs (RelayOutputsUsingDeviceIO_GetRelayOutputsUsingDeviceIO)

Test Purpose: To verify that relay outputs provided by Device is received by Client using the **GetRelayOutputs** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetRelayOutputs** operation for Device IO Service present.
- Device supports RelayOutputs for Device IO Service (RelayOutputs).
- Client supports Capabilities feature.
- The Client Test Tool retrives Device IO Service address from device's response on GetServices or GetCapabilities Client request.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetRelayOutputs** request message to Device IO Service to retrieve relay outputs from the Device.
2. Device responds with code HTTP 200 OK and **GetRelayOutputsResponse** message.

Test Result:

PASS -

- Client **GetRelayOutputs** request messages to Device IO Service are valid according to XML Schemas listed in Namespaces AND
- Client **GetRelayOutputs** request to Device IO Service in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:GetRelayOutputs** AND
- Device response on the **GetRelayOutputs** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:GetRelayOutputsResponse**.

FAIL -

- The Client failed PASS criteria.

6.33.4 SET RELAY OUTPUT STATE USING DEVICE IO

Test Label: Relay Output - Set Relay Output State

Test Case ID: RELAYOUTPUTSUSINGDEVICEIO-2

Feature Under Test: Set Relay Output (RelayOutputsUsingDeviceIO_SetRelayOutputStateUsingDeviceIO)

Test Purpose: To verify that Client is able to trigger a relay output using the **SetRelayOutputState** operation.

Pre-Requisite:

- Device supports RelayOutputs for Device IO Service (RelayOutputs).
- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetRelayOutputState** operation for Device IO Service present.
- Client supports Capabilities feature.
- The Client Test Tool retrives Device IO Service address from device's response on GetServices or GetCapabilities Client request.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetRelayOutputState** request message to Device IO Service to trigger a relay output on the Device.
2. Device responds with code HTTP 200 OK and **SetRelayOutputStateResponse** message.

Test Result:

PASS -

- Client **SetRelayOutputState** request messages to Device IO Service are valid according to XML Schemas listed in Namespaces AND
- Client **SetRelayOutputState** request to Device IO Service in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tds:SetRelayOutputState** AND
- Device response on the **SetRelayOutputState** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tds:SetRelayOutputStateResponse**.

FAIL -

- The Client failed PASS criteria.

6.34 Get Digital Inputs Test Cases

6.34.1 Feature Level Normative Reference:

Validated Feature: Get Digital Inputs (GetDigitalInputs)

Check Condition based on Device Features: Digital Inputs (Device IO Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.34.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve of Digital Inputs connected to a device using **GetDigitalInputs** operation.
2. Client subscribes to device messages using **CreatePullPointSubscription** operation to get **tns1:Device/Trigger/DigitalInput** notifications.
3. Client uses Pull Point event mechanism to retrieve notification events from Device.
4. Client is considered as supporting Digital Inputs if the following conditions are met:
   - Client is able to retrieve available Digital Inputs using **GetDigitalInputs** operation AND
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to monitor the state of the input pins using **tns1:Device/Trigger/DigitalInput** event if device supports DigitalInputs feature.
5. Client is considered as NOT supporting Digital Inputs if ANY of the following is TRUE:
   - No valid response to **GetDigitalInputs** request OR
   - Client does not support EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - Client does not support **tns1:Device/Trigger/DigitalInput** event.

6.34.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Test Purpose: To verify that the Client is able to retrieve events using Pull Point.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Pull Point event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes PullMessages command with Timeout and MessageLimit elements.
4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.34.4 GET DIGITAL INPUTS

Test Label: Get Digital Inputs

Test Case ID: GETDIGITALINPUTS-1

Feature Under Test: Get Digital Inputs (GetDigitalInputs_GetDigitalInputsRequest)

Test Purpose: To verify that digital inputs provided by Device is received by Client using the **GetDigitalInputs** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetDigitalInputs** operation present.
- Device supports Digital Inputs feature (DigitalInputs).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetDigitalInputs** request message to retrieve digital inputs from the Device.
2. Device responds with code HTTP 200 OK and **GetDigitalInputsResponse** message.

Test Result:

PASS -

- Client **GetDigitalInputs** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetDigitalInputs** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tmd:GetDigitalInputs** AND
- Device response on the **GetDigitalInputs** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tmd:GetDigitalInputsResponse**.

FAIL -

- The Client failed PASS criteria.

6.35 Get Supported Rules Test Cases

6.35.1 Feature Level Requirement:

Validated Feature: Get Supported Rules (GetSupportedRules)

Check Condition based on Device Features: Media 2 Service and Rule Engine is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.35.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve supported Rules using the **GetSupportedRules** operation.
2. Client is considered as supporting Get Supported Rules if the following conditions are met:
   - Client is able to retrieve supported Rules using the **GetSupportedRules** operation.
3. Client is considered as NOT supporting Get Supported Rules if ANY of the following is TRUE:
   - No valid device response to **GetSupportedRules** request.

6.35.3 GET SUPPORTED RULES

Test Label: Get Supported Rules

Test Case ID: GETSUPPORTEDRULES-1

Feature Under Test: Get Supported Rules (GetSupportedRules_GetSupportedRulesRequest)

Test Purpose: To verify that Client is able to retrieve supported rules using the **GetSupportedRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetSupportedRules operation present.
- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetSupportedRules** request message to retrieve supported Rules from the Device.
2. Device responds with code HTTP 200 OK and **GetSupportedRulesResponse** message.

Test Result:

PASS -

- Client **GetSupportedRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetSupportedRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:GetSupportedRules** AND
- Device response on the **GetSupportedRules** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **axt:GetSupportedRulesResponse**.

FAIL -

- The Client failed PASS criteria.

6.36 Get Rules Test Cases

6.36.1 Feature Level Requirement:

Validated Feature: Get Rules (GetRules)

Check Condition based on Device Features: Media 2 Service and Rule Engine is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.36.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve available Rules using the **GetRules** operation.
2. Client is considered as supporting Get Rules if the following conditions are met:

- Client is able to retrieve available Rules using the **GetRules** operation.

3. Client is considered as NOT supporting Get Rules if ANY of the following is TRUE:
   - No valid device response to **GetRules** request.

6.36.3 GET RULES

Test Label: Get Rules

Test Case ID: GETRULES-1

Feature Under Test: Get Rules (GetRules_GetRulesRequest)

Test Purpose: To verify that Client is able to retrieve available rules using the **GetRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetRules operation present.
- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetRules** request message to retrieve available Rules from the Device.
2. Device responds with code HTTP 200 OK and **GetRulesResponse** message.

Test Result:

PASS -

- Client **GetRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:GetRules** AND
- Device response on the **GetRules** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **axt:GetRulesResponse**.

FAIL -

- The Client failed PASS criteria.

6.37 Motion Detection Test Cases

6.37.1 Feature Level Requirement:

Validated Feature: Motion Detection (MotionDetection)

Check Condition based on Device Features: Media 2 Service and Rule Engine and Rule Options and Motion Region Detector Rule and Pull Point Notification is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Conditional

6.37.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve available Rules using the **GetRules** and **GetSupportedRules** operations.
2. Client creates Rules of type **tt:MotionRegionDetector** using the **GetRuleOptions** and **CreateRules** operations.
3. Client deletes Rules of type **tt:MotionRegionDetector** using the **DeleteRules** operations.
4. Client subscribes to device messages using **CreatePullPointSubscription** operation to get **Motion Region Detector** notifications.
5. Client is considered as supporting Motion Detection if the following conditions are met:
   - Client supports **GetSupportedRules_GetSupportedRulesRequest** feature (please see GETSUPPORTEDRULES-1 GET SUPPORTED RULES section) AND
   - Client supports **GetRules_GetRulesRequest** (please see GETRULES-1 GET RULES section) feature AND
   - Client supports **EventHandling_Pullpoint** feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to retrieve options of Rules of type **tt:MotionRegionDetector** using **GetRuleOptions** operation AND
   - Client is able to create Rules of type **tt:MotionRegionDetector** using **CreateRules** operation AND

- Client is able to delete Rules of type **tt:MotionRegionDetector** using **DeleteRules** operation AND
- Client is able to retrieve **tns1:RuleEngine/MotionRegionDetector/Motion** notifications about Motion Region Detector if device supports MotionRegionDetector Rule.

6. Client is considered as NOT supporting Motion Detection if ANY of the following is TRUE:
   - Client does not support **GetSupportedRules_GetSupportedRulesRequest** feature (please see GETSUPPORTEDRULES-1 GET SUPPORTED RULES section) OR
   - Client does not support **GetRules_GetRulesRequest** (please see GETRULES-1 GET RULES section) feature OR
   - Client does not support **EventHandling_Pullpoint** feature (please see EVENTHANDLING-1 PULLPOINT section) OR
   - No valid device response to **GetRuleOptions** request with RuleType value is equal to **tt:MotionRegionDetector** OR
   - No valid device response to **CreateRules** request with Rule Type value is equal to **tt:MotionRegionDetector** OR
   - No valid device response to **DeleteRules** request with Rule Type value is equal to **tt:MotionRegionDetector** OR
   - Client is not able to retrieve **tns1:RuleEngine/MotionRegionDetector/Motion** notifications about Motion Region Detector if device supports MotionRegionDetector Rule.

6.37.3 GET SUPPORTED RULES

Test Label: Get Supported Rules

Test Case ID: GETSUPPORTEDRULES-1

Feature Under Test: Get Supported Rules (GetSupportedRules_GetSupportedRulesRequest)

Test Purpose: To verify that Client is able to retrieve supported rules using the **GetSupportedRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetSupportedRules operation present.

- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetSupportedRules** request message to retrieve supported Rules from the Device.
2. Device responds with code HTTP 200 OK and **GetSupportedRulesResponse** message.

Test Result:

PASS -

- Client **GetSupportedRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetSupportedRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:GetSupportedRules** AND
- Device response on the **GetSupportedRules** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **axt:GetSupportedRulesResponse**.

FAIL -

- The Client failed PASS criteria.

6.37.4 GET RULES

Test Label: Get Rules

Test Case ID: GETRULES-1

Feature Under Test: Get Rules (GetRules_GetRulesRequest)

Test Purpose: To verify that Client is able to retrieve available rules using the **GetRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetRules operation present.
- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetRules** request message to retrieve available Rules from the Device.
2. Device responds with code HTTP 200 OK and **GetRulesResponse** message.

Test Result:

PASS -

- Client **GetRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:GetRules** AND
- Device response on the **GetRules** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **axt:GetRulesResponse**.

FAIL -

- The Client failed PASS criteria.

6.37.5 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Test Purpose: To verify that the Client is able to retrieve events using Pull Point.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Pull Point event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes PullMessages command with Timeout and MessageLimit elements.

4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

6.37.6 GET MOTION REGION DETECTOR RULE OPTIONS

Test Label: Get Rule Options

Test Case ID: MOTIONDETECTION-1

Feature Under Test: Get Rule Options (MotionDetection_GetRuleOptions)

Test Purpose: To verify that Client is able to retrieve MotionRegionDetector rule options using the **GetRuleOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with GetRuleOptions operation with RuleType value is equal to **tt:MotionRegionDetector** present.
- Device supports Rule Options (RuleOptions).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetRuleOptions** request message with RuleType value is equal to **tt:MotionRegionDetector** to retrieve motion region detector rule options from the Device.
2. Device responds with code HTTP 200 OK and **GetRuleOptions** message.

Test Result:

PASS -

- Client **GetRuleOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetRuleOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:GetRuleOptions** AND
  - [S2] **axt:GetRuleOptions** element has child element **axt:RuleType** AND
  - [S3] **axt:GetRuleOptions/axt:RuleType** element value is equal to **tt:MotionRegionDetector** AND
- Device response on the **GetRuleOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **axt:GetRuleOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

6.37.7 CREATE MOTION REGION DETECTOR RULE

Test Label: Create Rules

Test Case ID: MOTIONDETECTION-2

Feature Under Test: Create Motion Region Detector Rule (MotionDetection_CreateRules)

Test Purpose: To verify that Client is able to create MotionRegionDetector rule using the **CreateRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with **CreateRules** operation with Rule Type value is equal to **tt:MotionRegionDetector** present.
- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreateRules** request message with Rule element that has Type value is equal to **tt:MotionRegionDetector** to create motion region detector rule on the Device.
2. Device responds with code HTTP 200 OK and **CreateRules** message.

Test Result:

PASS -

- Client **CreateRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreateRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:CreateRules** AND
  - [S2] **axt:CreateRules** element has child element **axt:Rule** with **@Type** = **tt:MotionRegionDetector** AND
- Device response on the **CreateRules** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **axt:CreateRulesResponse**.

FAIL -

- The Client failed PASS criteria.

6.37.8 DELETE MOTION REGION DETECTOR RULE

Test Label: Delete Rules

Test Case ID: MOTIONDETECTION-3

Feature Under Test: Delete Rules (MotionDetection_DeleteRules)

Feature Under Test: Delete Rules

Test Purpose: To verify that Client is able to delete MotionRegionDetector rule using the **DeleteRules** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one conversation between Client and Device with **DeleteRules** operation present.
- Device supports Rule Engine (RuleEngine).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **DeleteRules** request message with RuleName that corresponds to Rule with **tt:MotionRegionDetector** Type to delete motion region detector rule on the Device.
2. Device responds with code HTTP 200 OK and **DeleteRules** message.

Test Result:

PASS -

- Client **DeleteRules** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **DeleteRules** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **axt:DeleteRules** AND
  - [S2] **axt:DeleteRules** contains **tr2:RuleName** that corresponds to rule with **tt:MotionRegionDetector** type (see Annex A.2) AND
- Device response on the **DeleteRules** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **axt:DeleteRulesResponse**.

FAIL -

- The Client failed PASS criteria.

7 Test Cases for Profile Optional Features

7.1 Unsubscribe Test Cases

Validated Feature: Unsubscribe (Unsubscribe)

Check Condition based on Device Features: Pull Point Notification OR WS-Basic Notification is supported by Device.

Required Number of Devices: 1

Profile A Requirement: Optional

Profile C Requirement: Optional

Profile S Requirement: Optional

Profile G Requirement: Optional

Profile T Requirement: Optional

7.1.1 Expected Scenarios Under Test:

1. Client connects to Device to Unsubscribe subscribtions.
2. Client is considered as supporting Unsubscribe if the following conditions are met:
   - Client is able to unsubscribe subscribtions using **Unsubscribe** operation.
3. Client is considered as NOT supporting Unsubscribe if the following is TRUE:
   - No valid responses for **Unsubscribe** request OR
   - **Unsubscribe** request does not contains valid **wsa:Action** header.

7.1.2 UNSUBSCRIBE

Test Label: Unsubscribe - Unsubscribe

Test Case ID: UNSUBSCRIBE-1

Feature Under Test: Unsubscribe (Unsubscribe_UnsubscribeAction)

Test Purpose: To verify that the Client is able to use **Unsubscribe** operation to terminate a subscribtion.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Unsubscribe** operations present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **Unsubscribe** message with valid **wsa:Action** header to terminete a subscription.
2. Device responses with code HTTP 200 OK and **UnsubscribeResponse** message.

Test Result:

PASS -

- Client **Unsubscribe** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Unsubscribe** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **wsnt:Unsubscribe** AND
  - [S2] It contains **wsa:Action** element in header equal to "http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest" AND
- Device response on the **Unsubscribe** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **wsnt:UnsubscribeResponse**

FAIL -

- The Client failed PASS criteria.

7.2 Keep Alive for Pull Point Event Handling Test Cases

7.2.1 Feature Level Requirement:

Validated Feature: Keep Alive for Pull Point Event Handling (KeepAliveForPullPointEventHandling)

Check Condition based on Device Features: Pull Point Notification is supported by Device.

Required Number of Devices: 3

Profile A Requirement: Mandatory

Profile C Requirement: Mandatory

Profile S Requirement: Conditional

Profile G Requirement: Conditional

Profile T Requirement: Optional

7.2.2 Expected Scenarios Under Test:

1. Client connects to Device to initiate Pull Point Event Handling.
2. Client is considered as supporting Keep Alive for Pull Point Event Handling if the following conditions are met:
   - Client supports EventHandling_Pullpoint feature (please see EVENTHANDLING-1 PULLPOINT section) AND
   - Client is able to renew pull point subscribtion using **Renew** operation OR **PullMessages** operation mechanism.
3. Client is considered as NOT supporting Keep Alive for Pull Point Event Handling if the following is TRUE:
   - No valid responses for **Renew** request AND for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive OR
   - No valid responses for **Renew** request if detected OR
   - No valid responses for **CreatePullPointSubscription** request in the case if **PullMessages** used for keep alive if detected OR
   - **Renew** request was invoked to address which was not specified in **tev:SubscriptionReference\wsa:Address** element of corresponding **CreatePullPointSubscriptionResponse** message.

7.2.3 PULLPOINT

Test Label: Event Handling - Pull Point

Test Case ID: EVENTHANDLING-1

Feature Under Test: Pull Point (EventHandling_PullPoint)

Test Purpose: To verify that the Client is able to retrieve events using Pull Point.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with Pull Point event type.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes CreatePullPointSubscription message.
2. Device responds with code HTTP 200 OK and CreatePullPointSubscriptionResponse message.
3. Client invokes PullMessages command with Timeout and MessageLimit elements.
4. Device responds with code HTTP 200 OK and PullMessagesResponse message.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:
  - [S1] Client request contains "<CreatePullPointSubscription>" tag after the "<Body>" tag AND
  - [S2] Device response contains "HTTP/* 200 OK" AND
  - [S3] Device response contains "<CreatePullPointSubscriptionResponse>" tag AND
- Client **PullMessages** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **PullMessages** request in Test Procedure fulfills the following requirements:
  - [S4] Client request contains "<PullMessages>" tag after the "<Body>" tag AND
  - [S7] Device response contains "HTTP/* 200 OK" AND
  - [S8] Device response contains "<PullMessagesResponse>" tag.

FAIL -

- The Client failed PASS criteria.

7.2.4 RENEW

Test Label: Advanced Pull Point Event Handling - Renew

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-1

Feature Under Test: Renew (KeepAliveForPullPointEventHandling_Renew)

Test Purpose: To verify that the Client is able to use **Renew** operation as keep alive for Pull Point subscribtion.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **Renew** operations present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreatePullPointSubscription** message.
2. Device responds with code HTTP 200 OK and **CreatePullPointSubscriptionResponse** message.
3. Client invokes **Renew** message to valid address recieved in **CreatePullPointSubscriptionResponse** message for the created Pull Point subscribtion with valid address recieved in **CreatePullPointSubscriptionResponse** message.
4. Device responds with code HTTP 200 OK and **RenewResponse** message.

Test Result:

PASS -

- Client **Renew** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **Renew** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **wsnt:Renew** AND
- Device response on the **Renew** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **wsnt:RenewResponse** AND
- There is a Device response on the **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:

- [S4] It has HTTP 200 response code AND
- [S5] It received for the same Device as for the Client **Renew** request AND
- [S6] It received before the Client **Renew** request AND
- [S7] It contains **tev:SubscriptionReference\wsa:Address** element which is equal to HTTP address that was used to send the **Renew** request.

FAIL -

- The Client failed PASS criteria.

7.2.5 PULL MESSAGES AS KEEP ALIVE

Test Label: Advanced Pull Point Event Handling - Pull Messages as Keep Alive

Test Case ID: KEEPALIVEFORPULLPOINTEVENTHANDLING-2

Feature Under Test: Pull Messages as Keep Alive (KeepAliveForPullPointEventHandling_PullMessagesAsKeepAlive)

Test Purpose: To verify that the Client is able to use **PullMessages** operation as keep alive for Pull Point subscribtion.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreatePullPointSubscription** operations whithout **tev:InitialTerminationTime** element present.

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **CreatePullPointSubscription** message.
2. Device responds with code HTTP 200 OK and **CreatePullPointSubscriptionResponse** message whithout **tev:InitialTerminationTime** element.

Test Result:

PASS -

- Client **CreatePullPointSubscription** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **CreatePullPointSubscription** request in Test Procedure fulfills the following requirements:

- [S1] **soapenv:Body** element has child element **tev:CreatePullPointSubscription** AND
- [S2] It does not contain **tev:InitialTerminationTime** element AND

- Device response on the **CreatePullPointSubscription** request fulfills the following requirements:
  - [S3] It has HTTP 200 response code AND
  - [S4] **soapenv:Body** element has child element **tev:CreatePullPointSubscriptionResponse**.

FAIL -

- The Client failed PASS criteria.

7.3 Audio Source Configuration Using Media2 Test Cases

7.3.1 Feature Level Normative Reference:

Validated Feature: Audio Source Configuration Using Media2 (Media2_AudioSourceConfiguration)

Check Condition based on Device Features: Audio (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Optional

7.3.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Audio Source Configuration.
2. Client is considered as supporting Audio Source Configuration if the following conditions are met:
   - Client is able to retrieve audio source configuratoins using **GetAudioSourceConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve audio source configuratoin options using **GetAudioSourceConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify audio source configuratoin using **SetAudioSourceConfiguration** operation (Media2 Service) AND

3. Client is considered as NOT supporting Audio Source Configuration if ANY of the following is TRUE:
   - No valid response to **GetAudioSourceConfigurations** request (Media2 Service) OR
   - No valid response to **GetAudioSourceConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetAudioSourceConfiguration** request (Media2 Service) OR

7.3.3 GET AUDIO SOURCE CONFIGURATIONS USING MEDIA2

Test Label: Audio Source Configuration - Get Audio Source Configurations

Test Case ID: MEDIA2_AUDIOSOURCECONFIGURATION-1

Feature Under Test: Get Audio Source Configurations Using Media2 (Media2_AudioSourceConfiguration_Media2_GetAudioSourceConfigurations)

Test Purpose: To verify that audio source configuration provided by Device is received by Client using the **GetAudioSourceConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioSourceConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioSourceConfigurations** request message to retrieve an audio source configuration or a list of audio source configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioSourceConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioSourceConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND

- Client **GetAudioSourceConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioSourceConfigurations** AND
- Device response on the **GetAudioSourceConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioSourceConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

7.3.4 GET AUDIO SOURCE CONFIGURATION OPTIONS USING MEDIA2

Test Label: Audio Source Configuration - Get Audio Source Configuration Options

Test Case ID: MEDIA2_AUDIOSOURCECONFIGURATION-2

Feature Under Test: Get Audio Source Configuration Options Using Media2 (Media2_AudioSourceConfiguration_Media2_GetAudioSourceConfigurationOptions)

Test Purpose: To verify that audio source configuration options provided by Device is received by Client using the **GetAudioSourceConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioSourceConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioSourceConfigurationOptions** request message to retrieve an audio source configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioSourceConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetAudioSourceConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioSourceConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioSourceConfigurationOptions** AND
- Device response on the **GetAudioSourceConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioSourceConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

7.3.5 SET AUDIO SOURCE CONFIGURATION USING MEDIA2

Test Label: Audio Source Configuration - Set Audio Source Configuration

Test Case ID: MEDIA2_AUDIOSOURCECONFIGURATION-3

Feature Under Test: Set Audio Source Configuration Using Media2 (Media2_AudioSourceConfiguration_Media2_SetAudioSourceConfiguration)

Test Purpose: To verify that Client is able to change audio source configuration provided by Device using the **SetAudioSourceConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetAudioSourceConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio feature for Media2 Service (Media2_Audio).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetAudioSourceConfiguration** request message to change an audio source configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetAudioSourceConfigurationResponse** message.

Test Result:

PASS -

- Client **SetAudioSourceConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetAudioSourceConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetAudioSourceConfiguration** AND
- Device response on the **SetAudioSourceConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetAudioSourceConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

7.4 Audio Output Configuration Using Media2 Test Cases

7.4.1 Feature Level Normative Reference:

Validated Feature: Audio Output Configuration Using Media2 (Media2_AudioOutputConfiguration)

Check Condition based on Device Features: Audio Output (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Optional

7.4.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Audio Output Configuration.
2. Client is considered as supporting Audio Output Configuration if the following conditions are met:
   - Client is able to retrieve audio output configuratoins using **GetAudioOutputConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve audio output configuratoin options using **GetAudioOutputConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify audio output configuratoin using **SetAudioOutputConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Audio Output Configuration if ANY of the following is TRUE:
   - No valid response to **GetAudioOutputConfigurations** request (Media2 Service) OR
   - No valid response to **GetAudioOutputConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetAudioOutputConfiguration** request (Media2 Service) OR

7.4.3 GET AUDIO OUTPUT CONFIGURATIONS USING MEDIA2

Test Label: Audio Output Configuration - Get Audio Output Configurations

Test Case ID: MEDIA2_AUDIOOUTPUTCONFIGURATION-1

Feature Under Test: Get Audio Output Configurations Using Media2 (Media2_AudioOutputConfiguration_Media2_GetAudioOutputConfigurations)

Test Purpose: To verify that audio output configuration provided by Device is received by Client using the **GetAudioOutputConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioOutputConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).

- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioOutputConfigurations** request message to retrieve an audio output configuration or a list of audio output configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioOutputConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioOutputConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioOutputConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioOutputConfigurations** AND
- Device response on the **GetAudioOutputConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioOutputConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

7.4.4 GET AUDIO OUTPUT CONFIGURATION OPTIONS USING MEDIA2

Test Label: Audio Output Configuration - Get Audio Output Configuration Options

Test Case ID: MEDIA2_AUDIOOUTPUTCONFIGURATION-2

Feature Under Test: Get Audio Output Configuration Options Using Media2 (Media2_AudioOutputConfiguration_Media2_GetAudioOutputConfigurationOptions)

Test Purpose: To verify that audio output configuration options provided by Device is received by Client using the **GetAudioOutputConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioOutputConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioOutputConfigurationOptions** request message to retrieve an audio output configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioOutputConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetAudioOutputConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioOutputConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioOutputConfigurationOptions** AND
- Device response on the **GetAudioOutputConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioOutputConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

7.4.5 SET AUDIO OUTPUT CONFIGURATION USING MEDIA2

Test Label: Audio Output Configuration - Set Audio Output Configuration

Test Case ID: MEDIA2_AUDIOOUTPUTCONFIGURATION-3

Feature Under Test: Set Audio Output Configuration Using Media2 (Media2_AudioOutputConfiguration_Media2_SetAudioOutputConfiguration)

Test Purpose: To verify that Client is able to change audio output configuration provided by Device using the **SetAudioOutputConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetAudioOutputConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetAudioOutputConfiguration** request message to change an audio output configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetAudioOutputConfigurationResponse** message.

Test Result:

PASS -

- Client **SetAudioOutputConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetAudioOutputConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetAudioOutputConfiguration** AND
- Device response on the **SetAudioOutputConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetAudioOutputConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

7.5 Audio Decoder Configuration Using Media2 Test Cases

7.5.1 Feature Level Normative Reference:

Validated Feature: Audio Decoder Configuration Using Media2 (Media2_AudioDecoderConfiguration)

Check Condition based on Device Features: Audio Output (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Optional

7.5.2 Expected Scenarios Under Test:

1. Client connects to Device to modify Audio Decoder Configuration.
2. Client is considered as supporting Audio Decoder Configuration if the following conditions are met:
   - Client is able to retrieve audio decoder configuratoins using **GetAudioDecoderConfigurations** operation (Media2 Service) AND
   - Client is able to retrieve audio decoder configuratoin options using **GetAudioDecoderConfigurationOptions** operation (Media2 Service) AND
   - Client is able to modify audio decoder configuratoin using **SetAudioDecoderConfiguration** operation (Media2 Service) AND
3. Client is considered as NOT supporting Audio Decoder Configuration if ANY of the following is TRUE:
   - No valid response to **GetAudioDecoderConfigurations** request (Media2 Service) OR
   - No valid response to **GetAudioDecoderConfigurationOptions** request (Media2 Service) OR
   - No valid response to **SetAudioDecoderConfiguration** request (Media2 Service) OR

7.5.3 GET AUDIO DECODER CONFIGURATIONS USING MEDIA2

Test Label: Audio Decoder Configuration - Get Audio Decoder Configurations

Test Case ID: MEDIA2_AUDIODECODERCONFIGURATION-1

Feature Under Test: Get Audio Decoder Configurations Using Media2 (Media2_AudioDecoderConfiguration_Media2_GetAudioDecoderConfigurations)

Test Purpose: To verify that audio decoder configuration provided by Device is received by Client using the **GetAudioDecoderConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioDecoderConfigurations** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioDecoderConfigurations** request message to retrieve an audio decoder configuration or a list of audio decoder configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioDecoderConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetAudioDecoderConfigurations** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioDecoderConfigurations** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurations** AND
- Device response on the **GetAudioDecoderConfigurations** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

7.5.4 GET AUDIO DECODER CONFIGURATION OPTIONS USING MEDIA2

Test Label: Audio Decoder Configuration - Get Audio Decoder Configuration Options

Test Case ID: MEDIA2_AUDIODECODERCONFIGURATION-2

Feature Under Test: Get Audio Decoder Configuration Options Using Media2 (Media2_AudioDecoderConfiguration_Media2_GetAudioDecoderConfigurationOptions)

Test Purpose: To verify that audio decoder configuration options provided by Device is received by Client using the **GetAudioDecoderConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioDecoderConfigurationOptions** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetAudioDecoderConfigurationOptions** request message to retrieve an audio decoder configuration options from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioDecoderConfigurationOptionsResponse** message.

Test Result:

PASS -

- Client **GetAudioDecoderConfigurationOptions** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetAudioDecoderConfigurationOptions** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurationOptions** AND

- Device response on the **GetAudioDecoderConfigurationOptions** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetAudioDecoderConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

7.5.5 SET AUDIO DECODER CONFIGURATION USING MEDIA2

Test Label: Audio Decoder Configuration - Set Audio Decoder Configuration

Test Case ID: MEDIA2_AUDIODECODERCONFIGURATION-3

Feature Under Test: Set Audio Decoder Configuration Using Media2 (Media2_AudioDecoderConfiguration_Media2_SetAudioDecoderConfiguration)

Test Purpose: To verify that Client is able to change audio decoder configuration provided by Device using the **SetAudioDecoderConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetAudioDecoderConfiguration** operation for Media2 Service present.
- Device supports Media2 Service (Media2Service).
- Device supports Audio outputs feature for Media2 Service (Media2_AudioOutput).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **SetAudioDecoderConfiguration** request message to change an audio decoder configuration on the Device.
2. Device responds with code HTTP 200 OK and **SetAudioDecoderConfigurationResponse** message.

Test Result:

PASS -

- Client **SetAudioDecoderConfiguration** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **SetAudioDecoderConfiguration** request in Test Procedure fulfills the following requirements:
  - [S1] **soapenv:Body** element has child element **tr2:SetAudioDecoderConfiguration** AND
- Device response on the **SetAudioDecoderConfiguration** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:SetAudioDecoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

7.6 Get Snapshot Uri Using Media2 Test Cases

7.6.1 Feature Level Normative Reference:

Validated Feature: Get Snapshot Uri Using Media2 (Media2_GetSnapshotUri)

Check Condition based on Device Features: Snapshot Uri (Media2 Service) is supported by Device.

Required Number of Devices: 1

Profile T Requirement: Optional

7.6.2 Expected Scenarios Under Test:

1. Client connects to Device to obtain a JPEG snapshot uri from the device using **GetSnapshotUri** operation.
2. Client gets JPEG images from the device using **HTTP GET** sent to the Uri provided by the Device in GetSnapshotUriResponse.
3. Client is considered as supporting Snapshot Uri if the following conditions are met:
   - Client is able to retrieve JPEG snapshot URI using **GetSnapshotUri** operation (Media2 Service) AND

- Client is able to retrieve JPEG images using **HTTP GET**.

4. Client is considered as NOT supporting JPEG snapshot URI if ANY of the following is TRUE:
   - No valid response to **GetSnapshotUri** request (Media2 Service).
   - No valid response for **HTTP GET** request the Uri provided by the Device in GetSnapshotUriResponse.

7.6.3 GET SNAPSHOT URI USING MEDIA2

Test Label: SnapshotUri - Get Snapshot Uri

Test Case ID: MEDIA2_GETSNAPSHOTURI-1

Feature Under Test: Get Snapshot Uri (Media2_GetSnapshotUri_Media2_GetSnapshotUriRequest)

Test Purpose: To verify that snapshot URI provided by Device is received by Client using the **GetSnapshotUri** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetSnapshotUri** operation present.
- Device supports SnapshotUri feature for Media2 Service (Media2_SnapshotUri).

Test Procedure (expected to be reflected in network trace file):

1. Client invokes **GetSnapshotUri** request message to retrieve snapshot Uri from the Device.
2. Device responds with code HTTP 200 OK and **GetSnapshotUriResponse** message.
3. Client invokes **HTTP GET** request to snapshot Uri.
4. Client responds with code HTTP 200 OK.

Test Result:

PASS -

- Client **GetSnapshotUri** request messages are valid according to XML Schemas listed in Namespaces AND
- Client **GetSnapshotUri** request in Test Procedure fulfills the following requirements:

- [S1] **soapenv:Body** element has child element **tr2:GetSnapshotUri** AND

- Device response on the **GetSnapshotUri** request fulfills the following requirements:
  - [S2] It has HTTP 200 response code AND
  - [S3] **soapenv:Body** element has child element **tr2:GetSnapshotUriResponse** AND
- There is **HTTP GET** request in Test Procedure that fulfills the following requirements:
  - [S4] It invoked to address which equal to **tr2:GetSnapshotUriResponse/tr2:Uri** value from the Device response to GetSnapshotUri request AND
  - [S5] It invoked after the Client **GetSnapshotUri** request AND
- Device response on the **HTTP GET** request fulfills the following requirements:
  - [S6] It has **HTTP 200** response code.

FAIL -

- The Client failed PASS criteria.

Annex A Test for Appendix A

A.1 Get default PTZ space of PTZ Configuration corresponding to Move Operation

Name: HelperGetDefaultPtzSpaceOfPtzConfigurationCorrespondingToMove

Procedure Purpose: Get default PTZ space of PTZ Configuration corresponding to PTZ Move Operation.

Pre-requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetProfiles** (Media2 Service) or **AddConfiguration** (Media2 Service) operations present.

Input: PTZ Move Opertaion request (AbsoluteMove or ContinuousMove) (*moveOpertaion*), Default space element name to get (*defaultSpace*)

Returns: Default PTZ space value (*spaceValue*).

Annex Procedure:

- The Client Test Tool checks that there is Client **AddConfiguration** request or Device **GetProfilesResponse** in Test Procedure that fulfills the following requirements:
  - [S1] It is invoked before *moveOpertaion* request AND
  - If it is **AddConfiguration** request:
    - [S2] **tr2:AddConfiguration/tr2:ProfileToken** value is equal to **ProfileToken** value from *moveOpertaion* request AND
    - [S3] **tr2:AddConfiguration** has **tr2:Configuration** element with **tr2:Type** value is equal to **PTZ** AND

    If it is **GetProfilesResponse** message:
    - [S4] It contains **tr2:Profiles** element with **tr2:Profiles/@token** value is equal to **ProfileToken** value from *moveOpertaion* request (hereinafter *profile*) AND
    - [S5] *profile* has **tr2:Configurations/tr2:PTZ** element AND
  - [S6] It is the closest one preceding *moveOpertaion* request that fullfils ([S2] and [S3]) or ([S4] and [S5]) requirements AND

- The Client Test Tool checks if there is **SetConfiguration** command that fulfills the following requirements:
  - If **AddConfiguration** request was found during previous steps:
    - [S7] It invoked after **AddConfiguration** request AND
    - [S8] It is the closest one preceding the *moveOpertaion* request AND
    - [S9] **tptz:SetConfiguration/tptz:PTZConfiguration/@token** value is equal to **tr2:AddConfiguration/tr2:Configuration/tr2:Token** value of **tr2:Configuration** with **tr2:Type** value is equal to **PTZ** AND
  - If **GetProfiles** request was found during previous steps:
    - [S10] It invoked after **GetProfiles** request AND
    - [S11] It is the closest one preceding the *moveOpertaion* request AND
    - [S12] **tptz:SetConfiguration/tptz:PTZConfiguration/@token** value is equal to **tr2:PTZ/@token** value of *profile* AND
- IF **SetConfiguration** command was detected during previous steps then *defaultSpace* element value of **tptz:SetConfiguration/tptz:PTZConfiguration** will be returened as result of current procedure
- ELSE IF **GetProfiles** response was found during previous steps then *defaultSpace* value of **tr2:Configurations/tr2:PTZ** from *profile* will be returened as result of current procedure
- ELSE IF **AddConfiguration** request was found during previous steps and no **SetConfiguration** was found during previous steps, the Client Test Tool checks the following:
  - There is **GetCompatibleConfigurations** request in Test Procedure that fulfills the following requirements:
    - [S13] It is the closest one preceding the **AddConfiguration** request AND
    - [S14] **tptz:GetCompatibleConfigurations/tptz:ProfileToken** value is equal to **ProfileToken** value from *moveOpertaion* request AND
  - Device response on the **GetCompatibleConfigurations** request fulfills the following requirements:
    - [S15] It has HTTP 200 response code AND
    - [S16] **soapenv:Body** element has child element **tptz:GetCompatibleConfigurationsResponse** AND

- [S17] It contains **tptz:PTZConfiguration/@token** value is equal to **tr2:AddConfiguration/tr2:Configuration/tr2:Token** value of **tr2:Configuration** with **tr2:Type** value is equal to **PTZ** AND

- [S18] *defaultSpace* value from **tptz:GetCompatibleConfigurationsResponse/ tptz:PTZConfiguration** element with @token is equal to **tr2:AddConfiguration/ tr2:Configuration/tr2:Token** value of **tr2:Configuration** with **tr2:Type** value is equal to **PTZ** will be returened as result of current procedure.

A.2 Define rule type corresponding to Rule Name

Name: HelperDefineTypeCorrespondingToRuleName

Procedure Purpose: Define rule type corresponding to Rule Name.

Pre-requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetRules** or **CreateRules** operations present.

Input: Delete Rules request (*deleteRules*), Rule Name (*ruleName*)

Returns: Rule Type (*ruleType*).

Annex Procedure:

- The Client Test Tool checks that there is Client **CreateRules** request or Device **GetRulesResponse** in Test Procedure that fulfills the following requirements:

  - [S1] It is invoked before *deleteRules* request AND

  - If **CreateRules** request was found during previous steps:

    - [S2] **axt:CreateRules/axt:ConfigurationToken** value is equal to **axt:ConfigurationToken** value in *deleteRules* AND

    - [S3] **CreateRules** has **axt:Rule** element (*rule1*) with **@Name** value is equal to *ruleName* AND

    - Device response on the **CreateRules** request fulfills the following requirements:

      - [S4] It has HTTP 200 response code AND

      - [S5] **soapenv:Body** element has child element **axt:CreateRulesResponse** AND

  If **GetRulesResponse** was found during previous steps:

- [S6] It has **tr2:Rule** element (*rule2*) with **@Name** value is equal to *ruleName* AND
- [S7] It has HTTP 200 response code AND
- [S8] **soapenv:Body** element has child element **axt:GetRulesResponse** AND
- [S9] **GetRules** request fulfills the following requirements:
  - [S10] **axt:GetRules/axt:ConfigurationToken** value is equal to **axt:ConfigurationToken** value in *deleteRules* AND
- [S11] It is the closest one preceding *deleteRules* request that fullfils ([S2] and [S3]) or ([S4] and [S5]) requirements AND

- The Client Test Tool checks if there is **ModifyRules** command that fulfills the following requirements:
  - [S12] It invoked after operation detected at previous steps (**CreateRules** OR **GetRulesResponse**) AND
  - [S13] It is the closest one preceding the *deleteRules* request AND
  - [S14] **axt:ModifyRules/axt:ConfigurationToken** value is equal to **axt:ConfigurationToken** value in *deleteRules* request AND
  - [S15] **ModifyRules** has **tr2:Rule** element (*rule3*) with **@Name** value is equal to *ruleName* AND
  - Device response on the **ModifyRules** request fulfills the following requirements:
    - [S16] It has HTTP 200 response code AND
    - [S17] **soapenv:Body** element has child element **axt:ModifyRulesResponse** AND
- [S18] IF **ModifyRules** command was detected during previous steps then *ruleType* = *rule3*.@Type
- [S19] ELSE IF **GetRulesResponse** response was found during previous steps then *ruleType* = *rule2*.@Type
- [S20] ELSE IF **CreateRules** request was found during previous steps then *ruleType* = *rule1*.@Type

A.3 Required Number of Devices Summary

Required number of devices and Device feature dependency used in this test specification are listed in the Table.

Table A.1. Required Number of Devices Summary

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.HTTPDigest	HTTP Digest	3	Digest	Digest
tc.Capabilities	Capabilities	3	None	All
tc.GetServices	Get Services	3	GetServices is supported by Device.	GetServices
tc.EventHandling	Event Handling	3	Pull Point Notification OR WS Basic Notification OR Profile S OR Metadata under Media2 service is supported by Device.	no UnsupportedPullPointNotification OR WSBasicNotification OR Profile S OR Media2_Metadata
tc.SetSynchronizationPoint	Set Synchronization Point (Event Service)	1	Pull Point Notification OR WS-Basic Notification is supported by Device.	no UnsupportedPullPointNotification OR WSBasicNotification
tc.Discovery	Discovery	3	Discovery	All
tc.DeviceDiscoveryTypeFilter	Device Discovery Type Filter	3	Device Discovery Type is supported by Device.	DiscoveryTypesTdsDevice
tc.NetworkConfiguration	Network Configuration	3	Network Configuration	no NetworkConfigNotSupported
tc.HTTPDigestForRTSP	HTTP Digest Authentication for RTSP	3	Profile T or Profile M	ProfileTSupported or ProfileMSupported

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.Media2_GetProfiles	Get Profiles Using Media2	3	Media2 Service is supported by Device.	Media2Service
tc.Media2_GetStreamURI	Get Stream Uri Using Media2	3	Real Time Streaming (Media2 Service) is supported by Device.	Media2_RealTimeStreaming
tc.Media2_MediaStreaming	Media Streaming Using Media2	3	Real Time Streaming (Media2 Service) is supported by Device.	Media2_RealTimeStreaming
tc.Media2_VideoStreaming_H264	H264 Video Streaming Using Media2	3	Real Time Streaming (Media2 Service) and H264 (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_H264
tc.Media2_VideoStreaming_H265	H265 Video Streaming Using Media2	3	Real Time Streaming (Media2 Service) and H265 (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_H265
tc.Media2_VideoEncoderConfiguration	Video Encoder Configuration Using Media2	3	Video (Media2 Service) is supported by Device.	Media2_Video
tc.GetImagingSettings	Get Imaging Settings	3	Imaging Service is supported by Device.	ImagingService
tc.SetImagingSettings	Imaging Settings Configuration	3	Imaging Service is supported by Device.	ImagingService

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.MotionAlarm	Motion Alarm	3	Motion Alarm is supported by Device.	MotionAlarm
tc.PtzPanTiltContinuousPositioning	PTZ Pan Tilt Continuous Positioning	1	PTZ Continuous Pan Tilt movement is supported by Device.	PTZContinuousPanTilt
tc.PtzZoomContinuousPositioning	PTZ Zoom Continuous Positioning	1	PTZ Continuous Zoom movement is supported by Device.	PTZContinuousZoom
tc.Media2_PanTiltSpaces_SphericalPositionSpaceDegrees	PTZ Using Media2 Absolute Positioning - Spherical Position Space Degrees	3	Profile T, PTZ Absolute Move and PTZ Spherical Coordinate Spaces are supported by Device.	PTZAbsolute AND PTZSpherical CoordinateSpaces AND ProfileTSupported
tc.Media2_PanTiltSpaces_PositionGenericSpace	PTZ Using Media2 Absolute Positioning - Pan Tilt Position Generic Space	3	PTZ Absolute Pan/Tilt Move and PTZ Generic Coordinate Spaces are supported by Device.	PTZAbsolutePanTilt AND PTZGenericCoordinateSpaces AND Media2Service
tc.Media2_ZoomSpaces_PositionGenericSpace	PTZ Using Media2 Absolute Positioning - Zoom Position Generic Space	3	PTZ Absolute Zoom Move and PTZ Generic Coordinate Spaces are supported by Device.	PTZAbsoluteZoom AND PTZGenericCoordinateSpaces AND Media2Service
tc.Media2_PanTiltSpaces_V	PTZ Using Media2 Pan	3	PTZ Continuous PanTilt and Media2 Service	PTZContinuousPanTilt AND Media2Service

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
elocityGeneri cSpace	Tilt Continuous Positioning		are supported by Device.	
tc.Media2_Zo omSpaces_Velo cityGenericSpace	PTZ Using Media2 Zoom Continuous Positioning	3	PTZ Continuous Zoom and Medis2 Service are supported by Device.	PTZContinuou sZoom AND Media2Service
tc.System	System	3	None	All
tc.UserHandling	User Handling	3	User Configuratio n	no UserConfigNo tSupported
tc.NTP	NTP	1	NTP is supported by Device.	NTP
tc.Auxiliary Commands	Auxiliary Commands (Device Management Service)	1	None (ONVIF Profile T Simulator is used as device).	None (ONVIF Profile T Simulator is used as device)
tc.Media2_Me tadataProfile Configuration	Metadata Profile Configuration Using Media2	1	Media2 Service is supported by Device.	Media2Service
tc.Media2_Me tadataConfigu ration	Metadata Configuration Using Media2	1	Media2 Service is supported by Device.	Media2Service
tc.Media2_Me tadataStreaming	Metadata Streaming Using Media2	1	Media2 Service is supported by Device.	Media2Service
tc.Media2_Mu lticastStreaming	Multicast Streaming Using Media2	1	RTP-Multicast/ UDP (Media2 Service) is supported by Device.	Media2_RTPMu lticastUDP
tc.Media2_Vi deoProfileCon figuration	Video Profile Configuration Using Media2	1	Video (Media2 Service) is supported by Device.	Media2_Video

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.Media2_AudioEncoderConfiguration	Audio Encoder Configuration Using Media2	1	Audio (Media2 Service) is supported by Device.	Media2_Audio
tc.Media2_AudioStreaming_G711	G.711 Audio Streaming Using Media2	1	Real Time Streaming (Media2 Service) and Audio G711 (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_G711
tc.Media2_AudioStreaming_AAC	AAC Audio Streaming Using Media2	1	Real Time Streaming (Media2 Service) and Audio AAC (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_AAC
tc.Media2_AudioProfileConfiguration	Audio Profile Configuration Using Media2	1	Audio (Media2 Service) is supported by Device.	Media2_Audio
tc.Media2_AudioBackchannelStreaming	Audio Backchannel Streaming Using Media2	1	Real Time Streaming (Media2 Service) and Audio Output (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_Audio Output
tc.Media2_AudioOutputProfileConfiguration	Audio Output Profile Configuration Using Media2	1	Audio Output (Media2 Service) is supported by Device.	Media2_Audio Output
tc.Media2_AnalyticsProfileConfiguration	Analytics Profile Configuration Using Media2	1	Analytics (Media2 Service) is supported by Device.	Media2_Analytics

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.VideoSourceMode	Video Source Mode	1	Video Source Mode (Media2 Service) is supported by Device.	Media2_VideoSourceMode
tc.Media2_VideoSourceConfiguration	Video Source Configuration Using Media2	1	Video (Media2 Service) or VideoSource (Media2 Service) is supported by Device.	Media2_Video OR Media2_VideoSource
tc.Media2_ListVideoSourceConfigurations	List Video Source Configurations Using Media2	1	Video (Media2 Service) is supported by Device.	Media2_Video
tc.Media2_OSDConfiguration	OSD Configuration Using Media2	1	OSD (Media2 Service) is supported by Device.	Media2_OSD
tc.Media2_MediaProfileManagement	Media Profile Management	1	Real Time Streaming (Media2 Service) and Video (Media2 Service) are supported by Device.	Media2_RealTimeStreaming AND Media2_Video
tc.Media2_HTTPSStreaming	HTTPS Streaming Using Media2	1	No (ONVIF Profile T Simulator is used as device).	
tc.GetMoveOptions	Focus Move Capabilities	1	Imaging Service is supported by Device.	ImagingService
tc.FocusControl	Focus Control	1	Focus Control is supported by Device.	FocusControl

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.Tampering	Tampering	1	None (ONVIF Profile T Simulator is used as device).	None (ONVIF Profile T Simulator is used as device)
tc.PtzListing	PTZ - Listing	1	PTZ Service is supported by Device.	PTZService
tc.PtzPresets	PTZ Presets	1	PTZ Presets is supported by Device.	PTZPresets
tc.PtzHomePosition	PTZ Home Position	1	PTZ Home Position is supported by Device.	PTZHome
tc.PtzSetPreset	PTZ - Set Preset	1	PTZ Presets is supported by Device.	PTZPresets
tc.PtzGetCompatibleConfigurations	PTZ Get Compatible Configurations	1	GetCompatibleConfigurations (PTZ Service) is supported by Device.	PTZGetCompatibleConfigurations
tc.PTZUsingMedia2ProfileConfiguration	PTZ Media2 Profile Configuration	1	PTZ Service and Media2 Service are supported by Device.	PTZService AND Medi2Service
tc.PtzSetConfiguration	PTZ Set Configuration	1	PTZ Service is supported by Device.	PTZService
tc.RelayOutputsUsingDeviceIO	Relay Outputs Using Device IO	1	Relay Outputs (Device IO Service) is supported by Device.	RelayOutputs
tc.GetDigitalInputs	Get Digital Inputs	1	Digital Inputs (Device IO	DigitalInputs

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			Service) is supported by Device.	
tc.GetSupportedRules	Get Supported Rules	1	Media 2 Service and Rule Engine is supported by Device.	Media2Service AND RuleEngine
tc.GetRules	Get Rules	1	Media 2 Service and Rule Engine is supported by Device.	Media2Service AND RuleEngine
tc.MotionDetection	Motion Detection	1	Media 2 Service and Rule Engine and Rule Options and Motion Region Detector Rule and Pull Point Notification is supported by Device.	Media2Service AND RuleEngine AND RuleOptions AND MotionRegion DetectorRule AND no UnsupportedPullPointNotification
tc.KeepAliveForPullPointEventHandling	Keep Alive for Pull Point Event Handling	3	Pull Point Notification is supported by Device.	no UnsupportedPullPointNotificatio
tc.Media2_AudioSourceConfiguration	Audio Source Configuration Using Media2	1	Audio (Media2 Service) is supported by Device.	Media2_Audio
tc.Media2_AudioOutputConfiguration	Audio Output Configuration Using Media2	1	Audio Output (Media2 Service) is supported by Device.	Media2_Audio Output
tc.Media2_AudioDecoderConfiguration	Audio Decoder Configuration Using Media2	1	Audio Output (Media2 Service) is supported by Device.	Media2_Audio Output

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.Media2_GetSnapshotUri	Get Snapshot Uri Using Media2	1	Snapshot Uri (Media2 Service) is supported by Device.	Media2_SnapshotUri