

ONVIF[®]

Other Features Client Test Specification

Version 22.06

June 2022

© 2022 ONVIF, Inc. All rights reserved.

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

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

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

REVISION HISTORY

Vers.	Date	Description
22.06	Apr 07, 2022	Media2Service AND Mask were added to Check Condition based on Device Features
22.06	Mar 18, 2022	The following test cases added according to #327: Privacy Mask Using Media2 Test Cases
22.06	Dec 27, 2022	Profile Normative Reference were removed from test cases according to #364
21.12	Oct 07, 2021	The following was done according to #426: Monitoring Notifications section and Monitoring Notifications Test Cases was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: HTTP System Restore section and HTTP System Restore Test Cases section was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: HTTP System Backup section and HTTP System Backup Test Cases section was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: HTTP Firmware Upgrade section and HTTP Firmware Upgrade Test Cases section was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: Device Management Notifications section and Device Management Notifications Test Cases section was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: TLS Configuration section and TLS Configuration Test Cases section was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
21.12	Oct 07, 2021	The following was done according to #426: Operational State section and Operational State Test Cases was moved from ONVIF Profile Q Client Test Specification to ONVIF Other Features Client Test Specifications.
19.12	Oct 04, 2019	Note about not found GetStreamUri was added in the following test cases according to #339: AUDIOBACKCHANNELSTREAMING-2 G.711 AUDIO BACKCHANNEL STREAMING

		AUDIOBACKCHANNELSTREAMING-3 BACKCHANNEL STREAMING	G.726	AUDIO
		AUDIOBACKCHANNELSTREAMING-4 BACKCHANNEL STREAMING	AAC	AUDIO
19.12	Sep 18, 2019	The following was done according to #325: Scope\Supplementary Features and Test Cases sections was added. Supplementary Features and Test Cases sections was added.		
19.12	Aug 14, 2019	Initial version.		

Table of Contents

1	Introduction	11
1.1	Scope	11
1.2	Audio Backchannel for Media Features	12
1.2.1	Audio Backchannel Streaming	12
1.2.2	Get Audio Decoder Configurations List	12
1.2.3	Get Audio Output Configurations List	12
1.2.4	Get Audio Outputs List	12
1.2.5	Get Audio Decoder Configuration	12
1.2.6	Get Audio Output Configuration	12
1.2.7	Profile Configuration for Audio Backchannel	12
1.2.8	Configure Audio Decoder Configuration	12
1.2.9	Configure Audio Output Configuration	13
1.3	Imaging Features	13
1.3.1	Get Imaging Capabilities	13
1.4	OSD for Media Features	13
1.4.1	Get OSD Configuration	13
1.4.2	Get OSD List	13
1.4.3	OSD Configuration	13
1.5	Security Configuration Features	13
1.5.1	Enabled TLS Versions Configuration	13
1.6	Operational State Features	13
1.6.1	Transition to Operational State	13
1.7	Firmware Upgrade Features	14
1.7.1	HTTP Firmware Upgrade	14
1.8	Backup and Restore Features	14
1.8.1	HTTP System Backup	14
1.8.2	HTTP System Restore	14
1.9	Standard Events for Monitoring Features	14
1.9.1	Monitoring Notifications	14
1.10	Standard Events for Device Management Features	14

- 1.10.1 Device Management Notifications 14
- 1.11 TLS Configuration Features 14
 - 1.11.1 TLS Configuration 14
- 1.12 Mask for Media2 Features 15
 - 1.12.1 Mask Configuration Using Media2 15
- 2 Normative references 16**
- 3 Terms and Definitions 18**
 - 3.1 Conventions 18
 - 3.2 Definitions 18
 - 3.3 Abbreviations 18
 - 3.4 Namespaces 19
- 4 Test Overview 21**
 - 4.1 General 21
 - 4.1.1 Feature Level Requirement 21
 - 4.1.2 Expected Scenarios Under Test 22
 - 4.1.3 Test Cases 22
 - 4.2 Test Setup 22
 - 4.3 Prerequisites 23
- 5 Test Cases for Audio Backchannel for Media 24**
 - 5.1 Audio Backchannel Streaming Test Cases 24
 - 5.1.1 Feature Level Requirement: 24
 - 5.1.2 Expected Scenarios Under Test: 24
 - 5.1.3 GET AUDIO DECODER CONFIGURATION OPTIONS 25
 - 5.1.4 G.711 AUDIO BACKCHANNEL STREAMING 26
 - 5.1.5 G.726 AUDIO BACKCHANNEL STREAMING 29
 - 5.1.6 AAC AUDIO BACKCHANNEL STREAMING 31
 - 5.2 Get Audio Decoder Configurations List Test Cases 34
 - 5.2.1 Feature Level Requirement: 34
 - 5.2.2 Expected Scenarios Under Test: 35
 - 5.2.3 GET AUDIO DECODER CONFIGURATIONS 35
 - 5.3 Get Audio Output Configurations List Test Cases 36

- 5.3.1 Feature Level Requirement: 36
- 5.3.2 Expected Scenarios Under Test: 37
- 5.3.3 GET AUDIO OUTPUT CONFIGURATIONS 37
- 5.4 Get Audio Outputs List Test Cases 38
 - 5.4.1 Feature Level Requirement: 38
 - 5.4.2 Expected Scenarios Under Test: 38
 - 5.4.3 GET AUDIO OUTPUTS 39
- 5.5 Get Audio Decoder Configuration Test Cases 40
 - 5.5.1 Feature Level Requirement: 40
 - 5.5.2 Expected Scenarios Under Test: 40
 - 5.5.3 GET AUDIO DECODER CONFIGURATION 40
- 5.6 Get Audio Output Configuration Test Cases 42
 - 5.6.1 Feature Level Requirement: 42
 - 5.6.2 Expected Scenarios Under Test: 42
 - 5.6.3 GET AUDIO OUTPUT CONFIGURATION 42
- 5.7 Profile Configuration for Audio Backchannel Test Cases 43
 - 5.7.1 Feature Level Requirement: 43
 - 5.7.2 Expected Scenarios Under Test: 44
 - 5.7.3 GET COMPATIBLE AUDIO OUTPUT CONFIGURATIONS 45
 - 5.7.4 ADD AUDIO OUTPUT CONFIGURATION 46
 - 5.7.5 REMOVE AUDIO OUTPUT CONFIGURATION 48
 - 5.7.6 GET COMPATIBLE AUDIO DECODER CONFIGURATIONS 49
 - 5.7.7 ADD AUDIO DECODER CONFIGURATION 50
 - 5.7.8 REMOVE AUDIO DECODER CONFIGURATION 52
- 5.8 Configure Audio Decoder Configuration Test Cases 53
 - 5.8.1 Feature Level Requirement: 53
 - 5.8.2 Expected Scenarios Under Test: 53
 - 5.8.3 SET AUDIO DECODER CONFIGURATION 54
- 5.9 Configure Audio Output Configuration Test Cases 55
 - 5.9.1 Feature Level Requirement: 55
 - 5.9.2 Expected Scenarios Under Test: 55

5.9.3	GET AUDIO OUTPUT CONFIGURATION OPTIONS	56
5.9.4	SET AUDIO OUTPUT CONFIGURATION	57
6	Test Cases for Imaging	59
6.1	Get Imaging Capabilities Test Cases	59
6.1.1	Feature Level Requirement:	59
6.1.2	Expected Scenarios Under Test:	59
6.1.3	GET CAPABILITIES	59
6.1.4	GET SERVICE CAPABILITIES	60
7	Test Cases for OSD for Media	62
7.1	Get OSD Configuration Test Cases	62
7.1.1	Feature Level Requirement:	62
7.1.2	Expected Scenarios Under Test:	62
7.1.3	GET OSD	62
7.2	Get OSD List Test Cases	63
7.2.1	Feature Level Requirement:	63
7.2.2	Expected Scenarios Under Test:	63
7.2.3	GET OSDS	64
7.3	OSD Configuration Test Cases	65
7.3.1	Feature Level Requirement:	65
7.3.2	Expected Scenarios Under Test:	65
7.3.3	GET OSD OPTIONS	65
7.3.4	SET OSD	66
8	Test Cases for Security Configuration	68
8.1	Enabled TLS Versions Configuration Test Cases	68
8.1.1	Feature Level Requirement:	68
8.1.2	Expected Scenarios Under Test:	68
8.1.3	Get Enabled TLS Versions	69
8.1.4	Set Enabled TLS Versions	70
9	Test Cases for Operational State	72
9.1	Transition to Operational State Test Cases	72
9.1.1	Feature Level Requirement:	72

- 9.1.2 Expected Scenarios Under Test: 72
- 9.1.3 TRANSITION TO OPERATIONAL STATE BY CREATEUSERS 72
- 9.1.4 TRANSITION TO OPERATIONAL STATE BY SET USER 74
- 10 Test Cases for Firmware Upgrade 76**
- 10.1 HTTP Firmware Upgrade Test Cases 76
 - 10.1.1 Feature Level Requirement: 76
 - 10.1.2 Expected Scenarios Under Test: 76
 - 10.1.3 FIRMWARE UPGRADE VIA HTTP 76
- 11 Test Cases for Backup and Restore 79**
- 11.1 HTTP System Backup Test Cases 79
 - 11.1.1 Feature Level Requirement: 79
 - 11.1.2 Expected Scenarios Under Test: 79
 - 11.1.3 GET SYSTEM URIS 79
- 11.2 HTTP System Restore Test Cases 81
 - 11.2.1 Feature Level Requirement: 81
 - 11.2.2 Expected Scenarios Under Test: 81
 - 11.2.3 HTTP SYSTEM RESTORE 81
- 12 Test Cases for Standard Events for Monitoring 84**
- 12.1 Monitoring Notifications Test Cases 84
 - 12.1.1 Feature Level Requirement: 84
 - 12.1.2 Expected Scenarios Under Test: 84
 - 12.1.3 PULLPOINT 85
- 13 Test Cases for Standard Events for Device Management 87**
- 13.1 Device Management Notifications Test Cases 87
 - 13.1.1 Feature Level Requirement: 87
 - 13.1.2 Expected Scenarios Under Test: 87
 - 13.1.3 PULLPOINT 88
- 14 Test Cases for TLS Configuration 90**
- 14.1 TLS Configuration Test Cases 90
 - 14.1.1 Feature Level Requirement: 90
 - 14.1.2 Expected Scenarios Under Test: 90

14.1.3	PULLPOINT	93
14.1.4	SET NETWORK INTERFACES	94
14.1.5	UPLOAD PASSPHRASE	95
14.1.6	DELETE PASSPHRASE	96
14.1.7	CREATE PKCS#10 CERTIFICATION	97
14.1.8	UPLOAD CERTIFICATE	98
14.1.9	DELETE CERTIFICATE	99
14.1.10	DELETE CERTIFICATION PATH	100
14.1.11	DELETE KEY	101
14.1.12	GET KEY STATUS	102
14.1.13	UPLOAD PKCS12	103
14.1.14	ADD SERVER CERTIFICATE ASSIGNMENT	104
14.1.15	REMOVE SERVER CERTIFICATE ASSIGNMENT	105
14.1.16	REPLACE SERVER CERTIFICATE ASSIGNMENT	106
14.1.17	CREATE CERTIFICATION PATH	107
14.1.18	CREATE RSA KEY PAIR	108
15	Test Cases for Privacy Masks for Media2	110
15.1	Privacy Masks for Media2 Test Cases	110
15.1.1	Feature Level Normative Reference:	110
15.1.2	Expected Scenarios Under Test:	110
15.1.3	GET MASKS USING MEDIA2	110
15.1.4	CREATE MASK USING MEDIA2	111
15.1.5	GET MASK OPTIONS USING MEDIA2	112
15.1.6	DELETE MASK USING MEDIA2	113
15.1.7	SET MASK USING MEDIA2	114
A	Test for Appendix A	116
A.1	Required Number of Devices Summary	116

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 Client features that are out of any profiles. 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 Other Features Client Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant Clients in the scope of features which are out of any profile. 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 ONVIF Network Specification.

The principal intended purposes are:

- Provide self-assessment tool for implementations.
- Provide comprehensive test suite coverage for Audio Backchannel for Media features.
- Provide comprehensive test suite coverage for some Imaging features.
- Provide comprehensive test suite coverage for OSD features for Media.
- Provide comprehensive test suite coverage for TLS Enabled Version configuration.

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 Audio Backchannel for Media Features

1.2.1 Audio Backchannel Streaming

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

1.2.2 Get Audio Decoder Configurations List

Get Audio Decoder Configurations List section specifies Client ability to request audio decoder configurations list from a Device.

1.2.3 Get Audio Output Configurations List

Get Audio Output Configurations List section specifies Client ability to request audio output configurations list from a Device.

1.2.4 Get Audio Outputs List

Get Audio Outputs List section specifies Client ability to request audio outputs list from a Device.

1.2.5 Get Audio Decoder Configuration

Get Audio Decoder Configuration section specifies Client ability to request audio decoder settings from a Device.

1.2.6 Get Audio Output Configuration

Get Audio Output Configuration section specifies Client ability to request audio output settings from a Device.

1.2.7 Profile Configuration for Audio Backchannel

Profile Configuration for Audio Backchannel section specifies Client ability to configure media profile for audio backchannel streaming on a Device.

1.2.8 Configure Audio Decoder Configuration

Configure Audio Decoder Configuration section specifies Client ability to change audio decoder configuration on a Device.

1.2.9 Configure Audio Output Configuration

Configure Audio Output Configuration section specifies Client ability to change audio output configuration on a Device.

1.3 Imaging Features

1.3.1 Get Imaging Capabilities

Get Imaging Capabilities section specifies Client ability to request imaging capabilities from Device.

1.4 OSD for Media Features

1.4.1 Get OSD Configuration

Get OSD Configuration section specifies Client ability to request OSD configuration from Device.

1.4.2 Get OSD List

Get OSD List section specifies Client ability to request OSD list from Device.

1.4.3 OSD Configuration

OSD Configuration section specifies Client ability to change OSD settings on Device.

1.5 Security Configuration Features

1.5.1 Enabled TLS Versions Configuration

Enabled TLS Versions Configuration section specifies Client ability to configure enabled TLS versions on Device.

1.6 Operational State Features

1.6.1 Transition to Operational State

Transition to Operational State section specifies Client ability to transit an ONVIF Device from Factory Default State into Operational State.

1.7 Firmware Upgrade Features

1.7.1 HTTP Firmware Upgrade

HTTP Firmware Upgrade section defines Client ability to upgrade Device firmware over HTTP using StartFirmwareUpgrad operation and HTTP POST.

1.8 Backup and Restore Features

1.8.1 HTTP System Backup

HTTP System Backup section defines Client ability to backup system configurations over HTTP using GetSystemUris operation and HTTP GET.

1.8.2 HTTP System Restore

HTTP System Restore section defines Client ability to restore system configurations over HTTP using StartSystemRestore operation and HTTP POST.

1.9 Standard Events for Monitoring Features

1.9.1 Monitoring Notifications

Monitoring Notifications section specifies Client ability to receive from Device monitoring notifications.

1.10 Standard Events for Device Management Features

1.10.1 Device Management Notifications

Device Management Notifications section specifies Client ability to receive from Device device management notifications.

1.11 TLS Configuration Features

1.11.1 TLS Configuration

TLS Configuration section specifies Client ability to manage the associations between certification paths and the TLS server on Device.

1.12 Mask for Media2 Features

1.12.1 Mask Configuration Using Media2

Privacy Mask Using Media2 section specifies listing and modification of Mask configurations on 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 Media Service Specification:
<https://www.onvif.org/profiles/specifications/>
- ONVIF Streaming Specification:
<https://www.onvif.org/profiles/specifications/>

- ONVIF Imaging Service Specification:
<https://www.onvif.org/profiles/specifications/>
- ONVIF Security Configuration Specification:
<https://www.onvif.org/profiles/specifications/>
- 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.
Configuration Entity	A network video device media abstract component that is used to produce a media stream on the network, i.e. video and/or audio stream.
Media Profile	Maps a video or an audio source or an audio output to a video or an audio encoder, an audio decoder configuration and PTZ and analytics configuration

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.
RTSP	Real Time Streaming Protocol.
RTP	Realtime Transport Protocol.
SDP	Session Description Protocol.
AAC	Advanced Audio Coding.
OSD	On-Screen Display.

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
tt	http://www.onvif.org/ver10/schema	ONVIF XML schema descriptions
tds	http://www.onvif.org/ver10/device/wsd	The namespace for the WSDL device service
tev	http://www.onvif.org/ver10/events/wsd	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.

Prefix	Namespace URI	Description
wsa	http://www.w3.org/2005/08/addressing	Device addressing namespace as defined by [WS-Addressing].
trt	http://www.onvif.org/ver10/media/wsd	The namespace for the WSDL media service
timg	http://www.onvif.org/ver20/imaging/wsd	The namespace for the WSDL imaging service
tas	http://www.onvif.org/ver10/advancedsecurity/wsd	The namespace for the WSDL Security Configuration service
tr2	http://www.onvif.org/ver20/media/wsd	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 with audio backchannel features support can provide audio backchannel configuration and streaming with Media Service.

An ONVIF Client with Imaging features support can provide retrieve of Imaging capabilities.

An ONVIF Client with OSD features support can provide OSD configuration with Media Service.

An ONVIF Client with security configuration features support can provide TLS Enabled Versions Configuration configuration.

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-requisite 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 ONVIF compatibility, 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 Audio Backchannel for Media

5.1 Audio Backchannel Streaming Test Cases

5.1.1 Feature Level Requirement:

Validated Feature: Audio Backchannel Streaming (AudioBackchannelStreaming)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.1.2 Expected Scenarios Under Test:

1. Client connects to Device to stream audio for backchannel.
2. Client is considered as supporting Audio Backchannel Streaming if the following conditions are met:
 - Client is able to get audio decoder configuration options to check supported audio backchannel streaming parameters using **GetAudioOutputConfigurationOptions** operation AND
 - Client is able to stream audio for backchannel using **AAC OR G.711 OR G.726**.
3. Client is considered as NOT supporting Audio Backchannel Streaming if ANY of the following is TRUE:
 - No valid responses for **GetAudioOutputConfigurationOptions** request
 - No Audio Backchannel Streaming attempts were found OR
 - Detected AAC Audio Backchannel Streaming attempts have failed OR
 - Detected G.711 Audio Backchannel Streaming attempts have failed OR

- Detected G.726 Audio Backchannel Streaming attempts have failed.

5.1.3 GET AUDIO DECODER CONFIGURATION OPTIONS

Test Label: Audio Backchannel Streaming - Get Audio Decoder Configuration Options

Test Case ID: AUDIOBACKCHANNELSTREAMING-1

Feature Under Test: Get Audio Decoder Configuration Options
(AudioBackchannelStreaming_GetAudioDecoderConfigurationOptions)

Test Purpose: To verify that Client is able to get audio decoder configuration options provided by Device using the **GetAudioDecoderConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioDecoderConfigurationOptions** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioDecoderConfigurationOptions** request message to retrieve audio decoder configuration options for 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 **trt:GetAudioDecoderConfigurationOptions** AND
- Device response to the **GetAudioDecoderConfigurationOptions** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND

- [S3] **soapenv:Body** element has child element **trt:GetAudioDecoderConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

5.1.4 G.711 AUDIO BACKCHANNEL STREAMING

Test Label: Audio Backchannel Streaming - G.711

Test Case ID: AUDIOBACKCHANNELSTREAMING-2

Feature Under Test: G.711 Audio Backchannel Streaming
(AudioBackchannelStreaming_G711AudioBackchannelStreaming)

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 audio backchannel streaming with G.711 encoding.
- Device supports G.711 encoding for Audio Outputs.

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

1. Client invokes **GetStreamUri** request message for media profile that contains Audio Output Configuration and Audio Decoder Configuration with 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 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.

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

Test Result:

Note: If no **GetStreamUri** (Media Service) corresponding to detected RTSP session found, the test will be assumed as NOT DETECTED.

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:
 - [S3] It has RTSP 200 response code AND
 - [S4] SDP packet contains media type "audio" (m=audio) with session attribute "sendonly" (a=sendonly) and sessions attribute "rtptime" with encoding name "PCMU" AND
- There is Client **RTSP SETUP** request in Test Procedure that fulfills the following requirements:
 - [S5] It is invoked for the same Device as the response for **RTSP DESCRIBE** request AND
 - [S6] It is invoked after the Client **RTSP DESCRIBE** request AND
 - [S7] 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
 - [S8] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP SETUP** request fulfills the following requirements:
 - [S9] It has RTSP 200 response code AND

- There is a Device response to the **GetStreamUri** request 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 **trt:MediaUri** 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.

5.1.5 G.726 AUDIO BACKCHANNEL STREAMING

Test Label: Audio Backchannel Streaming - G.726

Test Case ID: AUDIOBACKCHANNELSTREAMING-3

Feature Under Test: G.726 Audio Backchannel Streaming
(AudioBackchannelStreaming_G726AudioBackchannelStreaming)

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 audio backchannel streaming with G.726 encoding.
- Device supports G.726 encoding for Audio Outputs.

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

1. Client invokes **GetStreamUri** request message for media profile that contains Audio Output Configuration and Audio Decoder Configuration with 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 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.

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

Test Result:

Note: If no **GetStreamUri** (Media Service) corresponding to detected RTSP session found, the test will be assumed as NOT DETECTED.

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:
 - [S3] It has RTSP 200 response code AND
 - [S4] SDP packet contains media type "audio" (m=audio) with session attribute "sendonly" (a=sendonly) and sessions attribute "rtpmap" with encoding name "G726-*" AND
- There is Client **RTSP SETUP** request in Test Procedure that fulfills the following requirements:
 - [S5] It is invoked for the same Device as the response for **RTSP DESCRIBE** request AND
 - [S6] It is invoked after the Client **RTSP DESCRIBE** request AND
 - [S7] 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
 - [S8] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP SETUP** request fulfills the following requirements:
 - [S9] It has RTSP 200 response code AND
- There is a Device response to the **GetStreamUri** request 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 **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 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.

5.1.6 AAC AUDIO BACKCHANNEL STREAMING

Test Label: Audio Backchannel Streaming - AAC

Test Case ID: AUDIOBACKCHANNELSTREAMING-4

Feature **Under** **Test:** AAC Audio Backchannel Streaming
(AudioBackchannelStreaming_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 audio backchannel streaming with AAC encoding.
- Device supports AAC encoding for Audio Outputs.

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

1. Client invokes **GetStreamUri** request message for media profile that contains Audio Output Configuration and Audio Decoder Configuration with 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 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.

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

Test Result:

Note: If no **GetStreamUri** (Media Service) corresponding to detected RTSP session found, the test will be assumed as NOT DETECTED.

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:
 - [S3] It has RTSP 200 response code AND
 - [S4] 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 for the same Device as the response for **RTSP DESCRIBE** request AND
 - [S6] It is invoked after the Client **RTSP DESCRIBE** request AND
 - [S7] 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
 - [S8] **Require** tag in RTSP header contains "www.onvif.org/ver20/backchannel" AND
- Device response to the **RTSP SETUP** request fulfills the following requirements:
 - [S9] It has RTSP 200 response code AND
- There is a Device response to the **GetStreamUri** request 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 **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 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.

5.2 Get Audio Decoder Configurations List Test Cases

5.2.1 Feature Level Requirement:

Validated Feature: Get Audio Decoder Configurations (GetAudioDecoderConfigurationsList)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.2.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a complete list of Audio Decoders.
2. Client is considered as supporting Get Audio Decoder Configurations List if the following conditions are met:
 - Client is able to list available Get Audio Decoder Configurations List using **GetAudioDecoderConfigurations** operation.
3. Client is considered as NOT supporting Get Audio Decoder Configurations List if ANY of the following is TRUE:
 - No valid responses for **GetAudioDecoderConfigurations** request.

5.2.3 GET AUDIO DECODER CONFIGURATIONS

Test Label: Get Audio Decoder Configurations List - Get Audio Decoder Configurations

Test Case ID: GETAUDIODECODERCONFIGURATIONSLIST-1

Feature Under Test: Get Audio Decoder Configurations
(GetAudioDecoderConfigurationsList_GetAudioDecoderConfigurations)

Test Purpose: To verify that list of all audio decoder configurations items 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 present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioDecoderConfigurations** request message to retrieve a list of all 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 **trt:GetAudioDecoderConfigurations** AND
- Device response to the **GetAudioDecoderConfigurations** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetAudioDecoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

5.3 Get Audio Output Configurations List Test Cases

5.3.1 Feature Level Requirement:

Validated Feature: Get Audio Output Configurations (GetAudioOutputConfigurationsList)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.3.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a complete list of Audio Outputs.
2. Client is considered as supporting Get Audio Output Configurations List if the following conditions are met:
 - Client is able to list available Get Audio Output Configurations List using **GetAudioOutputConfigurations** operation.
3. Client is considered as NOT supporting Get Audio Output Configurations List if ANY of the following is TRUE:
 - No valid responses for **GetAudioOutputConfigurations** request.

5.3.3 GET AUDIO OUTPUT CONFIGURATIONS

Test Label: Get Audio Output Configurations List - Get Audio Output Configurations

Test Case ID: GETAUDIOOUTPUTCONFIGURATIONSLIST-1

Feature Under Test: Get Audio Output Configurations
(GetAudioOutputConfigurationsList_GetAudioOutputConfigurations)

Test Purpose: To verify that list of all audio output configurations items 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 present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioOutputConfigurations** request message to retrieve a list of all 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 **trt:GetAudioOutputConfigurations** AND
- Device response to the **GetAudioOutputConfigurations** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetAudioOutputConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

5.4 Get Audio Outputs List Test Cases

5.4.1 Feature Level Requirement:

Validated Feature: Get Audio Outputs (GetAudioOutputsList)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.4.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a complete list of Audio Outputs.
2. Client is considered as supporting Get Audio Outputs List if the following conditions are met:

- Client is able to list available Get Audio Outputs List using **GetAudioOutputs** operation (Media Service or Device IO Service).
3. Client is considered as NOT supporting Get Audio Outputs List if ANY of the following is TRUE:
- No valid responses for **GetAudioOutputs** request (Media Service or Device IO Service).

5.4.3 GET AUDIO OUTPUTS

Test Label: Get Audio Outputs List - Get Audio Outputs

Test Case ID: GETAUDIOOUTPUTSLIST-1

Feature Under Test: Get Audio Outputs (GetAudioOutputsList_GetAudioOutputs)

Test Purpose: To verify that list of all audio outputs items provided by Device is received by Client using the **GetAudioOutputs** operation (Media Service or Device IO Service).

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioOutputs** operation (Media Service or Device IO Service) present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioOutputs** request message (Media Service or Device IO Service) to retrieve a list of all audio outputs from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioOutputsResponse** message.

Test Result:

PASS -

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

- [S3] **soapenv:Body** element has child element **trt:GetAudioOutputsResponse**.

FAIL -

- The Client failed PASS criteria.

5.5 Get Audio Decoder Configuration Test Cases

5.5.1 Feature Level Requirement:

Validated Feature: Get Audio Decoder Configuration (GetAudioDecoderConfiguration)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.5.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve an Audio Decoder Configuration.
2. Client is considered as supporting Get Audio Decoder Configuration if the following conditions are met:
 - Client is able to get Audio Decoder Configuration using **GetAudioDecoderConfiguration** operation OR Client supports `get_audio_decoder_configurations_list.get_audio_decoder_configurations` feature.
3. Client is considered as NOT supporting Get Audio Decoder Configuration if ANY of the following is TRUE:
 - No valid responses for **GetAudioDecoderConfiguration** request.

5.5.3 GET AUDIO DECODER CONFIGURATION

Test Label: Get Audio Decoder Configuration - Get Audio Decoder Configuration

Test Case ID: GETAUDIODECODERCONFIGURATION-1

Feature Under Test: Get Audio Decoder Configuration
(GetAudioDecoderConfiguration_GetAudioDecoderConfigurationFeature)

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

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioDecoderConfiguration** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioDecoderConfiguration** request message to retrieve audio decoder configuration for specified audio decoder configuration from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioDecoderConfigurationResponse** message.

Test Result:**PASS -**

- Client **GetAudioDecoderConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetAudioDecoderConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:GetAudioDecoderConfiguration** AND
- Device response to the **GetAudioDecoderConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetAudioDecoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.6 Get Audio Output Configuration Test Cases

5.6.1 Feature Level Requirement:

Validated Feature: Get Audio Output Configuration (GetAudioOutputConfiguration)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.6.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve an Audio Output Configuration.
2. Client is considered as supporting Get Audio Output Configuration if the following conditions are met:
 - Client is able to get Audio Output Configuration using **GetAudioOutputConfiguration** operation (Media Service OR Device IO Service) OR Client supports `get_audio_output_configurations_list.get_audio_output_configurations` feature.
3. Client is considered as NOT supporting Get Audio Output Configuration if ANY of the following is TRUE:
 - No valid responses for **GetAudioOutputConfiguration** request.

5.6.3 GET AUDIO OUTPUT CONFIGURATION

Test Label: Get Audio Output Configuration - Get Audio Output Configuration

Test Case ID: GETAUDIOOUTPUTCONFIGURATION-1

Feature Under Test: Get Audio Output Configuration
(GetAudioOutputConfiguration_GetAudioOutputConfigurationFeature)

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

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioOutputConfiguration** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioOutputConfiguration** request message to retrieve audio output configuration for specified audio output configuration from the Device.
2. Device responds with code HTTP 200 OK and **GetAudioOutputConfigurationResponse** message.

Test Result:**PASS -**

- Client **GetAudioOutputConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetAudioOutputConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:GetAudioOutputConfiguration** AND
- Device response to the **GetAudioOutputConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetAudioOutputConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.7 Profile Configuration for Audio Backchannel Test Cases

5.7.1 Feature Level Requirement:

Validated Feature: Profile Configuration for Audio Backchannel
(ProfileConfigurationForAudioBackchannel)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.7.2 Expected Scenarios Under Test:

1. Client connects to Device to configure profile for Audio Backchannel streaming.
2. Client is considered as supporting Profile Configuration for Audio Backchannel details if the following conditions are met:
 - Client is able to get compatible Audio Output Configuration using **GetCompatibleAudioOutputConfigurations** operation for specified profile AND
 - Client is able to add or replace Audio Output Configuration in media profile using **AddAudioOutputConfiguration** operation for specified audio output configuration and compatible with specified profile AND
 - Client may be able to remove Audio Output Configuration from media profile using **RemoveAudioOutputConfiguration** operation for specified profile AND
 - Client is able to get compatible Audio Decoder Configuration using **GetCompatibleAudioDecoderConfigurations** operation for specified profile AND
 - Client is able to add or replace Audio Decoder Configuration in media profile using **AddAudioDecoderConfiguration** operation for specified audio decoder configuration and compatible with specified profile AND
 - Client may be able to remove Audio Decoder Configuration from media profile using **RemoveAudioDecoderConfiguration** operation for specified profile.
3. Client is considered as NOT supporting Profile Configuration for Audio Backchannel if ANY of the following is TRUE:
 - No valid responses for **GetCompatibleAudioOutputConfigurations** request OR
 - No valid responses for **AddAudioOutputConfiguration** request OR

- Client tries to invoke **AddAudioOutputConfiguration** request without **GetCompatibleAudioOutputConfigurations** request for specified profile OR
- Detected RemoveAudioOutputConfiguration request attempt have failed OR
- No valid responses for **GetCompatibleAudioDecoderConfigurations** request OR
- No valid responses for **AddAudioDecoderConfiguration** request OR
- Client tries to invoke **AddAudioDecoderConfiguration** request without **GetCompatibleAudioDecoderConfigurations** request for specified profile OR
- Detected RemoveAudioDecoderConfiguration request attempt has failed.

5.7.3 GET COMPATIBLE AUDIO OUTPUT CONFIGURATIONS

Test Label: Profile Configuration for Audio Backchannel - Get Compatible Audio Output Configurations

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-1

Feature Under Test: Get Compatible Audio Output Configurations (ProfileConfigurationForAudioBackchannel_GetCompatibleAudioOutputConfigurations)

Test Purpose: To verify that compatible audio output configurations provided by Device for specified media profile is received by Client using the **GetCompatibleAudioOutputConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetCompatibleAudioOutputConfigurations** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetCompatibleAudioOutputConfigurations** request message to retrieve compatible audio output configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetCompatibleAudioOutputConfigurationsResponse** message.

Test Result:

PASS -

- Client **GetCompatibleAudioOutputConfigurations** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetCompatibleAudioOutputConfigurations** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:GetCompatibleAudioOutputConfigurations** AND
- Device response to the **GetCompatibleAudioOutputConfigurations** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetCompatibleAudioOutputConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

5.7.4 ADD AUDIO OUTPUT CONFIGURATION

Test Label: Profile Configuration for Audio Backchannel - Add Audio Output Configuration

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-2

Feature Under Test: Add Audio Output Configuration
(ProfileConfigurationForAudioBackchannel_AddAudioOutputConfiguration)

Test Purpose: To verify that Client is able to add or replace audio output configurations on a Device for specified audio output configuration and compatible with specified profile using the **AddAudioOutputConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddAudioOutputConfiguration** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetCompatibleAudioOutputConfigurations** request message to retrieve compatible audio output configurations for specified media profile from the Device.

2. Device responds with code HTTP 200 OK and **GetCompatibleAudioOutputConfigurationsResponse** message.
3. Client invokes **AddAudioOutputConfiguration** request message to add or replace audio output configurations for specified media profile and with audio output configuration token that was received in **GetCompatibleAudioOutputConfigurationsResponse** message from the Device for the same media profile.
4. Device responds with code HTTP 200 OK and **AddAudioOutputConfigurationResponse** message.

Test Result:**PASS -**

- Client **AddAudioOutputConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **AddAudioOutputConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:AddAudioOutputConfiguration** AND
- Device response to the **AddAudioOutputConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:AddAudioOutputConfigurationResponse** AND
- There is Client **GetCompatibleAudioOutputConfigurations** request in Test Procedure that fulfills the following requirements:
 - [S4] It is invoked for the same Device the response for **AddAudioOutputConfiguration** request AND
 - [S5] It is invoked before the Client **AddAudioOutputConfiguration** request AND
 - [S6] **trt:ProfileToken** element value is equal to **trt:ProfileToken** element from the **AddAudioOutputConfiguration** request AND
 - [S7] It is the last **GetCompatibleAudioOutputConfigurations** request which corresponds [S4], [S5] AND [S6] AND
- Device response to the **GetCompatibleAudioOutputConfigurations** request fulfills the following requirements:

- [S8] It has HTTP 200 response code AND
- [S9] **soapenv:Body** element has child element **trt:GetCompatibleAudioOutputConfigurationsResponse** AND
- [S10] It contains **trt:Configurations/@token** attribute value equal to **trt:ConfigurationToken** from the **AddAudioOutputConfiguration** request messages.

FAIL -

- The Client failed PASS criteria.

5.7.5 REMOVE AUDIO OUTPUT CONFIGURATION

Test Label: Profile Configuration for Audio Backchannel - Remove Audio Output Configuration

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-3

Feature Under Test: Remove Audio Output Configuration
(ProfileConfigurationForAudioBackchannel_RemoveAudioOutputConfiguration)

Test Purpose: To verify that Client is able to remove audio output configurations on a Device from specified profile using the **RemoveAudioOutputConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **RemoveAudioOutputConfiguration** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **RemoveAudioOutputConfiguration** request message to remove audio output configurations from specified media profile on the Device.
2. Device responds with code HTTP 200 OK and **RemoveAudioOutputConfigurationResponse** message.

Test Result:**PASS -**

- Client **RemoveAudioOutputConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND

- Client **RemoveAudioOutputConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:RemoveAudioOutputConfiguration** AND
- Device response to the **RemoveAudioOutputConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:RemoveAudioOutputConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.7.6 GET COMPATIBLE AUDIO DECODER CONFIGURATIONS

Test Label: Profile Configuration for Audio Backchannel - Get Compatible Audio Decoder Configurations

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-4

Feature Under Test: Get Compatible Audio Decoder Configurations (ProfileConfigurationForAudioBackchannel_GetCompatibleAudioDecoderConfigurations)

Test Purpose: To verify that compatible audio decoder configurations provided by Device for specified media profile is received by Client using the **GetCompatibleAudioDecoderConfigurations** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetCompatibleAudioDecoderConfigurations** operation present.
- Device supports Audio Decoders.

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

1. Client invokes **GetCompatibleAudioDecoderConfigurations** request message to retrieve compatible audio decoder configurations for specified media profile from the Device.

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

Test Result:**PASS -**

- Client **GetCompatibleAudioDecoderConfigurations** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetCompatibleAudioDecoderConfigurations** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:GetCompatibleAudioDecoderConfigurations** AND
- Device response to the **GetCompatibleAudioDecoderConfigurations** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetCompatibleAudioDecoderConfigurationsResponse**.

FAIL -

- The Client failed PASS criteria.

5.7.7 ADD AUDIO DECODER CONFIGURATION

Test Label: Profile Configuration for Audio Backchannel - Add Audio Decoder Configuration

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-5

Feature Under Test: Add Audio Decoder Configuration
(ProfileConfigurationForAudioBackchannel_AddAudioDecoderConfiguration)

Test Purpose: To verify that Client is able to add or replace audio decoder configurations on a Device for specified audio decoder configuration and compatible with specified profile using the **AddAudioDecoderConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddAudioDecoderConfiguration** operation present.
- Device supports Audio Decoders.

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

1. Client invokes **GetCompatibleAudioDecoderConfigurations** request message to retrieve compatible audio decoder configurations for specified media profile from the Device.
2. Device responds with code HTTP 200 OK and **GetCompatibleAudioDecoderConfigurationsResponse** message.
3. Client invokes **AddAudioDecoderConfiguration** request message to add or replace audio decoder configurations for specified media profile and with audio decoder configuration token that was received in **GetCompatibleAudioDecoderConfigurationsResponse** message from the Device for the same media profile.
4. Device responds with code HTTP 200 OK and **AddAudioDecoderConfigurationResponse** message.

Test Result:**PASS -**

- Client **AddAudioDecoderConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **AddAudioDecoderConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:AddAudioDecoderConfiguration** AND
- Device response to the **AddAudioDecoderConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:AddAudioDecoderConfigurationResponse** AND
- There is Client **GetCompatibleAudioDecoderConfigurations** request in Test Procedure that fulfills the following requirements:
 - [S4] It is invoked for the same Device as the response for **AddAudioDecoderConfiguration** request AND
 - [S5] It is invoked before the Client **AddAudioDecoderConfiguration** request AND
 - [S6] **trt:ProfileToken** element value is equal to **trt:ProfileToken** element from the **AddAudioDecoderConfiguration** request AND

- [S7] It is the last **GetCompatibleAudioDecoderConfigurations** request which corresponds [S4], [S5] AND [S6] AND
- Device response to the **GetCompatibleAudioDecoderConfigurations** request fulfills the following requirements:
 - [S8] It has HTTP 200 response code AND
 - [S9] **soapenv:Body** element has child element **trt:GetCompatibleAudioDecoderConfigurationsResponse** AND
 - [S10] It contains **trt:Configurations/@token** attribute value equal to **trt:ConfigurationToken** from the **AddAudioDecoderConfiguration** request messages.

FAIL -

- The Client failed PASS criteria.

5.7.8 REMOVE AUDIO DECODER CONFIGURATION

Test Label: Profile Configuration for Audio Backchannel - Remove Audio Decoder Configuration

Test Case ID: PROFILECONFIGURATIONFORAUDIOBACKCHANNEL-6

Feature Under Test: Remove Audio Decoder Configuration
(ProfileConfigurationForAudioBackchannel_RemoveAudioDecoderConfiguration)

Test Purpose: To verify that Client is able to remove audio decoder configurations on a Device from specified profile using the **RemoveAudioDecoderConfiguration** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **RemoveAudioDecoderConfiguration** operation present.
- Device supports Audio Decoders.

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

1. Client invokes **RemoveAudioDecoderConfiguration** request message to remove audio decoder configurations from specified media profile on the Device.
2. Device responds with code HTTP 200 OK and **RemoveAudioDecoderConfigurationResponse** message.

Test Result:

PASS -

- Client **RemoveAudioDecoderConfiguration** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **RemoveAudioDecoderConfiguration** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:RemoveAudioDecoderConfiguration** AND
- Device response to the **RemoveAudioDecoderConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:RemoveAudioDecoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.8 Configure Audio Decoder Configuration Test Cases

5.8.1 Feature Level Requirement:

Validated Feature: Configure Audio Decoder Configuration (SetAudioDecoderConfiguration)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.8.2 Expected Scenarios Under Test:

1. Client connects to Device to change Audio Decoder Configuration settings.

2. Client is considered as supporting Configure Audio Decoder Configuration if the following conditions are met:
 - Client is able to change Audio Decoder Configuration settings using **SetAudioDecoderConfiguration** operation.
3. Client is considered as NOT supporting Configure Audio Decoder Configuration if ANY of the following is TRUE:
 - No valid responses for **SetAudioDecoderConfiguration** request.

5.8.3 SET AUDIO DECODER CONFIGURATION

Test Label: Configure Audio Decoder Configuration - Set Audio Decoder Configuration

Test Case ID: SETAUDIODECODERCONFIGURATION-1

Feature Under Test: Set Audio Decoder Configuration
(SetAudioDecoderConfiguration_SetAudioDecoderConfigurationRequest)

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 present.
- Device supports Audio Outputs.

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

1. Client invokes **SetAudioDecoderConfiguration** request message to change 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 **trt:SetAudioDecoderConfiguration** AND
- Device response to the **SetAudioDecoderConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:SetAudioDecoderConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

5.9 Configure Audio Output Configuration Test Cases

5.9.1 Feature Level Requirement:

Validated Feature: Configure Audio Output Configuration (SetAudioOutputConfiguration)

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

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

5.9.2 Expected Scenarios Under Test:

1. Client connects to Device to change audio output configuration.
2. Client is considered as supporting Configure Audio Output Configuration if the following conditions are met:
 - Client is able to retrieve audio output configuration options using **GetAudioOutputConfigurationOptions** operation AND
 - Client is able to change audio output configuration settings using **SetAudioOutputConfiguration** operation.

3. Client is considered as NOT supporting Configure Audio Output Configuration if ANY of the following is TRUE:
 - No valid responses for **GetAudioOutputConfigurationOptions** request OR
 - No valid responses for **SetAudioOutputConfiguration** request.

5.9.3 GET AUDIO OUTPUT CONFIGURATION OPTIONS

Test Label: Configure Audio Output Configuration - Get Audio Output Configuration Options

Test Case ID: SETAUDIOOUTPUTCONFIGURATION-1

Feature Under Test: Get Audio Output Configuration Options
(SetAudioOutputConfiguration_GetAudioOutputConfigurationOptions)

Test Purpose: To verify that Client is able to get audio output configuration options provided by Device using the **GetAudioOutputConfigurationOptions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetAudioOutputConfigurationOptions** operation present.
- Device supports Audio Outputs.

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

1. Client invokes **GetAudioOutputConfigurationOptions** request message to retrieve audio output configuration options for 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 **trt:GetAudioOutputConfigurationOptions** AND

- Device response to the **GetAudioOutputConfigurationOptions** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:GetAudioOutputConfigurationOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

5.9.4 SET AUDIO OUTPUT CONFIGURATION

Test Label: Configure Audio Output Configuration - Set Audio Output Configuration

Test Case ID: SETAUDIOOUTPUTCONFIGURATION-2

Feature Under Test: Set Audio Output Configuration
(SetAudioOutputConfiguration_SetAudioOutputConfigurationRequest)

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 present.
- Device supports Audio Outputs.

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

1. Client invokes **SetAudioOutputConfiguration** request message to change 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 **trt:SetAudioOutputConfiguration** AND
- Device response to the **SetAudioOutputConfiguration** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **trt:SetAudioOutputConfigurationResponse**.

FAIL -

- The Client failed PASS criteria.

6 Test Cases for Imaging

6.1 Get Imaging Capabilities Test Cases

6.1.1 Feature Level Requirement:

Validated Feature: Get Imaging Capabilities (GetImagingCapabilities)

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

Required Number of Devices: 1

6.1.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a imaging capabilities.
2. Client is considered as supporting Get Imaging Capabilities if the following conditions are met:
 - Client is able to retrieve a imaging capabilities using **GetCapabilities** operation OR **GetServiceCapabilities** operation (Imaging Service) OR supports `get_services_capabilities.get_services` feature.
3. Client is considered as NOT supporting Get Imaging Capabilities if ANY of the following is TRUE:
 - No valid responses for **GetCapabilities** request if detected AND Device supportes GetCapabilities feature OR
 - No valid responses for **GetServiceCapabilities** request (Imaging Service) if detected AND Device supportes GetServices feature
 - No valid responses for **GetCapabilities** request AND no valid responses for **GetServiceCapabilities** request (Imaging Service) AND `get_services_capabilities.get_services` feature is not supported by Client.

6.1.3 GET CAPABILITIES

Test Label: Get Imaging Capabilities - Get Capabilities

Test Case ID: GETIMAGINGCAPABILITIES-1

Feature Under Test: Get Imaging Capabilities using Get Capabilities (GetImagingCapabilities_GetImgCapabilities)

Test Purpose: To verify that imaging capabilities provided by Device is received by Client using the **GetCapabilities** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetCapabilities** operation with **tds:Category** element equal to "All" OR "Imaging" OR without any **tds:Category** element present.
- Device supports Imaging Service.

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

1. Client invokes **GetCapabilities** request message with **tds:Category** element equal to "All" OR "Imaging" OR without any **tds:Category** element to retrieve imaging capabilities from the Device.
2. Device responds with code HTTP 200 OK and **GetCapabilitiesResponse** message.

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] **soapenv:Body** element has child element **tds:GetCapabilities** AND
 - [S2] IF it contains any **tds:Category** element THEN it contains **tds:Category** element equal to "All" OR "Imaging" AND
- Device response on the **GetCapabilities** request fulfills the following requirements:
 - [S3] It has HTTP 200 response code AND
 - [S4] **soapenv:Body** element has child element **tds:GetCapabilitiesResponse**.

FAIL -

- The Client failed PASS criteria.

6.1.4 GET SERVICE CAPABILITIES

Test Label: Get Imaging Capabilities - Get Service Capabilities

Test Case ID: GETIMAGINGCAPABILITIES-2

Feature Under Test: Get Imaging Capabilities using Get Service Capabilities (GetImagingCapabilities_GetImgServiceCapabilities)

Test Purpose: To verify that imaging capabilities provided by Device is received by Client using the **GetServiceCapabilities** operation.

Pre-Requisite:

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

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

1. Client invokes **GetServiceCapabilities** request message to retrieve imaging capabilities from the Device.
2. Device responds with code HTTP 200 OK and **GetServiceCapabilitiesResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

7 Test Cases for OSD for Media

7.1 Get OSD Configuration Test Cases

7.1.1 Feature Level Requirement:

Validated Feature: Get OSD Configuration (GetOSD)

Check Condition based on Device Features: TO BE DISCUSSED

Required Number of Devices: 1

7.1.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a OSD configuration.
2. Client is considered as supporting Get OSD Configuration if the following conditions are met:
 - Client is able to retrieve a OSD configuration using **GetOSD** operation.
3. Client is considered as NOT supporting Get OSD Configuration if ANY of the following is TRUE:
 - No valid responses for **GetOSD** request.

7.1.3 GET OSD

Test Label: Get OSD - Get OSD

Test Case ID: GETOSD-1

Feature Under Test: Get OSD (GetOSD_GetOsd)

Test Purpose: To verify that OSD list for Device is received by Client using the **GetOSD** operation.

Pre-Requisite:

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

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

1. Client invokes **GetOSD** request message to retrieve OSD configuration from the Device.
2. Device responds with code HTTP 200 OK and **GetOSDResponse** message.

Test Result:**PASS -**

- Client **GetOSD** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetOSD** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **trt:GetOSD** AND
 - [S2] **trt:OSDToken** element has non-empty string value of specific OSD token AND
- Device response on the **GetOSD** request fulfills the following requirements:
 - [S3] It has HTTP 200 response code AND
 - [S4] **soapenv:Body** element has child element **trt:GetOSDResponse**.

FAIL -

- The Client failed PASS criteria.

7.2 Get OSD List Test Cases

7.2.1 Feature Level Requirement:

Validated Feature: Get OSD List (GetOSDs)

Check Condition based on Device Features: TO BE DISCUSSED

Required Number of Devices: 1

7.2.2 Expected Scenarios Under Test:

1. Client connects to Device to retrieve a OSD list.
2. Client is considered as supporting Get OSD List if the following conditions are met:
 - Client is able to retrieve a OSD list using **GetOSDs** operation.
3. Client is considered as NOT supporting Get OSD List if ANY of the following is TRUE:

- No valid responses for **GetOSDs** request.

7.2.3 GET OSDS

Test Label: Get OSDs - Get OSDs

Test Case ID: GETOSDS-1

Feature Under Test: Get OSDs (GetOSDs_GetOsds)

Test Purpose: To verify that OSD list for Device 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 present.
- Device supports Media Service.

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

1. Client invokes **GetOSDs** request message to retrieve OSD list 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 **trt:GetOSDs** AND
 - If it contains **trt:ConfigurationToken** element then it fulfills the following requirements (else skip the check):
 - [S2] **trt:ConfigurationToken** element has non-empty string value of specific video source configuraton token 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 **trt:GetOSDsResponse**.

FAIL -

- The Client failed PASS criteria.

7.3 OSD Configuration Test Cases

7.3.1 Feature Level Requirement:

Validated Feature: OSD Configuration (SetOSD)

Check Condition based on Device Features: TO BE DISCUSSED

Required Number of Devices: 1

7.3.2 Expected Scenarios Under Test:

1. Client connects to Device to change OSD settings.
2. Client is considered as supporting OSD Configuration if the following conditions are met:
 - Client is able to retrieve a OSD options using **GetOSDOptions** operation AND
 - Client is able to change a OSD settings using **SetOSD** operation.
3. Client is considered as NOT supporting OSD Configuration if ANY of the following is TRUE:
 - No valid responses for **GetOSDOptions** request OR
 - No valid responses for **SetOSD** request.

7.3.3 GET OSD OPTIONS

Test Label: OSD Configuration - Get OSD Options

Test Case ID: SETOSD-1

Feature Under Test: Get OSD Options (SetOSD_GetOsdOptions)

Test Purpose: To verify that OSD options for 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 present.

- Device supports Media Service.

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

1. Client invokes **GetOSDOptions** request message to retrieve OSD options for specified Video Source Configuration 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 **trt:GetOSDOptions** AND
 - [S2] **trt:ConfigurationToken** element has non-empty string value of specific video source configuraton token AND
- Device response on the **GetOSDOptions** request fulfills the following requirements:
 - [S3] It has HTTP 200 response code AND
 - [S4] **soapenv:Body** element has child element **trt:GetOSDOptionsResponse**.

FAIL -

- The Client failed PASS criteria.

7.3.4 SET OSD

Test Label: OSD Configuration - Set OSD

Test Case ID: SETOSD-2

Feature Under Test: Set OSD (SetOSD_SetOsd)

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

Pre-Requisite:

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

- Device supports Media Service.

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

1. Client invokes **GetOSDOptions** request message to retrieve OSD options for specified Video Source Configuration from the Device.
2. Device responds with code HTTP 200 OK and **GetOSDOptionsResponse** message.
3. Client invokes **SetOSD** request message to change OSD settings for specified OSD which are correspond to the received options on the Device.
4. 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 **trt: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 **trt:SetOSDResponse** AND
- There is a Client **GetOSDOptions** request in Test Procedure fulfills the following requirements:
 - [S4] It invoked for the same Device as for the Client **SetOSD** request AND
 - [S5] It invoked before the Client **SetOSD** request AND
 - [S6] **trt:ConfigurationToken** element value is equal to **trt:OSD/tt:VideoSourceConfigurationToken** element from the **SetOSD** request AND
- Device response on the **GetOSDOptions** request fulfills the following requirements:
 - [S7] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

8 Test Cases for Security Configuration

8.1 Enabled TLS Versions Configuration Test Cases

8.1.1 Feature Level Requirement:

Validated Feature: Enabled TLS Versions Configuration (EnabledTLSVersionsConfiguration)

Check Condition based on Device Features: Enabled TLS Versions (Security Configuration Service) is supported by the Device.

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

8.1.2 Expected Scenarios Under Test:

1. Client connects to Device configure enabled TLS versions on Device.
2. Client is considered as supporting Enabled TLS Versions Configuration if the following conditions are met:
 - Client is able to retrieve supported TLS versions using **GetServices** operation with IncludeCapability = true or using **GetServiceCapabilities** operation for Security Configuration Service if Device supports Enabled TLS Versions feature AND
<listitem></listitem>
 - Client is able to setup enabled TLS versions using **SetEnabledTLSVersions** operation if Device supports Enabled TLS Versions feature.
3. Client is considered as NOT supporting Enabled TLS Versions Configuration if ANY of the following is TRUE:
 - No valid responses for **GetServices** request with IncludeCapability = true or for **GetServiceCapabilities** request for Security Configuration Service if detected if Device supports Enabled TLS Versions feature OR

- No valid responses for **SetEnabledTLSVersions** request if detected if Device supports Enabled TLS Versions feature OR
- No valid responses for **GetEnabledTLSVersions** request if detected if Device supports Enabled TLS Versions feature.

8.1.3 Get Enabled TLS Versions

Test Case ID: ENABLEDTLSVERSIONSCONFIGURATION-1

Feature Under Test: Get Enabled TLS Versions
(EnabledTLSVersionsConfiguration_GetEnabledTLSVersions)

Test Purpose: To verify that Client is able to get currently enabled TLS versions from Device using **GetEnabledTLSVersions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetEnabledTLSVersions** operation present.
- Device supports Security Configuration Service.
- Device supports Enabled TLS Versions (Security Configuration Service).

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

1. Client invokes **GetEnabledTLSVersions** request message to get currently enabled TLS versions from Device.
2. Device responds with code HTTP 200 OK and **GetEnabledTLSVersionsResponse** message.

Test Result:

PASS -

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

- [S3] **soapenv:Body** element has child element **tas:GetEnabledTLSVersionsResponse**.

FAIL -

- The Client failed PASS criteria.

8.1.4 Set Enabled TLS Versions

Test Case ID: ENABLEDTLSVERSIONSCONFIGURATION-2

Feature **Under** **Test:** Set Enabled TLS Versions
(EnabledTLSVersionsConfiguration_SetEnabledTLSVersions)

Test Purpose: To verify that Client is able to setup enabled TLS versions on Device using **SetEnabledTLSVersions** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **SetEnabledTLSVersions** operation present.
- Device supports Security Configuration Service.
- Device supports Enabled TLS Versions (Security Configuration Service).

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

1. Client invokes **GetServices** request message with IncludeCapability = true or **GetServiceCapabilities** request message for the Security Configuration Service request message to get supported TLS versions from a Device.
2. Device responds with code HTTP 200 OK and **GetServicesResponse** message or **GetServiceCapabilitiesResponse** message with Security Configuration Service capabilities.
3. Client invokes **SetEnabledTLSVersions** request message with non empty list to configure enabled TLS versions on a Device.

Test Result:

PASS -

- Client **SetEnabledTLSVersions** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **SetEnabledTLSVersions** request in Test Procedure fulfills the following requirements:

- [S1] **soapenv:Body** element has child element **tas:SetEnabledTLSVersions** AND
- [S2] **tas:Versions** element contains at least one TLS version AND
- Device response on the **SetEnabledTLSVersions** request fulfills the following requirements:
 - [S3] It has HTTP 200 response code AND
 - [S4] **soapenv:Body** element has child element **tas:SetEnabledTLSVersionsResponse** AND
- There is a Client **GetServices** request or **GetServiceCapabilities** request in Test Procedure that fulfills the following requirements:
 - [S5] It invoked before **SetEnabledTLSVersions** request AND
 - If **GetServices** was detected:
 - [S6] **soapenv:Body** element has child element **tds:GetServices** AND
 - [S7] **tds:IncludeCapability** element is equal to true AND
 - If **GetServiceCapabilities** was detected:
 - [S8] **soapenv:Body** element has child element **tas:GetServiceCapabilities** AND
- If **GetServices** was detected Device response on the **GetServices** request fulfills the following requirements:
 - [S9] It has HTTP 200 response code AND
 - [S10] **soapenv:Body** element has child element **tds:GetServicesResponse** AND
- If **GetServiceCapabilities** was detected Device response on the **GetServiceCapabilities** request fulfills the following requirements:
 - [S11] It has HTTP 200 response code AND
 - [S12] **soapenv:Body** element has child element **tas:GetServiceCapabilitiesResponse**.

FAIL -

- The Client failed PASS criteria.

9 Test Cases for Operational State

9.1 Transition to Operational State Test Cases

9.1.1 Feature Level Requirement:

Validated Feature: Transition to Operational State (TransitionToOperationalState)

Check Condition based on Device Features: TO BE DISCUSSED

Required Number of Devices: 3

9.1.2 Expected Scenarios Under Test:

1. A Client connects to a Device in Factory Default State to invoke its transition into Operational State.
2. The Client is considered as supporting Transition to Operational State if the following conditions are met:
 - The Client is able to invoke the Device transition into the Operational State by using EITHER **CreateUsers** OR **SetUser** operations.
3. The Client is considered as NOT supporting Transition to Operational State if ANY of the following is TRUE:
 - No valid response to **CreateUsers** request OR
 - No valid response to **SetUser** request AND
 - **SetUser** request does not contain user with **Username** value contained in **GetUsers** response.

9.1.3 TRANSITION TO OPERATIONAL STATE BY CREATEUSERS

Test Label: Transition to Operational State by Create User

Test Case ID: TRANSITIONTOOPERATIONALSTATE-1

Feature Under Test: Transition to Operational State by CreateUsers
(TransitionToOperationalState_TransitionToOperationalStateByCreateUsers)

Test Purpose: To verify that a Client is able to invoke Device transition into Operational State using the **CreateUsers**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device in Factory Default state with **CreateUsers** operation without any authentication which contains User with "Administrator" user level present.

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

1. Client invokes **CreateUsers** request message without any authentication and with non-empty password to create a new admin user.
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] **soapenv:Body** element has child element **tds:CreateUsers** AND
 - [S2] It does not contain Digest Authentication part AND
 - [S3] It does not contain WS-Username Token Authentication part AND
 - It contains **tds:User** element which fulfills the following requirements:
 - [S4] **tt:Username** element has non-empty string value AND
 - [S5] It contains **tt:Password** element AND
 - [S6] **tt:Password** element has non-empty string value AND
 - [S7] **tt:UserLevel** element value equals "Administrator" AND
- Device response to the **CreateUsers** request fulfills the following requirements:
 - [S8] It has HTTP 200 response code AND
 - [S9] **soapenv:Body** element has child element **tds:CreateUsersResponse**

FAIL -

- The Client failed PASS criteria.

9.1.4 TRANSITION TO OPERATIONAL STATE BY SET USER

Test Label: Transition to Operational State by Set User

Test Case ID: TRANSITIONTOOPERATIONALSTATE-2

Feature Under Test: Transition to Operational State by SetUser
(TransitionToOperationalState_TransitionToOperationalStateBySetUser)

Test Purpose: To verify that a Client is able to invoke Device transition into Operational State using the **SetUser**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device in Factory Default state with **SetUser** operation without any authentication and with UserLevel is equal to "Administrator" present.

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

1. Client invokes **GetUsers** request message without any authentication to retrieve user list from Device.
2. Device responds with code HTTP 200 OK and **GetUsersResponse** message.
3. Client invokes **SetUser** request message without any authentication to modify the password of an existing admin user.
4. 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] **soapenv:Body** element has child element **tds:SetUser** AND
 - [S2] It does not contain Digest Authentication part AND
 - [S3] It does not contain WS-Username Token Authentication part AND

- It contains **tds:User** element which fulfills the following requirements:
 - [S4] **tt:Username** element has non-empty string value AND
 - [S5] It contains **tt:Password** element AND
 - [S6] **tt:Password** element has non-empty string value AND
 - [S7] **tt:UserLevel** element value equals "Administrator" AND
- Device response to the **SetUser** request fulfills the following requirements:
 - [S8] It has HTTP 200 response code AND
 - [S9] **soapenv:Body** element has child element **tds:SetUserResponse**
- There is a Client **GetUsers** request message in Test Procedure fulfills the following requirements:
 - [S10] It is invoked for the same Device as the response for the **SetUser** request AND
 - [S11] It is invoked before the Client **SetUser** request AND
 - [S12] It does not contain digest authentication part AND
 - [S13] It does not contain WS-username token authentication part AND
- Device response to the **GetUsers** request fulfills the following requirements:
 - [S14] It has HTTP 200 response code AND
 - [S15] **soapenv:Body** element has child element **tds:GetUsersResponse**
 - [S16] It contains **tt:User** element which fulfills the following requirements:
 - [S17] **tt:Username** element value equals to **tt:Username** value from the **SetUser** request AND
 - [S18] **UserLevel** element value equals "Administrator".

FAIL -

- The Client failed PASS criteria.

10 Test Cases for Firmware Upgrade

10.1 HTTP Firmware Upgrade Test Cases

10.1.1 Feature Level Requirement:

Validated Feature: Firmware Upgrade via HTTP (HTTPFirmwareUpgrade)

Check Condition based on Device Features: HTTP Firmware Upgrade is supported by Device.

Required Number of Devices: 1

10.1.2 Expected Scenarios Under Test:

1. Client connects to the Device to instruct it to prepare for upgrade using the StartFirmwareUpgrade operation.
2. Client sends the firmware image using HTTP POST to the upload URI provided by the Device in StartFirmwareUpgradeResponse.
3. Client is considered as supporting HTTP Firmware Upgrade if the following conditions are met:
 - Client is able to instruct the Device to prepare for upgrade using **StartFirmwareUpgrade** operation if Device supports HTTP Firmware Upgrade AND
 - Client is able to send the firmware image using **HTTP POST** if Device supports HTTP Firmware Upgrade.
4. Client is considered as NOT supporting HTTP Firmware Upgrade if ANY of the following is TRUE:
 - No valid responses for **StartFirmwareUpgrade** request if Device supports HTTP Firmware Upgrade OR
 - No valid **HTTP POST** request to the upload URI if Device supports HTTP Firmware Upgrade.
 - No valid responses for **HTTP POST** request to the upload URI with firmware image if Device supports HTTP Firmware Upgrade.

10.1.3 FIRMWARE UPGRADE VIA HTTP

Test Label: Firmware Upgrade via HTTP - Start Firmware Upgrade

Test Case ID: HTTPFIRMWAREUPGRADE-1

Feature Under Test: Start Firmware Upgrade (HTTPFirmwareUpgrade_StartFirmwareUpgrade)

Test Purpose: To verify that Client is able to upgrade the Device firmware via HTTP using the **StartFirmwareUpgrade** operation and **HTTP POST**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **StartFirmwareUpgrade** operation present.
- Device supports Http Firmware Upgrade.

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

1. Client invokes **StartFirmwareUpgrade** request message to instruct the Device to prepare for upgrade.
2. Device responds with code HTTP 200 OK and **StartFirmwareUpgradeResponse** message.
3. Client sends the firmware image using **HTTP POST** to the upload URI provided by the Device in StartFirmwareUpgradeResponse.
4. Device responds with code HTTP 200 OK message.

Test Result:

PASS -

- Client **StartFirmwareUpgrade** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **StartFirmwareUpgrade** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tds:StartFirmwareUpgrade** AND
- Device response on the **StartFirmwareUpgrade** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **tds:StartFirmwareUpgradeResponse**.
- There is **HTTP POST** request in Test Procedure fulfills the following requirements:
 - [S4] It invoked to address which equal to **tds:StartFirmwareUpgradeResponse/tds:UploadUri** value from the Device response to **StartFirmwareUpgrade** request AND

- [S5] It invoked after the Client **StartFirmwareUpgrade** request AND
- [S6] It contains HTTP Content-Type Header with value is equal to “application/octet-stream” AND
- Device response on the **HTTP POST** request fulfills the following requirements:
 - [S7] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

11 Test Cases for Backup and Restore

11.1 HTTP System Backup Test Cases

11.1.1 Feature Level Requirement:

Validated Feature: System Backup via HTTP (HTTPSystemBackup)

Check Condition based on Device Features: HTTP System Backup is supported by Device.

Required Number of Devices: 1

11.1.2 Expected Scenarios Under Test:

1. Client connects to the Device to retrieve URI from which a system backup may be downloaded using the `GetSystemUri` operation.

Client gets the backup system configurations using HTTP GET sent to the System Backup Uri provided by the Device in `GetSystemUriResponse`.

2. Client is considered as supporting HTTP System Backup if the following conditions are met:
 - Client is able to retrieve URI from Device for system backup using **GetSystemUri** operation if Device supports HTTP System Backup AND
 - Client is able to backup system configurations using **HTTP GET** if Device supports HTTP System Backup AND
3. Client is considered as NOT supporting HTTP System Backup if ANY of the following is TRUE:
 - No valid responses for **GetSystemUri** request if Device supports HTTP System Backup OR
 - No valid responses for **HTTP GET** request to the System Backup Uri if Device supports HTTP System Backup.

11.1.3 GET SYSTEM URIS

Test Label: System Backup via HTTP - Get System Uri

Test Case ID: HTTPSYSTEMBACKUP-1

Feature Under Test: Get System Uri (HTTPSystemBackup_GetSystemUri)

Test Purpose: To verify that Client is able to backup system configurations via HTTP using the **GetSystemUris** operation and **HTTP GET**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetSystemUris** operation present.
- Device supports HTTP System Backup.

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

1. Client invokes **GetSystemUris** request message to retrieve URI from which a system backup file may be downloaded.
2. Device responds with code HTTP 200 OK and **GetSystemUrisResponse** message.
3. Client retrieves the backup file using **HTTP GET** to the System Backup Uri provided by the Device in **GetSystemUrisResponse**.
4. Device responds with code HTTP 200 OK message and with backup file.

Test Result:

PASS -

- Client **GetSystemUris** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetSystemUris** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tds:GetSystemUris** AND
- Device response on the **GetSystemUris** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **tds:GetSystemUrisResponse**.
- There is **HTTP GET** request in Test Procedure that fulfills the following requirements:
 - [S4] It invoked to address which equal to **tds:GetSystemUrisResponse/tds:SystemBackupUri** value from the Device response to **GetSystemUris** request AND
 - [S5] It invoked after the Client **GetSystemUris** 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.

11.2 HTTP System Restore Test Cases

11.2.1 Feature Level Requirement:

Validated Feature: System Restore via HTTP (HTTPSystemRestore)

Check Condition based on Device Features: HTTP System Backup is supported by Device.

Required Number of Devices: 1

11.2.2 Expected Scenarios Under Test:

1. Client connects to the Device to retrieve URI to which the backed up data may be uploaded using the StartSystemRestore operation.

Client uploads the backed up configuration data using HTTP POST to the Upload Uri provided by the Device in StartSystemRestoreResponse.

2. Client is considered as supporting HTTP System Restore if the following conditions are met:
 - Client is able to retrieve URI from Device for restore system configurations using **StartSystemRestore** operation if Device supports HTTP System Backup AND
 - Client is able to send the backed up data to the Device using **HTTP POST** if Device supports HTTP System Backup.
3. Client is considered as NOT supporting HTTP System Restore if ANY of the following is TRUE:
 - No valid responses for **StartSystemRestore** request if Device supports HTTP System Backup OR
 - No valid **HTTP POST** request to the Upload Uri if Device supports HTTP System Backup.
 - No valid responses for **HTTP POST** request to the Upload Uri if Device supports HTTP System Backup.

11.2.3 HTTP SYSTEM RESTORE

Test Label: System Restore via HTTP - Start System Restore

Test Case ID: HTTPSYSTEMRESTORE-1

Feature Under Test: Start System Restore (HTTPSystemRestore_StartSystemRestore)

Test Purpose: To verify that Client is able to restore system configurations via HTTP using the **StartSystemRestore** operation and **HTTP POST**.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **StartSystemRestore** operation present.
- Device supports HTTP System Backup.

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

1. Client invokes **StartSystemRestore** request message to retrieve upload URI from the Device.
2. Device responds with code HTTP 200 OK and **StartSystemRestoreResponse** message.
3. Client transmits the configuration data to the upload URI using **HTTP POST**.
4. Device responds with code HTTP 200 OK message.

Test Result:

PASS -

- Client **StartSystemRestore** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **StartSystemRestore** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tds:StartSystemRestore** AND
- Device response on the **StartSystemRestore** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **tds:StartSystemRestoreResponse**.
- There is **HTTP POST** request in Test Procedure that fulfills the following requirements:
 - [S4] It invoked to address which equal to **tds:StartSystemRestore/tds:UploadUri** value from the Device response to **StartSystemRestore** request AND
 - [S5] It invoked after the Client **StartSystemRestore** request AND

- [S6] It contains HTTP Content-Type Header with value is equal to “application/octet-stream”
AND
- Device response on the **HTTP POST** request fulfills the following requirements:
 - [S7] It has HTTP 200 response code.

FAIL -

- The Client failed PASS criteria.

12 Test Cases for Standard Events for Monitoring

12.1 Monitoring Notifications Test Cases

12.1.1 Feature Level Requirement:

Validated Feature: Monitoring Notifications (MonitoringNotifications)

Check Condition based on Device Features: Monitoring/ProcessorUsage or Monitoring/OperatingTime/LastReset or Monitoring/OperatingTime/LastReboot or Monitoring/OperatingTime/LastClockSynchronization is supported by Device.

Required Number of Devices: 1

12.1.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get monitoring notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Monitoring Notifications if the following conditions are met:
 - Client supports EventHandling_Pullpoint feature (please see [EVENTHANDLING-1 PULLPOINT](#) section) AND
 - Client is able to retrieve at least one of the following notifications:
 - tns1:Monitoring/ProcessorUsage notification about processor usage if Device supports MonitoringProcessorUsageEvent feature
 - tns1:Monitoring/OperatingTime/LastReset notification about last reset if Device supports MonitoringOperatingTimeLastResetEvent feature
 - tns1:Monitoring/OperatingTime/LastReboot notification about last reboot if Device supports MonitoringOperatingTimeLastRebootEvent feature
 - tns1:Monitoring/OperatingTime/LastClockSynchronization notification about last clock synchronization if Device supports MonitoringOperatingTimeLastClockSynchronizationEvent feature
4. Client is considered as NOT supporting Monitoring Notifications 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 the following notifications:
 - tns1:Monitoring/ProcessorUsage notification about processor usage if Device supports MonitoringProcessorUsageEvent feature
 - tns1:Monitoring/OperatingTime/LastReset notification about last reset if Device supports MonitoringOperatingTimeLastResetEvent feature
 - tns1:Monitoring/OperatingTime/LastReboot notification about last reboot if Device supports MonitoringOperatingTimeLastRebootEvent feature
 - tns1:Monitoring/OperatingTime/LastClockSynchronization notifications about last clock synchronization if Device supports MonitoringOperatingTimeLastClockSynchronizationEvent feature.

12.1.3 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.

13 Test Cases for Standard Events for Device Management

13.1 Device Management Notifications Test Cases

13.1.1 Feature Level Requirement:

Validated Feature: Device Management Notifications (DeviceManagementNotifications)

Check Condition based on Device Features: Device/HardwareFailure/FanFailure or Device/HardwareFailure/PowerSupplyFailure or Device/HardwareFailure/StorageFailure or Device/HardwareFailure/TemperatureCritical or Monitoring/Backup/Last is supported by Device.

Required Number of Devices: 1

13.1.2 Expected Scenarios Under Test:

1. Client subscribes to device messages using **CreatePullPointSubscription** operation to get device management notifications.
2. Client uses Pull Point event mechanism to retrieve notification events from Device.
3. Client is considered as supporting Device Management Notifications if the following conditions are met:
 - Client supports EventHandling_Pullpoint feature (please see [EVENTHANDLING-1 PULLPOINT](#) section) AND
 - Client is able to retrieve at least one of the following notifications:
 - tns1:Device/HardwareFailure/FanFailure notification about fan failure if Device supports DeviceHardwareFailureFanFailureEvent feature
 - tns1:Device/HardwareFailure/PowerSupplyFailure notification about power supply failure if Device supports DeviceHardwareFailurePowerSupplyFailureEvent feature
 - tns1:Device/HardwareFailure/StorageFailure notification about storage failure if Device supports DeviceHardwareFailureStorageFailureEvent feature
 - tns1:Device/HardwareFailure/TemperatureCritical notification about temperature critical if Device supports DeviceHardwareFailureTemperatureCriticalEvent feature

- tns1:Monitoring/Backup/Last notification about last backup if Device supports MonitoringBackupLastEvent feature
4. Client is considered as NOT supporting Device Management Notifications 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 the following notifications:
 - tns1:Device/HardwareFailure/FanFailure notification about fan failure if Device supports DeviceHardwareFailureFanFailureEvent feature
 - tns1:Device/HardwareFailure/PowerSupplyFailure notification about power supply failure if Device supports DeviceHardwareFailurePowerSupplyFailureEvent feature
 - tns1:Device/HardwareFailure/StorageFailure notification about storage failure if Device supports DeviceHardwareFailureStorageFailureEvent feature
 - tns1:Device/HardwareFailure/TemperatureCritical notification about temperature critical if Device supports DeviceHardwareFailureTemperatureCriticalEvent feature
 - tns1:Monitoring/Backup/Last notification about last backup if Device supports MonitoringBackupLastEvent feature

13.1.3 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.

14 Test Cases for TLS Configuration

14.1 TLS Configuration Test Cases

14.1.1 Feature Level Requirement:

Validated Feature: TLS Configuration (TLSConfiguration)

Check Condition based on Device Features: TLS Server (Security Configuration Service) is supported by Device.

Required Number of Devices: 1

Profile A Requirement: None

Profile C Requirement: None

Profile G Requirement: None

Profile S Requirement: None

14.1.2 Expected Scenarios Under Test:

1. Client connects to Device to manage the associations between certification paths and the TLS server.
2. Client is considered as supporting TLS Configuration if the following conditions are met:
 - Client may upload a passphrase from the keystore of the Device using **UploadPassphrase** operation if Device supports Passphrase handling AND
 - Client may delete a passphrase to the keystore of the Device using **DeletePassphrase** operation if Device supports Passphrase handling AND
 - Client is able to generate a DER-encoded PKCS#10 using **CreatePKCS10CSR** operation and upload created certificate using **UploadCertificate** operation if Device supports PKCS10ExternalCertificationWithRSA AND
 - Client is able to upload a certificate using **UploadCertificate** operation if Device supports PKCS10ExternalCertificationWithRSA AND
 - Client is able to delete a certificate to the keystore of the Device using **DeleteCertificate** operation if Device supports

- PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload AND
- Client is able to delete a certification path using **DeleteCertificationPath** operation if Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload AND
 - Client is able to delete a key using **DeleteKey** operation if MaximumNumberOfKeys is greater than zero on Device AND
 - Client is able to get key status using EITHER **GetKeyStatus** operation OR using **tns1:Advancedsecurity/Keystore/KeyStatus** event if MaximumNumberOfKeys is greater than zero on Device AND
 - Client supports EventHandling_Pullpoint feature (please see [EVENTHANDLING-1 PULLPOINT](#) section) when **tns1:Advancedsecurity/Keystore/KeyStatus** event is supported AND
 - Client is able to upload a certification path consisting of X.509 certificates using **UploadCertificateWithPrivateKeyInPKCS12** operation if Device supports PKCS12CertificateWithRSAPrivateKeyUpload AND
 - Client is able to assigns a key pair and certificate along with a certification path to the TLS server on the Device using **AddServerCertificateAssignment** operation if Device supports TLSServerSupport AND
 - Client is able to remove key pair and certificate assignment to the TLS server on the Device using **RemoveServerCertificateAssignment** operation if Device supports TLSServerSupport AND
 - Client is able to replace an existing key pair and certificate assignment to the TLS server on the Device by a new key pair and certificate assignment using **ReplaceServerCertificateAssignment** operation if Device supports TLSServerSupport AND
 - Client is able to create certification path using **CreateCertificationPath** operation if Device supports TLSServerSupport AND
 - Client is able to generate RSA key pair using **CreateRSAKeyPair** operation if Device supports RSAKeyPairGeneration AND
 - Client supports NetworkConfiguration_SetNetworkInterfaces feature (please see [NETWORKCONFIGURATION-2 SET NETWORK INTERFACES](#) section).
3. Client is considered as NOT supporting TLS Configuration if ANY of the following is TRUE:

- No valid responses for **UploadPassphrase** request if detected if Device supports Passphrase handling OR
- No valid responses for **DeletePassphrase** request if detected if Device supports Passphrase handling OR
- No valid responses for **CreatePKCS10CSR** request if Device supports Passphrase handling OR
- No valid responses for **UploadCertificate** request if Device supports Passphrase handling OR
- No valid responses for **DeleteCertificate** request if Device supports PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **DeleteCertificationPath** request if Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **DeleteKey** request if MaximumNumberOfKeys is greater than zero on Device OR
- No valid responses for **GetKeyStatus** request if detected if MaximumNumberOfKeys is greater than zero on Device OR
- Client unable to get key status using **GetKeyStatus** request OR using **tns1:Advancedsecurity/Keystore/KeyStatus** event if MaximumNumberOfKeys is greater than zero on Device OR
- Client does not support EventHandling_Pullpoint feature (please see [EVENTHANDLING-1 PULLPOINT](#) section) when Client supports **tns1:Advancedsecurity/Keystore/KeyStatus** notification if MaximumNumberOfKeys is greater than zero on Device OR
- No valid responses for **UploadCertificateWithPrivateKeyInPKCS12** request if Device supports PKCS12CertificateWithRSAPrivateKeyUpload OR
- No valid responses for **AddServerCertificateAssignment** request if Device supports TLSServerSupport OR
- No valid responses for **RemoveServerCertificateAssignment** request if Device supports TLSServerSupport OR

- No valid responses for **ReplaceServerCertificateAssignment** request if Device supports TLSServerSupport OR
- No valid responses for **CreateCertificationPath** request if Device supports TLSServerSupport OR
- No valid responses for **CreateRSAKeyPair** request if Device supports RSAKeyPairGeneration OR
- Client does not support NetworkConfiguration_SetNetworkInterfaces feature (please see [NETWORKCONFIGURATION-2 SET NETWORK INTERFACES](#) section).

14.1.3 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.

14.1.4 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.

14.1.5 UPLOAD PASSPHRASE

Test Label: Upload Passphrase

Test Case ID: TLSCONFIGURATION-1

Feature Under Test: Upload Passphrase (TLSConfiguration_UploadPassphrase)

Test Purpose: To verify that Client is able to upload a passphrase to the keystore of the Device using **UploadPassphrase** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadPassphrase** operation present.
- Device supports Security Configuration Service.
- Device supports Passphrase handling.

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

1. Client invokes **UploadPassphrase** request message to upload a passphrase to the Device.
2. Device responds with code HTTP 200 OK and **UploadPassphraseResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

14.1.6 DELETE PASSPHRASE

Test Label: Delete Passphrase

Test Case ID: TLSCONFIGURATION-2

Feature Under Test: Delete Passphrase (TLSConfiguration_DeletePassphrase)

Test Purpose: To verify that Client is able to delete a passphrase from the keystore of the Device using **DeletePassphrase** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeletePassphrase** operation present.
- Device supports Security Configuration Service.
- Device supports Passphrase handling.

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

1. Client invokes **DeletePassphrase** request message to delete a passphrase from the Device.
2. Device responds with code HTTP 200 OK and **DeletePassphraseResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.7 CREATE PKCS#10 CERTIFICATION

Test Label: Create PKCS#10 Certification

Test Case ID: TLSCONFIGURATION-3

Feature	Under	Test:	Create	PKCS#10	Certification
(TLSConfiguration_CreatePKCS10Certification)					

Test Purpose: To verify that Client is able to generates a DER-encoded PKCS#10 using **CreatePKCS10CSR** operation, create an X.509 certificate from a PKCS#10 certification request and upload created certificate using **UploadCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreatePKCS10CSR** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA.

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

1. Client invokes **CreatePKCS10CSR** request message to generate PKCS#10 on the Device.
2. Device responds with code HTTP 200 OK and **CreatePKCS10CSRResponse** message.
3. Client creates a certificate from the PKCS#10 request with RSAkey pair and associated CA certificate and a corresponding private key
4. Client invokes **UploadCertificate** request message to upload a certificate on the Device.

5. Device responds with code HTTP 200 OK and **UploadCertificateResponse** message.

Test Result:**PASS -**

- Client **CreatePKCS10CSR** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **CreatePKCS10CSR** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tas:CreatePKCS10CSR** AND
- Device response on the **CreatePKCS10CSR** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **tas:CreatePKCS10CSRResponse**.
- There is Client **UploadCertificate** request in Test Procedure that fulfills the following requirements:
 - [S4] It is invoked after the Client **CreatePKCS10CSR** request AND
 - **tas:UploadCertificate/tas:Certificate** element value fulfills the following requirements:
 - [S5] It contains Subject element with value equals to Subject element value from **tas:CreatePKCS10CSRResponse/tas:PKCS10CSR** AND
 - [S6] It contains Public Key element with value equals to Public Key element value from **tas:CreatePKCS10CSRResponse/tas:PKCS10CSR** AND
- Device response to the **UploadCertificate** request fulfills the following requirements:
 - [S7] It has RTSP 200 response code AND
 - [S8] **soapenv:Body** element has child element **tas:UploadCertificateResponse**.

FAIL -

- The Client failed PASS criteria.

14.1.8 UPLOAD CERTIFICATE

Test Label: Upload Certificate**Test Case ID:** TLSCONFIGURATION-4**Feature Under Test:** Upload Certificate (TLSConfiguration_UploadCertificate)

Test Purpose: To verify that Client is able to upload a certificate using **UploadCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadCertificate** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA.

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

1. Client invokes **UploadCertificate** request message to upload a certificate on the Device.
2. Device responds with code HTTP 200 OK and **UploadCertificateResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

14.1.9 DELETE CERTIFICATE

Test Label: Delete Certificate

Test Case ID: TLSCONFIGURATION-5

Feature Under Test: Delete Certificate (TLSConfiguration_DeleteCertificate)

Test Purpose: To verify that Client is able to delete a certificate using **DeleteCertificate** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **DeleteCertificate** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS10ExternalCertificationWithRSA or SelfSignedCertificateCreationWithRSA or PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **DeleteCertificate** request message to delete a certificate from the Device.
2. Device responds with code HTTP 200 OK and **DeleteCertificateResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.10 DELETE CERTIFICATION PATH

Test Label: Delete Certification Path**Test Case ID:** TLSCONFIGURATION-6**Feature Under Test:** Delete Certification Path (TLSConfiguration_DeleteCertificationPath)**Test Purpose:** To verify that Client is able to delete a certification path using **DeleteCertificationPath** operation.**Pre-Requisite:**

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

- Device supports Security Configuration Service.
- Device supports TLSServerSupport or PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **DeleteCertificationPath** request message to delete a certification path from the Device.
2. Device responds with code HTTP 200 OK and **DeleteCertificationPathResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.11 DELETE KEY

Test Label: DeleteKey**Test Case ID:** TLSCONFIGURATION-7**Feature Under Test:** Delete Key (TLSConfiguration_DeleteKey)**Test Purpose:** To verify that Client is able to delete a key using **DeleteKey** operation.**Pre-Requisite:**

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

- MaximumNumberOfKeys is greater than zero.

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

1. Client invokes **DeleteKey** request message to delete a key from the keystore of Device.
2. Device responds with code HTTP 200 OK and **DeleteKeyResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

14.1.12 GET KEY STATUS

Test Label: Get Key Status

Test Case ID: TLSCONFIGURATION-8

Feature Under Test: Get Key Status (TLSConfiguration_GetKeyStatus)

Test Purpose: To verify that Client is able to get key status using **GetKeyStatus** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **GetKeyStatus** operation present.
- Device supports Security Configuration Service.
- MaximumNumberOfKeys is greater than zero.

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

1. Client invokes **GetKeyStatus** request message to get a key status from the Device.
2. Device responds with code HTTP 200 OK and **GetKeyStatusResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.13 UPLOAD PKCS12

Test Label: Upload PKCS12**Test Case ID:** TLSCONFIGURATION-9**Feature Under Test:** Upload PKCS12 (TLSConfiguration_UploadPKCS12)**Test Purpose:** To verify that Client is able to upload a certification path consisting of X.509 certificates using **UploadCertificateWithPrivateKeyInPKCS12** operation.**Pre-Requisite:**

- The Network Trace Capture files contains at least one Conversation between Client and Device with **UploadCertificateWithPrivateKeyInPKCS12** operation present.
- Device supports Security Configuration Service.
- Device supports PKCS12CertificateWithRSAPrivateKeyUpload.

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

1. Client invokes **UploadCertificateWithPrivateKeyInPKCS12** request message to upload a PKCS12 to the Device.

2. Device responds with code HTTP 200 OK and **UploadCertificateWithPrivateKeyInPKCS12Response** message.

Test Result:**PASS -**

- Client **UploadCertificateWithPrivateKeyInPKCS12** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **UploadCertificateWithPrivateKeyInPKCS12** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tas:UploadCertificateWithPrivateKeyInPKCS12** AND
- Device response on the **UploadCertificateWithPrivateKeyInPKCS12** request fulfills the following requirements:
 - [S2] It has HTTP 200 response code AND
 - [S3] **soapenv:Body** element has child element **tas:UploadCertificateWithPrivateKeyInPKCS12Response**.

FAIL -

- The Client failed PASS criteria.

14.1.14 ADD SERVER CERTIFICATE ASSIGNMENT

Test Label: Add Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-10

Feature Under Test: Add Server Certificate Assignment
(TLSConfiguration_AddServerCertificateAssignment)

Test Purpose: To verify that Client is able to assigns a key pair and certificate along with a certification path to the TLS server on the Device using **AddServerCertificateAssignment** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **AddServerCertificateAssignment** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **AddServerCertificateAssignment** request message to assign of a certificate to a TLS server.
2. Device responds with code HTTP 200 OK and **AddServerCertificateAssignmentResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.15 REMOVE SERVER CERTIFICATE ASSIGNMENT

Test Label: Remove Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-11

Feature Under Test: Remove Server Certificate Assignment
(TLSConfiguration_RemoveServerCertificateAssignment)

Test Purpose: To verify that Client is able to remove key pair and certificate assignment to the TLS server on the Device using **RemoveServerCertificateAssignment** operation.

Pre-Requisite:

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

- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **RemoveServerCertificateAssignment** request message to remove server certification assignment.
2. Device responds with code HTTP 200 OK and **RemoveServerCertificateAssignmentResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

14.1.16 REPLACE SERVER CERTIFICATE ASSIGNMENT

Test Label: Replace Server Certificate Assignment

Test Case ID: TLSCONFIGURATION-12

Feature Under Test: Replace Server Certificate Assignment
(TLSConfiguration_ReplaceServerCertificateAssignment)

Test Purpose: To verify that Client is able to replace an existing key pair and certificate assignment to the TLS server on the Device by a new key pair and certificate assignment using **ReplaceServerCertificateAssignment** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **ReplaceServerCertificateAssignment** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **ReplaceServerCertificateAssignment** request message to replace certificate assignment to a TLS server.
2. Device responds with code HTTP 200 OK and **ReplaceServerCertificateAssignmentResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

14.1.17 CREATE CERTIFICATION PATH

Test Label: Create Certification Path**Test Case ID:** TLSCONFIGURATION-13**Feature Under Test:** Create Certification Path (TLSConfiguration_CreateCertificationPath)

Test Purpose: To verify that Client is able to create certification path using **CreateCertificationPath** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateCertificationPath** operation present.
- Device supports Security Configuration Service.
- Device supports TLSServerSupport.

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

1. Client invokes **CreateCertificationPath** request message to create certification path.
2. Device responds with code HTTP 200 OK and **CreateCertificationPathResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

14.1.18 CREATE RSA KEY PAIR

Test Label: Create RSA Key Pair

Test Case ID: TLSCONFIGURATION-14

Feature Under Test: Create RSA Key Pair (TLSConfiguration_CreateRSAKeyPair)

Test Purpose: To verify that Client is able to generate RSA key pair using **CreateRSAKeyPair** operation.

Pre-Requisite:

- The Network Trace Capture files contains at least one Conversation between Client and Device with **CreateRSAKeyPair** operation present.
- Device supports Security Configuration Service.
- Device supports RSAKeyPairGeneration.

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

1. Client invokes **CreateRSAKeyPair** request message to create RSA key pair.
2. Device responds with code HTTP 200 OK and **CreateRSAKeyPairResponse** message.

Test Result:**PASS -**

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

FAIL -

- The Client failed PASS criteria.

15 Test Cases for Privacy Masks for Media2

15.1 Privacy Masks for Media2 Test Cases

15.1.1 Feature Level Normative Reference:

Validated Feature: Mask Configuration (Media2_Mask)

Check Condition based on Device Features: Mask and Media2 Service are supported by Device.

Required Number of Devices: 1

15.1.2 Expected Scenarios Under Test:

1. Client connects to Device to list Masks, create Mask, remove Mask, and modify Mask on the device.
2. Client is considered as supporting Privacy Masks if the following conditions are met:
 - Client is able to retrieve Privacy Masks using **GetMasks** operation (Media2 Service) AND
 - Client is able to create Privacy Masks using **CreateMask** operation (Media2 Service) AND
 - Client is able to retrieve Mask options using **GetMaskOptions** operation (Media2 Service) AND
 - Client is able to delete Privacy Mask using **DeleteMask** operation (Media2 Service) AND
 - Client is able to modify Mask using **SetMask** operation (Media2 Service).
3. Client is considered as NOT supporting OSD Configuration if ANY of the following is TRUE:
 - No valid response to **GetMasks** request (Media2 Service) OR
 - No valid response to **CreateMask** operation (Media2 Service) OR
 - No valid response to **GetMaskOptions** operation (Media2 Service) OR
 - No valid response to **DeleteMask** operation (Media2 Service) OR
 - No valid response to **SetMask** operation (Media2 Service).

15.1.3 GET MASKS USING MEDIA2

Test Label: Mask - Get Masks

Test Case ID: MEDIA2_MASK-1

Feature Under Test: Get Masks (Media2_Mask_Media2_GetMasks)

Test Purpose: To verify that existing Mask configurations is received by Client using the **GetMasks** operation.

Pre-Requisite:

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

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

1. Client invokes **GetMasks** request message to retrieve Mask configurations from the Device.
2. Device responds with code HTTP 200 OK and **GetMasksResponse** message.

Test Result:

PASS -

- Client **GetMasks** request messages are valid according to XML Schemas listed in [Namespaces](#) AND
- Client **GetMasks** request in Test Procedure fulfills the following requirements:
 - [S1] **soapenv:Body** element has child element **tr2:GetMasks** AND
 - [S2] **tr2:GetMasks** element does not contain child element **tr2:Token** AND
- Device response on the **GetMasks** request fulfills the following requirements:
 - [S3] It has HTTP 200 response code AND
 - [S4] **soapenv:Body** element has child element **tr2:GetMasksResponse**.

FAIL -

- The Client failed PASS criteria.

15.1.4 CREATE MASK USING MEDIA2

Test Label: Mask - Create Mask

Test Case ID: MEDIA2_MASK-2

Feature Under Test: Create Mask (Media2_Mask_Media2_CreateMask)

Test Purpose: To verify that Client is able to create Mask using the **CreateMask** operation

Pre-Requisite:

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

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

1. Client invokes **CreateMask** request message to create Mask on the Device.
2. Device responds with code HTTP 200 OK and **CreateMaskResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

15.1.5 GET MASK OPTIONS USING MEDIA2

Test Label: Mask - Get Mask Options

Test Case ID: MEDIA2_MASK-3

Feature Under Test: Get Mask Options (Media2_Mask_Media2_GetMaskOptions)

Test Purpose: To verify that Mask options provided by Device is received by Client using the **GetMaskOptions** operation.

Pre-Requisite:

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

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

1. Client invokes **GetMaskOptions** request message to retrieve an Mask options from the Device.
2. Device responds with code HTTP 200 OK and **GetMaskOptionsResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

15.1.6 DELETE MASK USING MEDIA2

Test Label: Mask - Delete Mask

Test Case ID: MEDIA2_MASK-4

Feature Under Test: Delete Mask (Media2_Mask_Media2_DeleteMask)

Test Purpose: To verify that Client is able to delete Mask using the **DeleteMask** operation

Pre-Requirement:

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

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

1. Client invokes **DeleteMask** request message to delete Mask configuration from the Device.
2. Device responds with code HTTP 200 OK and **DeleteMaskResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

15.1.7 SET MASK USING MEDIA2

Test Label: Mask - Set Mask

Test Case ID: MEDIA2_MASK-5

Feature Under Test: Set Mask (Media2_Mask_Media2_SetMask)

Test Purpose: To verify that Client is able to change Mask provided by Device using the **SetMask** operation.

Pre-Requisite:

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

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

1. Client invokes **SetMask** request message to change a Mask on the Device.
2. Device responds with code HTTP 200 OK and **SetMaskResponse** message.

Test Result:

PASS -

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

FAIL -

- The Client failed PASS criteria.

Annex A Test for Appendix A

A.1 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.AudioBackchannelStreaming	Audio Backchannel Streaming	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetAudioDecoderConfigurationsList	Get Audio Decoder Configurations List	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetAudioOutputConfigurationsList	Get Audio Output Configurations List	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetAudioOutputsList	Get Audio Outputs List	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetAudioDecoderConfiguration	Get Audio Decoder Configuration	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetAudioOutputConfiguration	Get Audio Output Configuration	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.ProfileConfigurationForAudioBackchannel	Profile Configuration for Audio Backchannel	1	Audio Output (Media Service) is supported by Device.	AudioOutput

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
tc.SetAudioDecoderConfiguration	Configure Audio Decoder Configuration	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.SetAudioOutputConfiguration	Configure Audio Output Configuration	1	Audio Output (Media Service) is supported by Device.	AudioOutput
tc.GetImagingCapabilities	Get Imaging Capabilities	1	Imaging Service is supported by Device.	ImagingService
tc.GetOSD	Get OSD Configuration	1	TO BE DISCUSSED	TBD
tc.GetOSDs	Get OSD List	1	TO BE DISCUSSED	TBD
tc.SetOSD	OSD Configuration	1	TO BE DISCUSSED	TBD
tc.EnabledTLSVersionsConfiguration	Enabled TLS Versions Configuration	1	Enabled TLS Versions (Security Configuration Service) is supported by the Device.	EnabledTLSVersions
tc.TransitionToOperationalState	Transition to Operational State	3	TO BE DISCUSSED	TBD
tc.HTTPFirmwareUpgrade	HTTP Firmware Upgrade	1	HTTP Firmware Upgrade is supported by Device.	HttpFirmwareUpgrade
tc.HTTPSystemBackup	HTTP System Backup	1	HTTP System Backup is supported by Device.	HttpSystemBackup
tc.HTTPSystemRestore	HTTP System Restore	1	HTTP System Backup is	HttpSystemBackup

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			supported by Device.	
tc.MonitoringNotifications	Monitoring Notifications	1	Monitoring/ProcessorUsage or Monitoring/OperatingTime/LastReset or Monitoring/OperatingTime/LastReboot or Monitoring/OperatingTime/LastClockSynchronization is supported by Device.	MonitoringProcessorUsageEvent OR MonitoringOperatingTimeLastResetEvent OR MonitoringOperatingTimeLastRebootEvent OR MonitoringOperatingTimeLastClockSynchronizationEvent
tc.DeviceManagementNotifications	Device Management Notifications	1	Check Condition based on Device Features: Device/HardwareFailure/FanFailure or Device/HardwareFailure/PowerSupplyFailure or Device/HardwareFailure/StorageFailure or Device/HardwareFailure/TemperatureCritical or Monitoring/Backup/Last is supported by Device.	MonitoringBackupLastEvent OR DeviceHardwareFailureFanFailureEvent OR DeviceHardwareFailurePowerSupplyFailureEvent OR DeviceHardwareFailureStorageFailureEvent OR DeviceHardwareFailureTemperatureCriticalEvent
tc.TLSConfiguration	TLS Configuration	1	TLS Server (Security Configuration Service) is	TLSServerSupport

Feature ID	Feature Name	Required Number of Devices	Check Condition based on Device Features	Check Condition based on Device Features ID
			supported by Device.	
tc.Media2_Mask	Privacy Masks for Media2	1	Mask and Media2 Service are supported by Device.	Media2Service AND Mask