ONVIF®

Recording Control Device Test Specification

Version 26.06

June 2026

REVISION HISTORY

Vers.	Date	Description
12.12	Dec 20, 2012	First issue of Replay Control Test Specification
13.06	Jun, 2013	The following test cases were updated or added: RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE) RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER) SET RECORDINGS CONFIGURATION (MAXIMUM LENGTH OF RECORDING SOURCE INFORMATION) DYNAMIC RECORDINGS CONFIGURATION (MAXIMUM LENGTH OF RECORDING SOURCE INFORMATION) RECORDING CONTROL – JOB STATE EVENT RECORDING CONTROL – JOB STATE CHANGE EVENT RECORDING CONTROL – RECORDING CONFIGURATION EVENT RECORDING CONTROL – TRACK CONFIGURATION EVENT RECORDING CONTROL – RECORDING JOB CONFIGURATION EVENT RECORDING CONTROL – CREATE RECORDING EVENT RECORDING CONTROL – DELETE RECORDING EVENT RECORDING CONTROL – CREATE TRACK EVENT RECORDING CONTROL – DELETE TRACK EVENT RECORDING CONTROL – CREATE TRACK EVENT (CREATE RECORDING) RECORDING CONTROL – DELETE TRACK EVENT (DELETE RECORDING) RECORDING CONTROL – CREATE TRACK EVENT (CREATE TRACK) RECORDING CONTROL – DELETE TRACK EVENT (DELETE TRACK) RECORDING CONTROL – DELETE TRACK DATA EVENT
13.12	Dec, 2013	The following test cases were updated or added: RECORDING-2-1-1 START RECORDING USING A MEDIA PROFILE was replaced with RECORDING-2-1-10 START RECORDING ON MEDIA PROFILE RECORDING-2-1-2 START RECORDING ON RECEIVER was replaced with RECORDING-2-1-11 START RECORDING ON RECEIVER

		RECORDING-2-1-3 STOP RECORDING USING A MEDIA PROFILE - PUT JOB IN IDLE STATE was replaced with RECORDING-2-1-12 STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE RECORDING-2-1-4 STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE was replaced with RECORDING-2-1-13 STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE RECORDING-2-1-5 STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE was replaced with RECORDING-2-1-14 STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE RECORDING-2-1-6 STOP RECORDING USING A MEDIA PROFILE - DELETE JOB was replaced with RECORDING-2-1-15 STOP RECORDING ON MEDIA PROFILE - DELETE JOB RECORDING-2-1-7 STOP RECORDING ON RECEIVER - DELETE JOB was replaced with RECORDING-2-1-16 STOP RECORDING ON RECEIVER - DELETE JOB RECORDING-2-1-8 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - MEDIA PROFILE was replaced with RECORDING-2-1-17 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - MEDIA PROFILE RECORDING-2-1-9 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - RECEIVER was replaced with RECORDING-2-1-18 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - RECEIVER RECORDING-3-1-2 DYNAMIC TRACKS CONFIGURATION was replaced with RECORDING-3-1-7 DYNAMIC TRACKS CONFIGURATION RECORDING-3-1-5 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE) was replaced with RECORDING-3-1-8 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE) RECORDING-3-1-6 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER) was replaced with RECORDING-3-1-9 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER) RECORDING-4-1-6 GET RECORDING JOB CONFIGURATION WITH INVALID TOKEN was replaced with RECORDING-4-1-13 GET RECORDING JOB CONFIGURATION WITH INVALID TOKEN RECORDING-4-1-8 GET RECORDING JOB STATE WITH INVALID TOKEN was replaced with RECORDING-4-1-14 GET RECORDING JOB STATE WITH INVALID TOKEN RECORDING-5-1-1 RECORDING CONTROL – JOB STATE EVENT was replaced with RECORDING-5-1-18 RECORDING CONTROL – JOB STATE EVENT RECORDING-5-1-2 RECORDING CONTROL – JOB STATE CHANGE EVENT was replaced with RECORDING-5-1-19 RECORDING CONTROL – JOB STATE CHANGE EVENT RECORDING-5-1-5 RECORDING CONTROL – RECORDING JOB CONFIGURATION EVENT was replaced with RECORDING-5-1-20 RECORDING CONTROL – RECORDING JOB CONFIGURATION EVENT

		RECORDING-5-1-7 RECORDING CONTROL – DELETE RECORDING EVENT was replaced with RECORDING-5-1-17 RECORDING CONTROL – DELETE RECORDING EVENT RECORDING-5-1-12 RECORDING CONTROL – CREATE TRACK EVENT (CREATE TRACK) was replaced with RECORDING-5-1-15 RECORDING CONTROL – CREATE TRACK EVENT (CREATE TRACK) RECORDING-5-1-13 RECORDING CONTROL – DELETE TRACK EVENT (DELETE TRACK) was replaced with RECORDING-5-1-16 RECORDING CONTROL – DELETE TRACK EVENT (DELETE TRACK) The following annexes were updated or added: Annex A.6 was replaced with A.12 Selection or Creation of Recording for recording job creation Annex A.9 was replaced with A.13 Auto Creation of Receiver Annex A.14 Selection of Recording for track creation was added Annex A.15 Selection or Creation of Recording for recording job creation on a Media profile was added Annex A.5 was removed.
14.06	Feb, 2014	The following tests were updated with increasing of tests IDs: START RECORDING ON MEDIA PROFILE START RECORDING ON RECEIVER STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE STOP RECORDING ON MEDIA PROFILE - DELETE JOB STOP RECORDING ON RECEIVER - DELETE JOB MODIFY MEDIA ATTRIBUTE WHILE RECORDING - MEDIA PROFILE MODIFY MEDIA ATTRIBUTE WHILE RECORDING – RECEIVER DYNAMIC RECORDINGS CONFIGURATION RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE) RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER) GET SERVICES AND GET RECORDING CONTROL SERVICE CAPABILITIES CONSISTENCY
14.12	Dec, 2014	The following annexes were updated: A.16 **PullMessages** algorithm for check Recording Job State initializing

		The following tests were updated with increasing of tests IDs: START RECORDING ON MEDIA PROFILE STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE STOP RECORDING ON MEDIA PROFILE - DELETE JOB MODIFY MEDIA ATTRIBUTE WHILE RECORDING - MEDIA PROFILE
18.06	Mar, 2018	The following test was removed according #1552: RECORDING-2-1-31 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - MEDIA PROFILE
18.06	June, 2018	Reformatting document using new template
20.06	Dec 18, 2019	The following were updated in the scope of #1435: Reformatting document using new template and format.
20.06	May 13, 2020	Pre-Requisite of the following test cases updated with adding of Pull-Point Notification feature according to #1999: RECORDING-2-1-20 START RECORDING ON RECEIVER RECORDING-2-1-22 STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE RECORDING-2-1-23 STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE RECORDING-2-1-25 STOP RECORDING ON RECEIVER - DELETE JOB RECORDING-2-1-27 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - RECEIVER RECORDING-2-1-28 START RECORDING ON MEDIA PROFILE RECORDING-2-1-29 STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE RECORDING-2-1-30 STOP RECORDING ON MEDIA PROFILE - DELETE JOB RECORDING-5-1-3 RECORDING CONTROL – RECORDING CONFIGURATION EVENT RECORDING-5-1-4 RECORDING CONTROL – TRACK CONFIGURATION EVENT RECORDING-5-1-6 RECORDING CONTROL – CREATE RECORDING EVENT RECORDING-5-1-10 RECORDING CONTROL – CREATE TRACK EVENT (CREATE RECORDING) RECORDING-5-1-11 RECORDING CONTROL – DELETE TRACK EVENT (DELETE RECORDING) RECORDING-5-1-14 RECORDING CONTROL – DELETE TRACK DATA EVENT

		RECORDING-5-1-15 RECORDING CONTROL – CREATE TRACK EVENT (CREATE TRACK) RECORDING-5-1-16 RECORDING CONTROL – DELETE TRACK EVENT (DELETE TRACK) RECORDING-5-1-17 RECORDING CONTROL – DELETE RECORDING EVENT RECORDING-5-1-18 RECORDING CONTROL – JOB STATE EVENT RECORDING-5-1-19 RECORDING CONTROL – JOB STATE CHANGE EVENT RECORDING-5-1-20 RECORDING CONTROL – RECORDING JOB CONFIGURATION EVENT
20.06	May 18, 2020	Pre-Requisite of the following test cases updated with adding of Event Service according to #1999: RECORDING-5-1-8 RECORDING CONTROL – CREATE TRACK EVENT RECORDING-5-1-9 RECORDING CONTROL – DELETE TRACK EVENT
21.06	Apr 13, 2021	The following test cases were updated according to #2147: RECORDING-5-1-18 RECORDING CONTROL – JOB STATE EVENT (step 17, step 26) RECORDING-5-1-19 RECORDING CONTROL – JOB STATE CHANGE EVENT (step 23, step 28)
26.06	Jan 19, 2026	The following test cases and annexes were added/updated/deleted according to #365: RECORDING-1-1-4 GET SERVICES AND GET RECORDING CONTROL SERVICE CAPABILITIES CONSISTENCY (new) RECORDING-4-1-15 RECORDING CONFIGURATION (Static Recording) (new) RECORDING-4-1-16 RECORDING CONFIGURATION (Dynamic Recording) (new) RECORDING-4-1-17 TRACK CONFIGURATION (Static Track) (new) RECORDING-4-1-18 TRACK CONFIGURATION (Dynamic Track) (new) RECORDING-5-1-21 RECORDING CONTROL - RECORDING CONFIGURATION EVENT PROPERTIES (new) RECORDING-5-1-22 RECORDING CONTROL - CREATE RECORDING EVENT PROPERTIES (new) RECORDING-5-1-23 RECORDING CONTROL - DELETE RECORDING EVENT PROPERTIES (new) RECORDING-5-1-24 RECORDING CONTROL - TRACK CONFIGURATION EVENT PROPERTIES (new) HelperPullRecordingConfiguration (new)

		HelperPullCreateRecording (new) HelperPullDeleteRecording (new) HelperComparisonOfRecordings (updated with new fields) HelperGetServiceCapabilities_Recording (new) HelperStorageCreationForRecordingConfigurationTests (new) HelperCreateCertPathValidationPolicyForExternalStorage (new) HelperPullTrackConfiguration (new) HelperComparisonOfTracks (new) HelperRecordingCreation (new) HelperTrackCreationPrerequisite (new) HelperPullDeleteTrack (new) HelperPullCreateTrack (new) HelperTrackCreationPrerequisite (updated with link to new annex for recording creation) RECORDING-1-1-1 RECORDING CONTROL SERVICE CAPABILITIES (deleted) RECORDING-1-1-3 GET SERVICES AND GET RECORDING CONTROL SERVICE CAPABILITIES CONSISTENCY (deleted) RECORDING-3-1-7 DYNAMIC TRACKS CONFIGURATION (deleted) RECORDING-3-1-10 DYNAMIC RECORDINGS CONFIGURATION (deleted) RECORDING-4-1-1 GET RECORDINGS (deleted) RECORDING-4-1-2 GET RECORDING CONFIGURATION (deleted) RECORDING-4-1-9 GET TRACK CONFIGURATION (deleted) RECORDING-4-1-11 SET RECORDINGS CONFIGURATION (MAXIMUM LENGTH OF RECORDING SOURCE INFORMATION) (deleted) RECORDING-4-1-12 DYNAMIC RECORDINGS CONFIGURATION (MAXIMUM LENGTH OF RECORDING SOURCE INFORMATION) (deleted) RECORDING-5-1-3 RECORDING CONTROL – RECORDING CONFIGURATION EVENT (deleted) RECORDING-5-1-4 RECORDING CONTROL – TRACK CONFIGURATION EVENT (deleted) RECORDING-5-1-6 RECORDING CONTROL – CREATE RECORDING EVENT (deleted) RECORDING-5-1-10 RECORDING CONTROL – CREATE TRACK EVENT (CREATE RECORDING) (deleted)

		RECORDING-5-1-11 RECORDING CONTROL – DELETE TRACK EVENT (DELETE RECORDING) (deleted) RECORDING-5-1-15 RECORDING CONTROL – CREATE TRACK EVENT (CREATE TRACK) (deleted) RECORDING-5-1-16 RECORDING CONTROL – DELETE TRACK EVENT (DELETE TRACK) (deleted) RECORDING-5-1-17 RECORDING CONTROL – DELETE RECORDING EVENT (deleted)
26.06	Feb 03, 2026	The following test cases and annexes were added/updated/deleted according to #367: RECORDING-4-1-19 RECORDING JOB CONFIGURATION (new) RECORDING-5-1-25 RECORDING CONTROL - RECORDING JOB CONFIGURATION EVENT PROPERTIES (new) RECORDING-2-2-1 OBJECT STORAGE RECORDING - START AND STOP RECORDING (new) RECORDING-2-2-2 OBJECT STORAGE RECORDING - START AND STOP RECORDING, JOB PRIORITIES (new) RECORDING-5-1-26 RECORDING CONTROL - RECORDING JOB STATE EVENT PROPERTIES (new) RECORDING-2-2-3 OBJECT STORAGE RECORDING - STORAGE IS NOT ACCESSIBLE (new) HelperMedia2ExternalStorageRecordingWithStorageErrorPrerequisite (new) HelperPullRecordingJobConfiguration (new) HelperMedia2ExternalStorageRecordingWithPrioritiesPrerequisite (new) HelperComparisonOfRecordingJobs (new) HelperDeleteAllRecordingJobs (new) HelperRecordingJobCreationPrerequisite (new) HelperMedia2ExternalStorageRecordingPrerequisite (new) HelperStorageCreationForRecordingTests (new) RECORDING-5-1-20 RECORDING CONTROL – RECORDING JOB CONFIGURATION EVENT (removed) RECORDING-4-1-4 GET RECORDING JOBS (removed) RECORDING-4-1-5 GET RECORDING JOB CONFIGURATION (removed) RECORDING-4-1-13 GET RECORDING JOB CONFIGURATION WITH INVALID TOKEN (removed) Configuration section was removed. RECORDING-3-1-12 and RECORDING-3-1-11 were moved to other section.

		RECORDING-2-1-20 START RECORDING ON RECEIVER (onboard storage pre-requisite was added) RECORDING-2-1-22 STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE (onboard storage pre-requisite was added) RECORDING-2-1-23 STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE (onboard storage pre-requisite was added) RECORDING-2-1-25 STOP RECORDING ON RECEIVER - DELETE JOB (onboard storage pre-requisite was added) RECORDING-2-1-27 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - RECEIVER (onboard storage pre-requisite was added) RECORDING-3-1-12 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER) (onboard storage pre-requisite was added) RECORDING-2-1-28 START RECORDING ON MEDIA PROFILE (onboard storage pre-requisite was added) RECORDING-2-1-29 STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE (onboard storage pre-requisite was added) RECORDING-3-1-11 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE) (onboard storage pre-requisite was added) RECORDING-5-1-18 RECORDING CONTROL – JOB STATE EVENT (onboard storage pre-requisite was added, topic set check was removed, ID was updated to RECORDING-5-1-27) RECORDING-5-1-19 RECORDING CONTROL – JOB STATE CHANGE EVENT (onboard storage pre-requisite was added) RECORDING-4-1-14 GET RECORDING JOB STATE WITH INVALID TOKEN (3-10 steps were removed as not sufficient for test target)
26.06	Feb 18, 2026	The following test case/annexes were updated/added according #463: HelperStorageCreationForRecordingConfigurationTests (key pair algorithm selection logic was updated to use HelperKeyAlgorithmSelection) HelperKeyAlgorithmSelection (new)
26.06	Mar 19, 2026	The following test case/annexes were updated/added according #475: RECORDING-2-2-4 OBJECT STORAGE RECORDING - VIDEO RECORDING (H.264, MP4) (new) RECORDING-2-2-5 OBJECT STORAGE RECORDING - VIDEO RECORDING (H.264, CMAF) (new) RECORDING-2-2-6 OBJECT STORAGE RECORDING - AUDIO RECORDING (OPUS, MP4) (new) RECORDING-2-2-7 OBJECT STORAGE RECORDING - AUDIO RECORDING (AAC, MP4) (new)

		RECORDING-2-2-8 OBJECT STORAGE RECORDING - AUDIO RECORDING (OPUS, CMAF) (new) RECORDING-2-2-9 OBJECT STORAGE RECORDING - AUDIO RECORDING (AAC, CMAF) (new) RECORDING-2-2-10 OBJECT STORAGE RECORDING - VIDEO AND AUDIO RECORDING (H.264 + OPUS/AAC, MP4) (new) RECORDING-2-2-11 OBJECT STORAGE RECORDING - VIDEO AND AUDIO RECORDING (H.264 + OPUS/AAC, CMAF) (new) RECORDING-2-2-12 OBJECT STORAGE RECORDING - DIFFERENT STORAGES (ObjectStorageS3) (new) RECORDING-2-2-13 OBJECT STORAGE RECORDING - DIFFERENT STORAGES (ObjectStorageAzure) (new) HelperVideoRecordingExternalStoragePrerequisite (new) HelperAudioRecordingExternalStoragePrerequisite (new) HelperVideoAndAudioRecordingExternalStoragePrerequisite (new) HelperConfigureMediaProfileForVideoAndAudioStreaming (new) HelperMedia2ExternalStorageRecordingPrerequisite (updated to have possibility to set predefined storage type) Editorial changes that not affect tests implementation
26.06	Mar 19, 2026	The following test case/annexes were updated/added according #477: RECORDING-2-2-14 OBJECT STORAGE RECORDING - SYMMETRIC ENCRYPTION (CENC) (new) RECORDING-2-2-15 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (RSA) (new) RECORDING-2-2-16 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (ECC) (new) RECORDING-2-2-17 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (key rotation) (new) RECORDING-2-2-18 OBJECT STORAGE RECORDING - SYMMETRIC ENCRYPTION (CENC, different track encryption) (new) RECORDING-2-2-19 OBJECT STORAGE RECORDING - ENCRYPTION (ambiguous configuration) (new) HelperMedia2ExternalStorageRecordingPrerequisite (new optional encryption parameter was added) HelperVideoAndAudioRecordingExternalStoragePrerequisite (new optional encryption parameter was added, new output recording token parameter was added) HelperCENCSegmentCheck (new) HelperCreateCertificateSignedByCA (new) HelperAsymmetricSegmentCheck (new)

		Editorial changes that not affect tests implementation
26.06	May 04, 2026	The following test case was updated according #479: RECORDING-2-2-15 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (RSA) (test purpose was updated) RECORDING-2-2-16 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (ECC) (test purpose was updated) HelperAsymmetricSegmentCheck (Baseline requirements checks were added for version 1 and 2)
26.06	Mar 19, 2026	The following test case/annexes were updated/added according #481: RECORDING-2-2-20 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (new) RECORDING-2-2-21 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (override current) (new) RECORDING-2-2-22 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (recording configuration update) (new) RECORDING-2-2-23 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (invalid expiration) (new) HelperMedia2ExternalStorageRecordingPrerequisite (new optional parameter recording segment duration was added)
26.06	May 18, 2026	The following test case was updated according #482: RECORDING-2-2-24 OBJECT STORAGE RECORDING - RENEWAL (new) HelperMedia2ExternalStorageRecordingPrerequisite (updated with new input parameter for storage configuration optional parameter) HelperStorageCreationForRecordingTests (updated with new input parameter for storage configuration optional parameter)

Table of Contents

1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementation from different vendors. The set of ONVIF test specification describes the test cases need to verify the [ONVIF DeviceIO Service Specs] and [ONVIF Conformance] requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Recording Control Test Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases need to be executed and passed. Also, this specification acts as an input document to the development of test tool which will be used to test the ONVIF device implementation conformance towards ONVIF standard. This test tool is referred as ONVIF Client hereafter.

1.1 Scope

This ONVIF Recording Control Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be functional black-box testing. The objective of this specification is to provide the test cases to test individual requirements of ONVIF devices according to ONVIF core services which are defined in [ONVIF Network Interface Specs].

The principal intended purposes are:

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

This specification does not address the following:

1. Product use cases and non-functional (performance and regression) testing.
2. SOAP Implementation Interoperability test i.e. Web Service Interoperability Basic Profile version 2.0 (WS-I BP 2.0).
3. Network protocol implementation Conformance test for HTTP, HTTPS, RTP protocol.
4. Wi-Fi Conformance test

The set of ONVIF Test Specification will not cover the complete set of requirements as defined in [ONVIF Network Interface Specs]; instead it would cover subset of it. The scope of this specification is to derive all the normative requirements of [ONVIF DeviceIO Service Specs] which are related to ONVIF Device IO Service and some of the optional requirements.

This ONVIF Recording Control Test Specification covers ONVIF Recording Control service which is a functional block of [ONVIF Network Interface Specs]. The following sections describe the brief overview and scope of each functional block.

1.1.1 Capabilities

Capabilities test cases are covered for verification to get Recording Control Service capabilities. It means that GetServices and GetServiceCapabilities commands are covered by this test case.

1.1.2 Recording

Recording covers the test cases needed for the verification of recording process as mentioned in [ONVIF Network Interface Specs]. Recording section defines different recording control tests for starting and stopping recording.

2 Normative references

- [ONVIF Conformance] ONVIF Conformance Process Specification:

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

- [ONVIF Profile Policy] ONVIF Profile Policy:

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

- [ONVIF Network Interface Specs] ONVIF Network Interface Specification documents:

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

- [ONVIF Core Specs] ONVIF Core Specifications:

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

- [ONVIF Recording Service Specs] ONVIF Recording Control Specifications:

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

- [ONVIF Base Test] ONVIF Base Device Test Specification:

  https://www.onvif.org/profiles/conformance/device-test/

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

  http://www.iso.org/directives

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

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

- [SOAP 1.2, Part 1] W3C SOAP 1.2, Part 1, Messaging Framework:

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

- [XML-Schema, Part 1] W3C XML Schema Part 1: Structures Second Edition:

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

- [XML-Schema, Part 2] W3C XML Schema Part 2: Datatypes Second Edition:

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

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

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

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 defines terms that are specific to the ONVIF Receiver Service and tests. For the list of applicable general terms and definitions, please see [ONVIF Base Test].

Metadata All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications).

Recording A container for a set of audio, video and metadata tracks. A recording can hold one or more tracks. A track is viewed as an infinite timeline that holds data at certain times.

Recording Event An event associated with a Recording, represented by a notification message in the APIs

Recording Job A job performs the transfer of data from a data source to a particular recording using a particular configuration

Track An individual data channel consisting of video, audio, or metadata. This definition is consistent with the definition of track in [RFC 2326]

Video Analytics Algorithms or programs used to analyze video data and to generate data describing object location and behavior.

3.3 Abbreviations

This section describes abbreviations used in this document.

DUT Device Under Test

HTTP Hyper Text Transport Protocol

RTCP RTP Control Protocol

RTSP Real Time Streaming Protocol

RTP Real-time Transport Protocol

SDP Session Description Protocol

TCP Transport Control Protocol

UTC Coordinated Universal Time

UDP User Datagram Protocol

URI	Uniform Resource Identifier
WSDL	Web Services Description Language
WS-I BP 2.0	Web Services Interoperability Basic Profile version 2.0
XML	eXtensible Markup Language

4 Test Overview

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

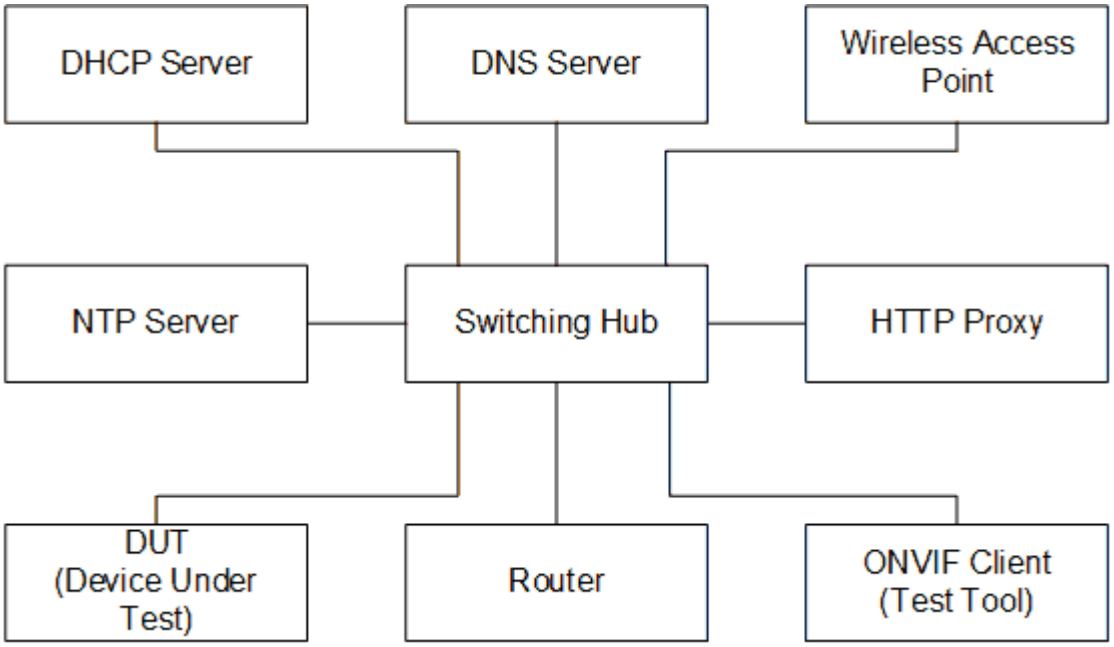
4.1 Test Setup

4.1.1 Network Configuration for DUT

The generic test configuration for the execution of test cases defined in this document is as shown below (Figure 4.1).

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

Figure 4.1. Test Configuration for DUT



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

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

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

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

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

DHCP Server: provides IPv4 Address to the connected devices.

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

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

Router: provides router advertisements for IPv6 configuration.

4.2 Prerequisites

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

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

4.3 Test Policy

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

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

4.3.1 Capabilities

The device under test shall demonstrate recording control service capability in GetServices and GetServiceCapabilities responses. A DUT that does not display recording control service capability constitutes failure of test procedure.

Please refer to Section 5.1 for Capabilities Test Cases.

4.3.2 Recording

The DUT shall give the Recording Control Service entry point by GetServices command.

Please refer to Section 5.2 for Recording Test Cases.

4.3.3 General

The DUT shall give the Recording Control Service entry point by GetServices command.

Please refer to Section 5.4 for General Test Cases.

4.3.4 Authentication Method Selection as a Testing Framework

According to later version of [ONVIF Network Interface Specs], it requires ONVIF client to support both HTTP digest and WS-UsernameToken functionality as authentication functionality. Therefore, ONVIF Client (ONVIF Device Test Tool in this context) as a testing framework shall properly select authentication method between the two based on the response from DUT toward specific request. The following is the deterministic procedure on which authentication method is to be selected.

Procedure:

1. ONVIF Client invokes a specific command which is under testing without any user credentials (no WS-UsernameToken, no HTTP digest authentication header).
2. If DUT returns correct response, then ONVIF Client determines that DUT does not require any user authentication toward the command according to the configured security policy.
3. If DUT returns HTTP 401 Unauthorized error along with WWW-Authentication: Digest header, then ONVIF Client determines that DUT supports HTTP digest authentication. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.
4. If the DUT returns SOAP fault (Sender/NotAuthorized) message, then ONVIF Client determines that WS-UsernameToken is supported by DUT. ONVIF Client shall provide with the proper level of user credential to continue the test procedure.

5 Recording Control Test Cases

5.1 Capabilities

5.1.1 GET SERVICES AND GET RECORDING CONTROL SERVICE CAPABILITIES CONSISTENCY

Test Case ID: RECORDING-1-1-4

Specification Coverage: Capability exchange (ONVIF Core Specification), GetServiceCapabilities (ONVIF Recording Control Service Specification)

Feature Under Test: GetServices, GetServiceCapabilities (Recording Control)

WSDL Reference: devicemgmt.wsdl, recording.wsdl

Test Purpose: To verify getting Recording Control Service using GetServices request. To verify Get Services and Recording Control Service Capabilities consistency.

Pre-Requisite: Recording Control Service is received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetServices** message with parameters:
   - IncludeCapability := false
4. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServicesWithoutCapabilities*
5. If *listOfServicesWithoutCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/recording/wsdl", FAIL the test and skip other steps.
6. Set *recordingServ* := item from *listOfServicesWithoutCapabilities* list with Namespace = "http://www.onvif.org/ver10/recording/wsdl".
7. If *recordingServ*.Capabilities is specified, FAIL the test and skip other steps.
8. ONVIF Client invokes **GetServices** message with parameters:

- IncludeCapability := true

9. The DUT responds with a **GetServicesResponse** message with parameters:
   - Service list =: *listOfServicesWithCapabilities*
10. If *listOfServicesWithCapabilities* does not contain item with Namespace = "http://www.onvif.org/ver10/recording/wsdl", FAIL the test and skip other steps.
11. Set *recordingServ* := item from *listOfServicesWithCapabilities* list with Namespace = "http://www.onvif.org/ver10/recording/wsdl".
12. If *recordingServ*.Capabilities is not specified, FAIL the test and skip other steps.
13. If *recordingServ*.Capabilities does not contain valid Capabilities element for Recording Control service from "http://www.onvif.org/ver10/recording/wsdl" namespace, FAIL the test and skip other steps.
14. ONVIF Client invokes **GetServiceCapabilities** (Recording) message.
15. The DUT responds with **GetServiceCapabilitiesResponse** message with parameters
    - Capabilities =: *cap*
16. If *cap* differs from *recordingServ*.Capabilities.Capabilities, FAIL the test and skip other steps.
17. If *cap*.MaxRate is zero or less and is not skipped, FAIL the test and skip other steps.
18. If *cap*.MaxTotalRate is zero or less and is not skipped, FAIL the test and skip other steps.
19. If *cap*.MaxRecordings is less than 1 and is not skipped, FAIL the test and skip other steps.
20. If *cap*.MaxRecordingJobs is zero or less and is not skipped, FAIL the test and skip other steps.
21. If *cap*.Encoding is empty or skipped, FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passed all assertions.

FAIL –

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

Note: The following fields are compared at step 16:

- DynamicRecordings
- DynamicTracks
- Encoding (order should not be taken in account for comparison)
- MaxRate
- MaxTotalRate
- MaxRecordings
- MaxRecordingJobs
- Options
- MetadataRecording
- SupportedExportFileFormats (order should not be taken in account for comparison)
- EventRecording
- BeforeEventLimit
- AfterEventLimit
- SupportedTargetFormats (order should not be taken in account for comparison)
- EncryptionEntryLimit
- SupportedEncryptionModes (order should not be taken in account for comparison)
- OverrideSegmentDuration
- AsymmetricEncryptionSupported

5.2 Recording

5.2.1 Internal Storage

5.2.1.1 Receiver

5.2.1.1.1 START RECORDING ON RECEIVER

Test Case ID: RECORDING-2-1-20

Specification Coverage: Creation of Recording Job with Active mode on receiver. Event generation when the state field of the RecordingJobStateInformation structure is changing (ONVIF Recording Control Service Specification)

Feature Under Test: CreateRecordingJob

WSDL Reference: recording.wsdl, receiver.wsdl, event.wsdl

Test Purpose: To verify Start Recording on Receiver.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- All recording jobs were stopped.
- Options are supported by DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
4. Execute Annex A.65 for auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. ONVIF Client will invoke **ConfigureReceiverRequest** message (ReceiverToken = ReceiverToken1, Configuration.Mode = "AlwaysConnect", Configuration.MediaUri as stream_uri of RTSP Simulator, Configuration.StreamSetup.Stream = "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to configure the receiver to receive media from RTSP Simulator.

8. Verify **ConfigureReceiverResponse** message from the DUT.
9. ONVIF Client will invoke **GetReceiverRequest** message with ReceiverToken = ReceiverToken1.
10. Verify **GetReceiverResponse** message from the DUT. Check that **GetReceiverResponse** message contains the same parameters values as the ones changed in **ConfigureReceiverRequest** message.
11. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start recording.
12. Verify **SetRecordingJobModeResponse** message from the DUT. Mark time of **CreateRecordingJobResponse** as T1.
13. Execute Annex A.62 for receiving event with "Active" or "PartiallyActive" Recording job state.
14. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
15. Verify the **GetRecordingJobStateResponse** message from the DUT.
16. Check that State.State is equal to "Active" or "PartiallyActive".
17. Check that State.Sources.State is equal to "Active".
18. Check that State.State value in the **GetRecordingJobStateResponse** message equals to State.State value in the notification message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetReceiverResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT sent **GetReceiverResponse** message with parameters values differ from sent in **ConfigureReceiverRequest** message.

- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Active" or "PartiallyActive".
- The DUT sent **GetRecordingJobStateResponse** message with State.Sources.State parameter value not equal to "Active".
- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".
- The DUT sent different values for State.State in the notification message and in **GetRecordingJobStateResponse** message.

5.2.1.1.2 STOP RECORDING ON RECEIVER - PUT JOB IN IDLE STATE

Test Case ID: RECORDING-2-1-22

Specification Coverage: SetRecordingJobMode, event generation when the state field of the RecordingJobStateInformation structure is changing (ONVIF Recording Control Service Specification).

Feature Under Test: SetRecordingJobMode

WSDL Reference: recording.wsdl, receiver.wsdl, event.wsdl

Test Purpose: To verify Stop Recording on Receiver by Putting It in Idle State.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- All recording jobs were stopped.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. Execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
4. Execute Annex A.65 for Auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. ONVIF Client will invoke **ConfigureReceiverRequest** message (ReceiverToken = ReceiverToken1, Configuration.Mode = "AlwaysConnect", Configuration.MediaUri as stream_uri of RTSP Simulator, Configuration.StreamSetup.Stream = "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to configure the receiver to receive media from RTSP Simulator.
8. Verify **ConfigureReceiverResponse** message from the DUT.
9. ONVIF Client will invoke **GetReceiverRequest** message with ReceiverToken = ReceiverToken1.
10. Verify **GetReceiverResponse** message from the DUT. Check that **GetReceiverResponse** message contains the same parameters values as were changed in **ConfigureReceiverRequest** message.
11. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start recording.
12. Verify **SetRecordingJobModeResponse** message from the DUT. Mark time of **CreateRecordingJobResponse** as T1.
13. Execute Annex A.62 for receiving event with "Active" or "PartiallyActive" Recording job state.
14. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
15. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State = "Active" or "PartiallyActive".
16. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
17. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

18. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Idle") to stop recording.
19. Verify **SetRecordingJobModeResponse** message from the DUT. Remark T1 as time of **SetRecordingJobModeResponse** message.
20. Execute Annex A.62 for receiving event with Idle state.
21. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
22. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State and State.Sources.State parameter values are equal to "Idle".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetReceiverResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **DeleteRecordingJobResponse** message.
- The DUT sent **GetReceiverResponse** message with parameters values differerent from sent in **ConfigureReceiverRequest** message.
- The DUT returned **GetRecordingJobStateResponse** State.State different from "Active" or "PartailActive" at step 15.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Idle" at step 22.
- The DUT sent **GetRecordingJobStateResponse** message with State.Sources.State parameter value not equal to "Idle" at step 22.
- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".

5.2.1.1.3 STOP RECORDING ON RECEIVER - NEVER CONNECTED MODE

Test Case ID: RECORDING-2-1-23

Specification Coverage: Event generation when the receiver state is changing (ONVIF Receiver Service Specification).

Feature Under Test: None

WSDL Reference: recording.wsdl, receiver.wsdl, event.wsdl

Test Purpose: To verify Stop Recording on Receiver by Putting Receiver in NeverConnect Mode.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- All recording jobs were stopped.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
4. Execute Annex A.65 for Auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:Receiver/ChangeState Topic as Filter and an InitialTerminationTime = Time1 to check Receiver state.
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. ONVIF Client will invoke **ConfigureReceiverRequest** message with ReceiverToken = ReceiverToken1 (Configuration.Mode = "AlwaysConnect", Configuration.MediaUri as stream_uri of RTSP Simulator, Configuration.StreamSetup.Stream

= "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to configure the receiver to receive media from RTSP Simulator.

8. Verify **ConfigureReceiverResponse** message from the DUT.
9. ONVIF Client will invoke **GetReceiverRequest** message with ReceiverToken = ReceiverToken1.
10. Verify **GetReceiverResponse** message from the DUT. Check that **GetReceiverResponse** message contains the same parameters values as were changed in **ConfigureReceiverRequest** message.
11. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start the recording.
12. Verify **SetRecordingJobModeResponse** message from the DUT. Mark the time of **SetRecordingJobModeRequest** message as T1.
13. Execute Annex A.63 for receiving event with Connected state of the receiver.
14. ONVIF Client will invoke **GetReceiverStateRequest** message to check Receiver State.
15. Verify **GetReceiverStateResponse** message from the DUT. Check that ReceiverState.State = "Connected".
16. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
17. Verify the **GetRecordingJobStateResponse** message from the DUT.
18. Check that State.State is equal to "Active" or "PartiallyActive".
19. Check that State.Sources.State is equal to "Active".
20. Check that State.State value in the **GetRecordingJobStateResponse** message equals to State.State value in the notification message.
21. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:Receiver/ChangeState Topic as Filter and an InitialTerminationTime = Time1 to check Receiver state.
22. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
23. ONVIF Client will invoke **SetReceiverModeRequest** message (ReceiverToken = "ReceiverToken1", Mode = "NeverConnect") to stop recording.

24. Verify **SetReceiverModeResponse** message from the DUT.
25. Execute Annex A.63 for receiving event with NotConnected state of the receiver.
26. ONVIF Client will invoke **GetReceiverStateRequest** message to check Receiver State.
27. Verify **GetReceiverStateResponse** message from the DUT. Check that ReceiverState.State = "NotConnected".
28. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
29. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State and State.Sources.State parameter values are equal to "Idle".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetReceiverResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetReceiverStateResponse** message.
- The DUT did not send a valid **SetReceiverModeResponse** message.
- The DUT sent **GetReceiverResponse** message with parameters values differ from sent in **ConfigureReceiverRequest** message.
- The DUT sent **GetReceiverStateResponse** with ReceiverState.State not equal to "Connected" at step 15.
- The DUT returned **GetRecordingJobStateResponse** State.State different from "Active" or "PartailActive" at step 18.
- The DUT sent **GetRecordingJobStateResponse** message with State.Sources.State parameter value not equal to "Active" at step 19.
- The DUT sent **GetReceiverStateResponse** with ReceiverState.State not equal to "NotConnected" at step 27.

- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".
- The DUT sent **GetRecordingJobStateResponse** message with State.State and State.Sources.State parameters values not equal to "Idle" at step 29.

5.2.1.1.4 STOP RECORDING ON RECEIVER - DELETE JOB

Test Case ID: RECORDING-2-1-25

Specification Coverage: DeleteRecordingJob (ONVIF Recording Control Service Specification)

Feature Under Test: DeleteRecordingJob

WSDL Reference: recording.wsdl, receiver.wsdl, event.wsdl

Test Purpose: To verify Stop Recording on Receiver by Job Deletion.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- All recording jobs were stopped.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
4. Execute Annex A.65 for auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.

6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. ONVIF Client will invoke **ConfigureReceiverRequest** message (ReceiverToken = ReceiverToken1, Configuration.Mode = "AlwaysConnect", Configuration.MediaUri as stream_uri of RTSP Simulator, Configuration.StreamSetup.Stream = "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to configure the receiver to receive media from RTSP Simulator.
8. Verify **ConfigureReceiverResponse** message from the DUT.
9. ONVIF Client will invoke **GetReceiverRequest** message with ReceiverToken = ReceiverToken1.
10. Verify **GetReceiverResponse** message from the DUT. Check that **GetReceiverResponse** message contains the same parameters values as were changed in **ConfigureReceiverRequest** message.
11. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start recording.
12. Verify **SetRecordingJobModeResponse** message from the DUT. Mark time of **CreateRecordingJobResponse** as T1.
13. Execute Annex A.62 for receiving event with "Active" or "PartiallyActive" Recording job state.
14. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
15. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State = "Active" or "PartiallyActive".
16. ONVIF Client will invoke **DeleteRecordingJobRequest** with JobToken = JobToken1 to delete recording job.
17. Verify **DeleteRecordingJobResponse** from the DUT.
18. ONVIF Client will invoke **GetRecordingJobsRequest** to get full list of the recording jobs.
19. Verify the **GetRecordingJobsResponse** from the DUT.
20. Check that there is no JobItem with JobToken1 in the **GetRecordingJobsResponse**.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetReceiverResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **DeleteRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT sent **GetReceiverResponse** message with parameters values different from the ones sent in **ConfigureReceiverRequest** message.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Active" or "PartiallyActive" at step 15.
- The DUT returned the JobToken1 in **GetRecordingJobsResponse** message after deleting.

5.2.1.1.5 MODIFY MEDIA ATTRIBUTE WHILE RECORDING - RECEIVER

Test Case ID: RECORDING-2-1-27

Specification Coverage: Event generation when the receiver state is changing (ONVIF Receiver Service Specification).

Feature Under Test: None

WSDL Reference: recording.wsdl, receiver.wsdl, event.wsdl

Test Purpose: To verify Media Attributes Modification for Remote Recordings while Recording.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- All recording jobs were stopped.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.

- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
4. Execute Annex A.65 for auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. ONVIF Client will invoke **ConfigureReceiverRequest** message (ReceiverToken = ReceiverToken1, Configuration.Mode = "AlwaysConnect", Configuration.MediaUri = stream_uri1, (stream_uri1 from RTSP Simulator), Configuration.StreamSetup.Stream = "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to configure the receiver to receive media from RTSP Simulator.
8. Verify **ConfigureReceiverResponse** message from the DUT.
9. ONVIF Client will invoke **GetReceiverRequest** message with ReceiverToken = ReceiverToken1.
10. Verify **GetReceiverResponse** message from the DUT. Check that **GetReceiverResponse** message contains the same parameters values as were changed in **ConfigureReceiverRequest** message.
11. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start recording.
12. Verify **SetRecordingJobModeResponse** message from the DUT. Mark time of **CreateRecordingJobResponse** as T1.
13. Execute Annex A.62 for receiving event with "Active" or "PartiallyActive" Recording job state.
14. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".

15. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State = "Active" or "PartiallyActive".

16. ONVIF Client will invoke **ConfigureReceiverRequest** message (ReceiverToken = ReceiverToken1, Configuration.Mode = "AlwaysConnect", Configuration.MediaUri = stream_uri2, (where stream_uri2 is streaming of RTSP Simulator with resolution, bitrate, framerate and quality are different from stream_uri1), Configuration.StreamSetup.Stream = "RTP-Unicast", StreamSetup.Transport.Tunnel.Protocol = "UDP", no StreamSetup.Transport.Tunnel.Tunnel) to modify stream settings of the RTSP simulator.

17. Verify **ConfigureReceiverResponse** message from the DUT.

18. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".

19. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State = "Active" or "PartiallyActive".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetReceiverResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT returned Recording parameters in **GetRecordingsResponse** message that differ from specified during Recording changing.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Active" or "PartiallyActive" at step 15 or 19.
- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".

5.2.1.1.6 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON RECEIVER)

Test Case ID: RECORDING-3-1-12

Specification Coverage: Priority scheme for recording jobs (ONVIF Recording Control Service Specification).

Feature Under Test: None

WSDL Reference: recording.wsdl

Test Purpose: To verify creation of recording jobs with different priorities.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Receiver Service by GetServices command.
- There are no active recording jobs.
- Options are supported by the DUT.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.64 with RequiredSpareJobs = 1 to create or select Recording to make sure that 2 recording jobs can be created.
4. Execute Annex A.65 for auto creation of receiver by create recording job with Idle mode.
5. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Active") to start recording.
6. Verify **SetRecordingJobModeResponse** message from the DUT.
7. Wait for JobState to get changed (use Operation Delay Time).
8. ONVIF Client will invoke **GetRecordingJobStateRequest** (JobToken = "JobToken1") request message to retrieve recording job state.
9. Verify the **GetRecordingJobStateResponse** from the DUT.

10. Check that State.State parameter value is equal to "Active" or "PartiallyActive".
11. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = "RecordingToken1", JobConfiguration.Mode = "Active", JobConfiguration.Priority = 2, JobConfiguration.Source.SourceToken.Token = "ReceiverToken1", JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Receiver", JobConfiguration.Source.AutoCreateReceiver = false) to create a recording job for the same receiver with higher priority.
12. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken2).
13. Wait for JobState to get changed (use Operation Delay Time).
14. ONVIF Client will invoke **GetRecordingJobStateRequest** (JobToken = JobToken1) request message to retrieve recording job state.
15. Verify the **GetRecordingJobStateResponse** from the DUT.
16. Check that State.State parameter value is equal to "Idle".
17. ONVIF Client will invoke **GetRecordingJobStateRequest** (JobToken = JobToken2) request message to retrieve recording job state.
18. Verify the **GetRecordingJobStateResponse** from the DUT.
19. Check that State.State parameter value is equal to "Active" or "PartiallyActive".
20. ONVIF Client will invoke **DeleteRecordingJobRequest** message (JobToken = JobToken2) to delete Recording Job.
21. Verify the **DeleteRecordingJobResponse** message from the DUT.
22. Wait for JobState to get changed (use Operation Delay Time).
23. ONVIF Client will invoke **GetRecordingJobStateRequest** (JobToken = JobToken1) request message to retrieve recording job state.
24. Verify the **GetRecordingJobStateResponse** from the DUT.
25. Check that State.State parameter value is equal to "Active" or "PartiallyActive". Check that State.State value equals to State.State value at step 9.
26. ONVIF Client will invoke **DeleteRecordingJobRequest** message (JobToken = JobToken1) to delete Recording Job.
27. Verify the **DeleteRecordingJobResponse** message from the DUT.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **ConfigureReceiverResponse** message.
- The DUT did not send a valid **GetRecordingOptionsResponse** message.
- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **DeleteRecordingJobResponse** message.
- The DUT returned incorrect RecordingJobState in **GetRecordingJobStateResponse** message at steps 9, 15, 18, 24.
- The DUT sent **GetRecordingJobStateResponse** at step 23 with State.State value different from State.State value at step 9.

5.2.1.2 Media Profile

5.2.1.2.1 START RECORDING ON MEDIA PROFILE

Test Case ID: RECORDING-2-1-28

Specification Coverage: Creation of a Recording Job with Active mode on local storage. Event generation when the state field of the RecordingJobStateInformation structure is changing (ONVIF Recording Control Service Specification).

Feature Under Test: CreateRecordingJob

WSDL Reference: recording.wsdl, media.wsdl, event.wsdl

Test Purpose: To verify Start Recording using a Media Profile.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Media Service by GetServices command.
- At least one media profile compatible with at least one existing or created recording exists on the DUT.
- All recording jobs were stopped.

- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.66 with RequiredSpareJobs = 0 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 1 recording job can be created and to get the compatible media profile tokens list.
4. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
5. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
6. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Active, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job for configured recording with Active mode.
7. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken1).
8. Verify that JobConfiguration in **CreateRecordingJobResponse** message contains the same parameters values as was sent in **CreateRecordingJobRequest** message.
9. Execute Annex A.67 for receiving Initialized event with "Active" or "PartiallyActive" Recording job state.
10. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
11. Verify the **GetRecordingJobStateResponse** message from the DUT.
12. Check that State.State is equal to "Active" or "PartiallyActive".

13. Check that State.Sources.State is equal to "Active".
14. Check that State.State value in the **GetRecordingJobStateResponse** message equals to State.State value in the notification message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionResponse** message.
- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT sent **CreateRecordingJobResponse** message with JobConfiguration parameters values differ from sent in **CreateRecordingJobRequest** message.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Active" or "PartiallyActive".
- The DUT sent **GetRecordingJobStateResponse** message with State.Sources.State parameter value not equal to "Active".
- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".
- The DUT sent different values for State.State in the notification message and in **GetRecordingJobStateResponse** message.

5.2.1.2.2 STOP RECORDING ON MEDIA PROFILE - PUT JOB IN IDLE STATE

Test Case ID: RECORDING-2-1-29

Specification Coverage: SetRecordingJobMode, event generation when the state field of the RecordingJobStateInformation structure is changing (ONVIF Recording Control Service Specification).

Feature Under Test: SetRecordingJobMode

WSDL Reference: recording.wsdl, media.wsdl, event.wsdl

Test Purpose: To verify Stop Recording using a Media Profile by Putting It in Idle State.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Media Service by GetServices command.
- At least one media profiles compatible with at least one existing or created recording exists on the DUT.
- All recording jobs were stopped.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.66 with RequiredSpareJobs = 0 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 1 recording job can be created and to get the compatible media profile tokens list.
4. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
5. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
6. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Active, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job for configured recording with Active mode.
7. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken1).
8. Verify that JobConfiguration in **CreateRecordingJobResponse** message contains the same parameters values as was sent in **CreateRecordingJobRequest** message.

9. Execute Annex A.67 for receiving Initialized event with "Active" or "PartiallyActive" Recording job state.
10. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
11. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State parameter value is equal to "Active" or "PartiallyActive".
12. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.
13. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
14. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = "JobToken1", Mode = "Idle") to stop recording.
15. Verify **SetRecordingJobModeResponse** message from the DUT. Remark Time1 as time of receiving of **SetRecordingJobModeResponse** message
16. Execute Annex A.62 for receiving changed event with "Idle" Recording job state.
17. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
18. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State and State.Sources.State parameter values are equal to "Idle".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **SetRecordingJobModeResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message

- The DUT returned JobConfiguration parameters in **CreateRecordingJobResponse** message that differs from specified during RecordingJob creation.
- The DUT returned **GetRecordingJobStateResponse** State.State differs from "Active" or "PartiallyActive" at step 11.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Idle" at step 18.
- The DUT sent **GetRecordingJobStateResponse** message with State.Sources.State parameter value not equal to "Idle" at the step 18.
- The DUT sent **GetRecordingJobStateResponse** message with State.RecordingToken parameter value not equal to "RecordingToken1".

5.2.1.2.3 STOP RECORDING ON MEDIA PROFILE - DELETE JOB

Test Case ID: RECORDING-2-1-30

Specification Coverage: DeleteRecordingJob (ONVIF Recording Control Service Specification)

Feature Under Test: DeleteRecordingJob

WSDL Reference: recording.wsdl, media.wsdl, event.wsdl

Test Purpose: To verify Stop Recording using a Media Profile by Job Deletion.

Pre-Requisite: ONVIF Client gets the entry points for Recording Control Service and Media Service by GetServices command. All recording jobs were stopped. At least one media profile compatible with at least one existing or created recording exists on the DUT. Options are supported by the DUT. Device supports Pull-Point Notification feature.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Execute Annex A.66 with RequiredSpareJobs = 0 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 1 recording job can be created and to get the compatible media profile tokens list.
4. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime = TerminationTime1 to check Recording Job State changing.

5. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
6. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Active, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job for configured recording with Active mode.
7. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken1).
8. Verify that JobConfiguration in **CreateRecordingJobResponse** message contains the same parameters values as was sent in **CreateRecordingJobRequest** message.
9. Execute Annex A.67 for receiving Initialized event with "Active" or "PartiallyActive" Recording job state.
10. ONVIF Client will invoke **GetRecordingJobStateRequest** message with JobToken = "JobToken1".
11. Verify the **GetRecordingJobStateResponse** message from the DUT. Check that State.State and State.Sources.State parameter values are equal to "Active" or "PartiallyActive".
12. ONVIF Client will invoke **DeleteRecordingJobRequest** message (JobToken = JobToken1) to delete Recording Job.
13. Verify the **DeleteRecordingJobResponse** message from the DUT.
14. ONVIF Client will invoke **GetRecordingJobsRequest** to get full list of the recording jobs.
15. Verify the **GetRecordingJobsResponse** from the DUT.
16. Check that there is no JobItem with JobToken1 in the **GetRecordingJobsResponse**.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreatePullPointSubscriptionRequest** message
- The DUT did not send a valid **CreateRecordingJobResponse** message.

- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT sent **CreateRecordingJobResponse** message with JobConfiguration parameters values different from sent in **CreateRecordingJobRequest** message.
- The DUT sent **GetRecordingJobStateResponse** message with State.State parameter value not equal to "Active" or "PartiallyActive.
- The DUT returned the JobToken1 in **GetRecordingJobsResponse** message after deleting.

5.2.1.2.4 RECORDING JOB CONFIGURATION - DIFFERENT PRIORITIES (ON MEDIA PROFILE)

Test Case ID: RECORDING-3-1-11

Specification Coverage: Priority scheme for recording jobs (ONVIF Recording Control Service Specification).

Feature Under Test: None

WSDL Reference: recording.wsdl

Test Purpose: To verify creation of recording jobs with different priorities.

Pre-Requisite:

- ONVIF Client gets the entry points for Recording Control Service and Media Service by GetServices command.
- No active recording jobs.
- At least one media profile compatible with at least one existing or created recording exists on the DUT.
- Options are supported by the DUT.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. Execute Annex A.66 with RequiredSpareJobs = 1 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 2 recording jobs can be created and to get the compatible media profile tokens list.
4. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Active, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job for configured recording with Active mode.
5. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken1).
6. Wait for JobState to get changed (use Operation Delay Time).
7. ONVIF Client will invoke **GetRecordingJobStateRequest** message (JobToken = JobToken1) to get actual recording job state.
8. Verify the **GetRecordingJobStateResponse** message from the DUT.
9. Check that State.State parameter value is "Active" or "PartiallyActive".
10. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Active, JobConfiguration.Priority = 2, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile configured for recording, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a new recording job for configured recording with Active mode and higher priority.
11. Verify the **CreateRecordingJobResponse** message (JobToken = JobToken2).
12. Wait for JobState to get changed (use Operation Delay Time).
13. ONVIF Client will invoke **GetRecordingJobStateRequest** message (JobToken = JobToken1) to get actual recording job state.
14. Verify the **GetRecordingJobStateResponse** message from the DUT.
15. Check that State.State parameter value is "Idle".
16. ONVIF Client will invoke **GetRecordingJobStateRequest** message (JobToken = JobToken2) to get actual recording job state.
17. Verify the **GetRecordingJobStateResponse** message from the DUT.

18. Check that State.State parameter value is "Active" or "PartiallyActive".
19. ONVIF Client will invoke **DeleteRecordingJobRequest** message (JobToken = JobToken2) to delete Recording Job.
20. Verify the **DeleteRecordingJobResponse** message from the DUT.
21. Wait for JobState to get changed (use Operation Delay Time).
22. ONVIF Client will invoke **GetRecordingJobStateRequest** message (JobToken = JobToken1) to get actual recording job state.
23. Verify the **GetRecordingJobStateResponse** message.
24. Check that State.State parameter value is "Active" or "PartiallyActive". Check that State.State value equals to State.State value at step 8.
25. ONVIF Client will invoke **DeleteRecordingJobRequest** message (JobToken = JobToken1) to delete Recording Job.
26. Verify the **DeleteRecordingJobResponse** message from the DUT.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingOptionsResponse** message.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT did not send a valid **DeleteRecordingJobResponse** message.
- The DUT returned RecordingJob parameters in **CreateRecordingJobResponse** message that differ from specified during RecordingJob creation.
- The DUT returned incorrect RecordingJobState in **GetRecordingJobStateResponse** message at steps 8, 14, 17 or 23.
- The DUT sent **GetRecordingJobStateResponse** at step 23 with State.State value different from State.State value at step 8.

5.2.2 Object Storage

5.2.2.1 Media2 Profile

5.2.2.1.1 OBJECT STORAGE RECORDING - START AND STOP RECORDING

Test Case ID: RECORDING-2-2-1

Specification Coverage: SetRecordingJobMode, Recording job state changes, Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: SetRecordingJobMode, tns1:RecordingConfig/JobState, Recording to Object Storage

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording to object storage.
- To verify start of the recording by changing recording job state with SetRecordingJobMode command.
- To verify stop of the recording by changing recording job state with SetRecordingJobMode command.
- To verify receiving recording state with tns1:RecordingConfig/JobState event.
- To verify that the DUT will open a new span and segment on recording start.
- To verify that the DUT will close a span and segment on recording stop.
- To verify segment object key without prefix and postfix configured.
- To verify span object key.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - in **tns1:RecordingConfig/JobState** - Notification Topic
   - out *s* - Subscription Reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription Termination time
6. ONVIF Client retrieves and checks **tns1:RecordingConfig/JobState** event for the specified recording job by following the procedure mentioned in Annex A.7 with the following parameters:
   - in *s* - Subscription reference
   - in *currentTime* - current time for the DUT
   - in *terminationTime* - subscription termination time
   - in *recordingJobToken1* - expected recording job token
   - in **Idle** - expected recording job state
   - out *recordingJobStateInitial* - received recording job state

7. If *recordingJobStateInitial*.RecordingToken is not equal to *recordingToken1*, FAIL the test, restore DUT settings and skip other steps.
8. If *recordingJobStateInitial*.State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
9. If *recordingJobStateInitial*.Sources[0].SourceToken.Type is not equal to *recordingJobConfiguration1*.Sources[0].SourceToken.Type, FAIL the test, restore DUT settings and skip other steps.
10. If *recordingJobStateInitial*.Sources[0].SourceToken.Token is not equal to *recordingJobConfiguration1*.Sources[0].SourceToken.Token, FAIL the test, restore DUT settings and skip other steps.
11. If *recordingJobStateInitial*.Sources[0].State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
12. If for at least one Track at *recordingJobStateInitial*.Sources[0].Tracks State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
13. If for at least one Track at *recordingJobStateInitial*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.
14. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Active**
15. The DUT responds with **SetRecordingJobModeResponse** message.
16. ONVIF Client retrieves and checks **tns1:RecordingConfig/JobState** event for the specified recording job by following the procedure mentioned in Annex A.7 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingJobToken1* - expected recording job token
    - in **Active** - expected recording job state
    - out *recordingJobState1* - received recording job state

17. If *recordingJobState1*.State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

18. If *recordingJobState1*.Sources[0].State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

19. If for at least one Track at *recordingJobState1*.Sources[0].Tracks State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

20. If for at least one Track at *recordingJobState1*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.

21. Wait for *recordingSegmentDuration* + 20%.

22. ONVIF Client verifies recorded data:

    22.1. ONVIF Client finds the latest span *span1* by checking segments object keys dates.

    22.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.

    22.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

    22.4. ONVIF Client parses *segment1* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

    - *lead1* - value of **lead** of object key
    - *date1* - value of **date** of *lead1*
    - *prefix1* (optional) - value of **prefix** of *lead1*
    - *postfix1* (optional) - value of **postfix** of *lead1*
    - *time1* - value of **time** of object key
    - *counter1* - value of **counter** of object key
    - *extension1* - value of **extension** of object key

    22.5. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

    22.6. If *lead1* is not equal to *date1* (no prefix and postfix and no "/" symbols), FAIL the test, restore DUT settings and skip other steps.

22.7. If *time1* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

22.8. Set *spanUTCTime1* := combination of *date1* and *time1*.

22.9. If *spanUTCTime1* is earlier than **SetRecordingJobMode** request with recording job activation, FAIL the test, restore DUT settings and skip other steps.

22.10 If *spanUTCTime1* is later than **SetRecordingJobMode** request with recording job activation + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

23. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Idle**
24. The DUT responds with **SetRecordingJobModeResponse** message.
25. ONVIF Client retrieves and checks **tns1:RecordingConfig/JobState** event for the specified recording job by following the procedure mentioned in Annex A.7 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingJobToken1* - expected recording job token
    - in **Idle** - expected recording job state
    - out *recordingJobState2* - received recording job state
26. If *recordingJobState2*.State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
27. If *recordingJobState2*.Sources[0].State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
28. If for at least one Track at *recordingJobState2*.Sources[0].Tracks State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.
29. If for at least one Track at *recordingJobState2*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.

30. ONVIF Client verifies recorded data using *span1* found before:

    30.1. At *span1* ONVIF Client finds segment *segment2* with maximum **counter** by checking segments object keys dates.

    30.2. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

    30.3. ONVIF Client parses *segment2* object key based on the rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

    - *lead2* - value of **lead** of object key
    - *date2* - value of **date** of *lead2*
    - *prefix2* (optional) - value of **prefix** of *lead2*
    - *postfix2* (optional) - value of **postfix** of *lead2*
    - *time2* - value of **time** of object key
    - *counter2* - value of **counter** of object key
    - *extension2* - value of **extension** of object key

    30.4. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

    30.5. If *lead2* is not equal to *date2* (no prefix and postfix and no "/" symbols), FAIL the test, restore DUT settings and skip other steps.

    30.6. If *time2* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

    30.7. Set *spanUTCTime2* := combination of *date2* and *time2*.

    30.8. If *spanUTCTime2* is earlier than **SetRecordingJobMode** request with recording job deactivation, FAIL the test, restore DUT settings and skip other steps.

    30.9. If *spanUTCTime2* is later than **SetRecordingJobMode** request with recording job deactivation + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

    30.10 If *extension2* is not equal to *extension1* in combination with **_end**, FAIL the test, restore DUT settings and skip other steps.

31. ONVIF Client closes event subscription if required.

32. ONVIF Client restores DUT configuration if required.

33. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.2 OBJECT STORAGE RECORDING - START AND STOP RECORDING, JOB PRIORITIES

Test Case ID: RECORDING-2-2-2

Specification Coverage: CreateRecordingJob, DeleteRecordingJob, GetRecordingJobState, RecordingJobConfiguration, Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: CreateRecordingJob, DeleteRecordingJob, Recording to Object Storage

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording to object storage.
- To verify start of the recording by creating a recording job in Active state with CreateRecordingJob command.
- To verify recording with different priorities. (**Note:** check will be done only if the DUT allows creating more than 1 recording job per record.)
- To verify stop of the recording by deletion of recording job with DeleteRecordingJob command.
- To verify receiving recording state with GetRecordingJobState command.
- To verify that the DUT will open new span and segment on recording start.

- To verify that the DUT will close a span and segment on recording stop.
- To verify segment object key with prefix and postfix configured.
- To verify span object key.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with a recording job for media2 profile with at least video configured by following the procedure mentioned in Annex A.9 with the following input and output parameters:
   - in **"Prefix1"** - recording prefix
   - in **"Postfix1"** - recording postfix
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *source1* - source for recording job for test
   - out *spareJobs1* - number of jobs that could be created for the recording
4. Set:
   - *recordingJob1Configuration*.ScheduleToken is skipped
   - *recordingJob1Configuration*.RecordingToken := *recordingToken1*
   - *recordingJob1Configuration*.Mode := **Active**

- *recordingJob1Configuration*.Priority := 3
- *recordingJob1Configuration*.Source[0] := *source1*
- *recordingJob1Configuration*.Extension is skipped
- *recordingJob1Configuration*.EventFilter is skipped

5. ONVIF Client invokes **CreateRecordingJob** request with parameters:
   - JobConfiguration := *recordingJob1Configuration*
6. The DUT responds with **CreateRecordingJobResponse** message with parameters:
   - JobToken =: *recordingJob1Token*
   - JobConfiguration =: *createdRecordingJob1Configuration*
7. Wait for *operationDelay*.
8. ONVIF Client invokes **GetRecordingJobState** request with parameters
   - JobToken := *recordingJob1Token*
9. The DUT responds with **GetRecordingJobStateResponse** message with the following parameters:
   - State =: *recordingJob1StateInitial*
10. If *recordingJob1StateInitial*.State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.
11. If *recordingJob1StateInitial*.Sources[0].State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.
12. If for at least one Track at *recordingJob1StateInitial*.Sources[0].Tracks State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.
13. If for at least one Track at *recordingJob1StateInitial*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.
14. ONVIF Client verifies recorded data:
    14.1. ONVIF Client finds latest span *span1* by checking segments object keys dates.
    14.2. At *span1* ONVIF Client finds segment *span1_segment1* with **counter** = 0 by checking segments object keys dates.

14.3. If ONVIF Client was not able to find *span1_segment1*, FAIL the test, restore DUT settings and skip other steps.

14.4. ONVIF Client parses *span1_segment1* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span1_segment1_lead* - value of **lead** of object key
- *span1_segment1_date* - value of **date** of *span1_segment1_lead*
- *span1_segment1_prefix* (optional) - value of **prefix** of *span1_segment1_lead*
- *span1_segment1_postfix* (optional) - value of **postfix** of *span1_segment1_lead*
- *span1_segment1_time* - value of **time** of object key
- *span1_segment1_counter* - value of **counter** of object key
- *span1_segment1_extension* - value of **extension** of object key

14.5. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

14.6. If *span1_segment1_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

14.7. If *span1_segment1_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

14.8. If *span1_segment1_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

14.9. Set *span1_segment1_spanUTCTime* := combination of *span1_segment1_date* and *span1_segment1_time*.

14.10 If *span1_segment1_spanUTCTime* earlier than **CreateRecordingJob** request with recording job creation, FAIL the test, restore DUT settings and skip other steps.

14.11 If *span1_segment1_spanUTCTime* later than **CreateRecordingJob** request with recording job creation + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

15. If *spareJobs1* > 1:

15.1. Set:

- *recordingJob2Configuration*.ScheduleToken is skipped
- *recordingJob2Configuration*.RecordingToken := *recordingToken1*
- *recordingJob2Configuration*.Mode := **Active**
- *recordingJob2Configuration*.Priority := 15
- *recordingJob2Configuration*.Source[0] := *source1*
- *recordingJob2Configuration*.Extension is skipped
- *recordingJob2Configuration*.EventFilter is skipped

15.2. ONVIF Client invokes **CreateRecordingJob** request with parameters:

- JobConfiguration := *recordingJob2Configuration*

15.3. The DUT responds with **CreateRecordingJobResponse** message with parameters:

- JobToken =: *recordingJob2Token*
- JobConfiguration =: *createdRecordingJob2Configuration*

15.4. Wait for *operationDelay*.

15.5. ONVIF Client invokes **GetRecordingJobState** request with parameters

- JobToken := *recordingJob2Token*

15.6. The DUT responds with **GetRecordingJobStateResponse** message with the following parameters:

- State =: *recordingJob2StateInitial*

15.7. If *recordingJob2StateInitial*.State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.8. If *recordingJob2StateInitial*.Sources[0].State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.9. If for at least one Track at *recordingJob2StateInitial*.Sources[0].Tracks State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.10 If for at least one Track at *recordingJob2StateInitial*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.

15.11. ONVIF Client verifies recorded data:

15.11.1. ONVIF Client finds latest span *span2* by checking segments object keys dates.

15.11.2. At *span2* ONVIF Client finds segment *span2_segment1* with **counter** = 0 by checking segments object keys dates.

15.11.3. If ONVIF Client was not able to find *span2_segment1*, FAIL the test, restore DUT settings and skip other steps.

15.11.4. ONVIF Client parses *span2_segment1* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span2_segment1_lead* - value of **lead** of object key
- *span2_segment1_date* - value of **date** of *span2_segment1_lead*
- *span2_segment1_prefix* (optional) - value of **prefix** of *span2_segment1_lead*
- *span2_segment1_postfix* (optional) - value of **postfix** of *span2_segment1_lead*
- *span2_segment1_time* - value of **time** of object key
- *span2_segment1_counter* - value of **counter** of object key
- *span2_segment1_extension* - value of **extension** of object key

15.11.5. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

15.11.6. If *span2_segment1_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

15.11.7. If *span2_segment1_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

15.11.8. If *span2_segment1_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

15.11.9. Set *span2_segment1_spanUTCTime* := combination of *span2_segment1_date* and *span2_segment1_time*.

15.11.10 If *span2_segment1_spanUTCTime* earlier than **CreateRecordingJob** request with second recording job creation, FAIL the test, restore DUT settings and skip other steps.

15.11.11 If *span2_segment1_spanUTCTime* later than **CreateRecordingJob** request with second recording job creation + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

15.12 ONVIF Client invokes **GetRecordingJobState** request with parameters

- JobToken := *recordingJob1Token*

15.13 The DUT responds with **GetRecordingJobStateResponse** message with the following parameters:

- State =: *recordingJob1State2*

15.14 If *recordingJob1State2*.State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.

15.15 If *recordingJob1State2*.Sources[0].State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.

15.16 If for at least one Track at *recordingJob1State2*.Sources[0].Tracks State is not equal to **Idle**, FAIL the test, restore DUT settings and skip other steps.

15.17 If for at least one Track at *recordingJob1State2*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.

15.18 ONVIF Client verifies recorded data using *span1* found before:

15.18.1 At *span1* ONVIF Client finds segment *span1_segment2* with maximum **counter** by checking segments object keys dates.

15.18.2 If ONVIF Client was not able to find *span1_segment2*, FAIL the test, restore DUT settings and skip other steps.

15.18.3 ONVIF Client parses *span1_segment2* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span1_segment2_lead* - value of **lead** of object key
- *span1_segment2_date* - value of **date** of *span1_segment2_lead*

- *span1_segment2_prefix* (optional) - value of **prefix** of *span1_segment2_lead*
- *span1_segment2_postfix* (optional) - value of **postfix** of *span1_segment2_lead*
- *span1_segment2_time* - value of **time** of object key
- *span1_segment2_counter* - value of **counter** of object key
- *span1_segment2_extension* - value of **extension** of object key

15.18.4. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

15.18.5. If *span1_segment2_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

15.18.6. If *span1_segment2_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

15.18.7. If *span1_segment2_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

15.18.8. Set *span1_segment2_spanUTCTime* := combination of *span1_segment2_date* and *span1_segment2_time*.

15.18.9. If *span1_segment2_spanUTCTime* earlier than **CreateRecordingJob** request with second recording job creation, FAIL the test, restore DUT settings and skip other steps.

15.18.10. If *span1_segment2_spanUTCTime* later than **CreateRecordingJob** request with second recording job creation + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

15.18.11. If *span1_segment2_extension* is not equal to *span1_segment1_extension* in combination with **_end**, FAIL the test, restore DUT settings and skip other steps.

15.19. ONVIF Client invokes **DeleteRecordingJob** request with the following parameters:

- JobToken := *recordingJob2Token*

15.20. The DUT responds with **DeleteRecordingJobResponse** message.

15.21 Wait for *operationDelay*.

15.22 ONVIF Client verifies recorded data using *span2* found before:

15.22.1 At *span2* ONVIF Client finds segment *span2_segment2* with maximum **counter** by checking segments object keys dates.

15.22.2 If ONVIF Client was not able to find *span2_segment2*, FAIL the test, restore DUT settings and skip other steps.

15.22.3 ONVIF Client parses *span2_segment2* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span2_segment2_lead* - value of **lead** of object key
- *span2_segment2_date* - value of **date** of *span2_segment2_lead*
- *span2_segment2_prefix* (optional) - value of **prefix** of *span2_segment2_lead*
- *span2_segment2_postfix* (optional) - value of **postfix** of *span2_segment2_lead*
- *span2_segment2_time* - value of **time** of object key
- *span2_segment2_counter* - value of **counter** of object key
- *span2_segment2_extension* - value of **extension** of object key

15.22.4 If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

15.22.5 If *span2_segment2_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

15.22.6 If *span2_segment2_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

15.22.7 If *span2_segment2_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

15.22.8 Set *span2_segment2_spanUTCTime* := combination of *span2_segment2_date* and *span2_segment2_time*.

15.22.9 If *span2_segment2_spanUTCTime* earlier than **DeleteRecordingJob** request with second recording job deletion, FAIL the test, restore DUT settings and skip other steps.

15.22.10 If *span2_segment2_spanUTCTime* later than **DeleteRecordingJob** request with second recording job deletion + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

15.22.11 If *span2_segment2_extension* is not equal to *span2_segment1_extension* in combination with **_end**, FAIL the test, restore DUT settings and skip other steps.

15.23 ONVIF Client invokes **GetRecordingJobState** request with parameters

- JobToken := *recordingJob1Token*

15.24 The DUT responds with **GetRecordingJobStateResponse** message with the following parameters:

- State =: *recordingJob1State3*

15.25 If *recordingJob1State3*.State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.26 If *recordingJob1State3*.Sources[0].State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.27 If for at least one Track at *recordingJob1State3*.Sources[0].Tracks State is not equal to **Active**, FAIL the test, restore DUT settings and skip other steps.

15.28 If for at least one Track at *recordingJob1State3*.Sources[0].Tracks Error is not skipped or empty, FAIL the test, restore DUT settings and skip other steps.

15.29 ONVIF Client verifies recorded data:

15.29.1 ONVIF Client finds latest span *span3* by checking segments object keys dates.

15.29.2 In *span3* ONVIF Client finds segment *span3_segment1* with **counter** = 0 by checking segments object keys dates.

15.29.3 If ONVIF Client was not able to find *span3_segment1*, FAIL the test, restore DUT settings and skip other steps.

15.294. ONVIF Client parses *span3_segment1* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span3_segment1_lead* - value of **lead** of object key
- *span3_segment1_date* - value of **date** of *span3_segment1_lead*
- *span3_segment1_prefix* (optional) - value of **prefix** of *span3_segment1_lead*
- *span3_segment1_postfix* (optional) - value of **postfix** of *span3_segment1_lead*
- *span3_segment1_time* - value of **time** of object key
- *span3_segment1_counter* - value of **counter** of object key
- *span3_segment1_extension* - value of **extension** of object key

15.295. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

15.296. If *span3_segment1_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

15.297. If *span3_segment1_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

15.298. If *span3_segment1_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

15.299. Set *span3_segment1_spanUTCTime* := combination of *span3_segment1_date* and *span3_segment1_time*.

15.2910. If *span3_segment1_spanUTCTime* earlier than **DeleteRecordingJob** request with second recording job deletion, FAIL the test, restore DUT settings and skip other steps.

15.2911. If *span3_segment1_spanUTCTime* later than **DeleteRecordingJob** request with second recording job deletion + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

16. If *spareJobs1* = 1:

16.1. ONVIF Client invokes **DeleteRecordingJob** request with the following parameters:

- JobToken := *recordingJob1Token*

16.2. The DUT responds with **DeleteRecordingJobResponse** message.

16.3. Wait for *operationDelay*.

16.4. ONVIF Client verifies recorded data using *span1* found before:

16.4.1. At *span1* ONVIF Client finds segment *span1_segment2* with maximum **counter** by checking segments object keys dates.

16.4.2. If ONVIF Client was not able to find *span1_segment2*, FAIL the test, restore DUT settings and skip other steps.

16.4.3. ONVIF Client parses *span1_segment2* object key based on rules defined in B.4 Segment objects of ONVIF Recording Control Service Specification with the following values as result:

- *span1_segment2_lead* - value of **lead** of object key
- *span1_segment2_date* - value of **date** of *span1_segment2_lead*
- *span1_segment2_prefix* (optional) - value of **prefix** of *span1_segment2_lead*
- *span1_segment2_postfix* (optional) - value of **postfix** of *span1_segment2_lead*
- *span1_segment2_time* - value of **time** of object key
- *span1_segment2_counter* - value of **counter** of object key
- *span1_segment2_extension* - value of **extension** of object key

16.4.4. If ONVIF Client was not able to parse object key, FAIL the test, restore DUT settings and skip other steps.

16.4.5. If *span1_segment2_prefix* is not equal to **"Prefix1"**, FAIL the test, restore DUT settings and skip other steps.

16.4.6. If *span1_segment2_postfix* is not equal to **"Postfix1"**, FAIL the test, restore DUT settings and skip other steps.

16.4.7. If *span1_segment2_time* does not contain **Z** at the end, FAIL the test, restore DUT settings and skip other steps.

16.4.8. Set *span1_segment2_spanUTCTime* := combination of *span1_segment2_date* and *span1_segment2_time*.

16.4.9. If *span1_segment2_spanUTCTime* earlier than **DeleteRecordingJob** request with recording job deletion, FAIL the test, restore DUT settings and skip other steps.

16.4.10. If *span1_segment2_spanUTCTime* later than **DeleteRecordingJob** request with recording job deletion + *operationDelay*, FAIL the test, restore DUT settings and skip other steps.

16.4.11. If *span1_segment2_extension* is not equal to *span1_segment1_extension* in combination with **_end**, FAIL the test, restore DUT settings and skip other steps.

17. ONVIF Client restores DUT configuration if required.

18. ONVIF Client deletes all recorded data from external storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingJobStateResponse** message(s).
- The DUT did not send **DeleteRecordingJobResponse** message(s).

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

5.2.2.1.3 OBJECT STORAGE RECORDING - STORAGE IS NOT ACCESSIBLE

Test Case ID: RECORDING-2-2-3

Specification Coverage: GetRecordingJobState (ONVIF Recording Control Service Specification)

Feature Under Test: GetRecordingJobState

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording to object storage.
- To verify that the DUT reports a recording job error state if the object storage is not accessible.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with a recording job for a Media2 profile with at least video configured by following the procedure mentioned in Annex A.10 with the following input and output parameters:
   - out *recordingJobToken1* - recording job for test
4. Wait for *operationDelay*.
5. ONVIF Client invokes **GetRecordingJobState** request with parameters
   - JobToken := *recordingJobToken1*
6. The DUT responds with **GetRecordingJobStateResponse** message with the following parameters:
   - State =: *recordingJob1StateInitial*
7. If *recordingJob1StateInitial*.State is not equal to **Error**, FAIL the test, restore the DUT settings and skip other steps.
8. If *recordingJob1StateInitial*.Sources[0].State is not equal to **Error**, FAIL the test, restore the DUT settings and skip other steps.
9. ONVIF Client restores DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingJobStateResponse** message(s).

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

5.2.2.1.4 OBJECT STORAGE RECORDING - VIDEO RECORDING (H.264, MP4)

Test Case ID: RECORDING-2-2-4

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, H.264 Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of H.264 video to object storage in MP4 format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of H.264 encoding is supported by the DUT as indicated by Encoding capability which contains H264.

- Recording to MP4 format is supported by the DUT as indicated by SupportedTargetFormats capability which contains MP4.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with video only configured by following the procedure mentioned in Annex A.20 with the following input and output parameters:
   - in **H264** - required encoding
   - in **MP4** - required format type
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.

8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

8.6. If duration of *segment1* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* recording does not have content type equal to **video/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.12. If *segment2* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording does not contain **H.264** video data, FAIL the test, restore DUT settings and skip other steps.

8.14. For all fragments of *span1*:

8.14.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.5 OBJECT STORAGE RECORDING - VIDEO RECORDING (H.264, CMAF)

Test Case ID: RECORDING-2-2-5

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, H.264 Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of H.264 video to object storage in CMAF format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of H.264 encoding is supported by the DUT as indicated by Encoding capability which contains H264.
- Recording to CMAF format is supported by the DUT as indicated by SupportedTargetFormats capability which contains CMAF.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with video only configured by following the procedure mentioned in Annex A.20 with the following input and output parameters:
   - in **H264** - required encoding
   - in **CMAF** - required format type
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

8.6. If duration of *segment1* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* does not have **m4v** as extension for the span type, FAIL the test, restore DUT settings and skip other steps.

8.12. If *span1* recording does not have content type equal to **video/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording is not in **CMAF** format, FAIL the test, restore DUT settings and skip other steps.

8.14. If *segment2* recording does not contain **H.264** video data, FAIL the test, restore DUT settings and skip other steps.

8.15. For all fragments of *span1*:

8.15.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.6 OBJECT STORAGE RECORDING - AUDIO RECORDING (OPUS, MP4)

Test Case ID: RECORDING-2-2-6

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, OPUS Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of OPUS audio to object storage in MP4 format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of OPUS encoding is supported by the DUT as indicated by Encoding capability which contains opus.
- Recording to MP4 format is supported by the DUT as indicated by SupportedTargetFormats capability which contains MP4.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with audio only configured by following the procedure mentioned in Annex A.23 with the following input and output parameters:
   - in **opus** - required encoding
   - in **MP4** - required format type
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
   8.6. If duration of *segment1* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.

8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* recording does not have content type equal to **video/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.12. If *segment2* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording does not contain **OPUS** audio data, FAIL the test, restore DUT settings and skip other steps.

8.14. For all fragments of *span1*:

8.14.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.7 OBJECT STORAGE RECORDING - AUDIO RECORDING (AAC, MP4)

Test Case ID: RECORDING-2-2-7

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, AAC Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of AAC audio to object storage in MP4 format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of AAC encoding is supported by the DUT as indicated by Encoding capability which contains AAC.
- Recording to MP4 format is supported by the DUT as indicated by SupportedTargetFormats capability which contains MP4.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client prepares recording with recording job for Media2 profile with audio only configured by following the procedure mentioned in Annex A.23 with the following input and output parameters:
   - in **AAC** - required encoding
   - in **MP4** - required format type
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   - 8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   - 8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   - 8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   - 8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   - 8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
   - 8.6. If duration of *segment1* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.
   - 8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* recording does not have content type equal to **video/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.12. If *segment2* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording does not contain **AAC** audio data, FAIL the test, restore DUT settings and skip other steps.

8.14. For all fragments of *span1*:

8.14.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.8 OBJECT STORAGE RECORDING - AUDIO RECORDING (OPUS, CMAF)

Test Case ID: RECORDING-2-2-8

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, OPUS Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of OPUS audio to object storage in CMAF format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by the SupportedTargetFormats capability.
- Object storage is supported as indicated by the System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of OPUS encoding is supported by the DUT as indicated by the Encoding capability which contains opus.
- Recording to CMAF format is supported by the DUT as indicated by the SupportedTargetFormats capability which contains CMAF.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with audio only configured by following the procedure mentioned in Annex A.23 with the following input and output parameters:
   - in **opus** - required encoding

- in **CMAF** - required format type
- out *recordingJobToken1* - recording job for test
- out *recordingSpanDuration* - recording span duration
- out *recordingSegmentDuration* - recording segment duration

4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
   8.6. If duration of *segment1* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.
   8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.
   8.9. If duration of *segment2* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* does not have **m4a** as extension for the span type, FAIL the test, restore DUT settings and skip other steps.

8.12. If *span1* recording does not have content type equal to **audio/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording is not in **CMAF** format, FAIL the test, restore DUT settings and skip other steps.

8.14. If *segment2* recording does not contain **OPUS** audio data, FAIL the test, restore DUT settings and skip other steps.

8.15. For all fragments of *span1*:

8.15.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.9 OBJECT STORAGE RECORDING - AUDIO RECORDING (AAC, CMAF)

Test Case ID: RECORDING-2-2-9

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, AAC Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of AAC audio to object storage in CMAF format type.
- To verify that the DUT will close existing and open a new span after *SpanDuration* is reached.
- To verify that the DUT will close existing and open a new segment after *SegmentDuration* is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of AAC encoding is supported by the DUT as indicated by Encoding capability which contains AAC.
- Recording to CMAF format is supported by the DUT as indicated by SupportedTargetFormats capability which contains CMAF.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with only audio configured by following the procedure mentioned in Annex A.23 with the following input and output parameters:
   - in **AAC** - required encoding

- in **CMAF** - required format type
- out *recordingJobToken1* - recording job for test
- out *recordingSpanDuration* - recording span duration
- out *recordingSegmentDuration* - recording segment duration

4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with the following parameters:
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
   8.6. If duration of *segment1* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.
   8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.
   8.9. If duration of *segment2* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* does not have **m4a** as the extension for the span type, FAIL the test, restore DUT settings and skip other steps.

8.12. If *span1* recording does not have content type equal to **audio/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording is not in **CMAF** format, FAIL the test, restore DUT settings and skip other steps.

8.14. If *segment2* recording does not contain **AAC** audio data, FAIL the test, restore DUT settings and skip other steps.

8.15. For all fragments of *span1*:

8.15.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.10 OBJECT STORAGE RECORDING - VIDEO AND AUDIO RECORDING (H.264 + OPUS/AAC, MP4)

Test Case ID: RECORDING-2-2-10

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, Video and Audio Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of audio and video to object storage in MP4 format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there is no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of OPUS or AAC encoding is supported by the DUT as indicated by Encoding capability which contains opus or AAC.
- Recording of H.264 encoding is supported by the DUT as indicated by Encoding capability which contains opus or H264.
- Recording to MP4 format is supported by the DUT as indicated by SupportedTargetFormats capability which contains MP4.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with video and audio only configured by following the procedure mentioned in Annex A.25 with the following input and output parameters:

- in **[H264]** - applicable video encoding list
- in **[opus, AAC]** - applicable audio encoding list
- in **MP4** - required format type
- out *recordingJobToken1* - recording job for test
- out *recordingSpanDuration* - recording span duration
- out *recordingSegmentDuration* - recording segment duration
- out *audioEncoding* - audio recording used for recording configuration

4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   8.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
   8.2. If ONVIF Client was not able to find *span1*, FAIL the test, restore DUT settings and skip other steps.
   8.3. If duration of *span1* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   8.4. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.5. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
   8.6. If duration of *segment1* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.
   8.7. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is a gap between *segment1* and *segment2*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *span1* recording does not have content type equal to **video/mp4**, FAIL the test, restore DUT settings and skip other steps.

8.12. If *segment2* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.13. If *segment2* recording does not contain **H.264** video data and *audioEncoding* audio data, FAIL the test, restore DUT settings and skip other steps.

8.14. For all fragments of *span1*:

8.14.1 If fragment is not processable or not presentable independently with the information of a MovieBox, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.11 OBJECT STORAGE RECORDING - VIDEO AND AUDIO RECORDING (H.264 + OPUS/AAC, CMAF)

Test Case ID: RECORDING-2-2-11

Specification Coverage: Span end objects, Segment objects, Segments, Spans (ONVIF Recording Control Service Specification)

Feature Under Test: Recording to Object Storage, Video and Audio Recording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording of audio and video to object storage in CMAF format type.
- To verify that the DUT will close existing and open a new span after SpanDuration is reached.
- To verify that the DUT will close existing and open a new segment after SegmentDuration is reached.
- To verify that there are no time gap between segments.
- To verify that all fragments of a span are independently processable and presentable with the information of a MovieBox in any of the segments of the span.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of OPUS or AAC encoding is supported by the DUT as indicated by Encoding capability which contains opus or AAC.
- Recording of H.264 encoding is supported by the DUT as indicated by Encoding capability which contains opus or H264.
- Recording to CMAF format is supported by the DUT as indicated by SupportedTargetFormats capability which contains CMAF.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client prepares recording with recording job for Media2 profile with video and audio only configured by following the procedure mentioned in Annex A.25 with the following input and output parameters:
   - in **[H264]** - applicable video encoding list
   - in **[opus, AAC]** - applicable audio encoding list
   - in **CMAF** - required format type
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
   - out *audioEncoding* - audio recording used for recording configuration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSpanDuration* * 2 + 20%.
8. ONVIF Client verifies recorded data:
   - 8.1. ONVIF Client finds the first span *span1Video* content type equal to **video/mp4** by checking segments object keys dates.
   - 8.2. If ONVIF Client was not able to find *span1Video*, FAIL the test, restore DUT settings and skip other steps.
   - 8.3. If duration of *span1Video* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.
   - 8.4. At *span1Video* ONVIF Client finds segment *segment1Video* with **counter** = 0 by checking segments object keys dates.
   - 8.5. If ONVIF Client was not able to find *segment1Video*, FAIL the test, restore DUT settings and skip other steps.

8.6. If duration of *segment1Video* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.7. At *span1Video* ONVIF Client finds segment *segment2Video* with **counter** = 1 by checking segments object keys dates.

8.8. If ONVIF Client was not able to find *segment2Video*, FAIL the test, restore DUT settings and skip other steps.

8.9. If duration of *segment2Video* exceeds *recordingSegmentDuration* plus one GOP size, FAIL the test, restore DUT settings and skip other steps.

8.10. If there is gap between *segment1Video* and *segment2Video*, FAIL the test, restore DUT settings and skip other steps.

8.11. If *segment2Video* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.12. If *segment2Video* recording does not contains **H.264** video data, FAIL the test, restore DUT settings and skip other steps.

8.13. ONVIF Client finds the first span *span1Audio* content type equal to **audio/mp4** by checking segments object keys dates.

8.14. If ONVIF Client was not able to find *span1Audio*, FAIL the test, restore DUT settings and skip other steps.

8.15. If duration of *span1Audio* exceeds *recordingSpanDuration*, FAIL the test, restore DUT settings and skip other steps.

8.16. At *span1Audio* ONVIF Client finds segment *segment1Audio* with **counter** = 0 by checking segments object keys dates.

8.17. If ONVIF Client was not able to find *segment1Audio*, FAIL the test, restore DUT settings and skip other steps.

8.18. If duration of *segment1Audio* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.

8.19. At *span1Audio* ONVIF Client finds segment *segment2Audio* with **counter** = 1 by checking segments object keys dates.

8.20. If ONVIF Client was not able to find *segment2Audio*, FAIL the test, restore DUT settings and skip other steps.

8.21. If duration of *segment2Audio* exceeds *recordingSegmentDuration*, FAIL the test, restore DUT settings and skip other steps.

8.22. If there is gap between *segment1Audio* and *segment2Audio*, FAIL the test, restore DUT settings and skip other steps.

8.23. If *segment2Audio* recording is not in **MP4** format, FAIL the test, restore DUT settings and skip other steps.

8.24. If *segment2Audio* recording does not contains *audioEncoding* audio data, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.12 OBJECT STORAGE RECORDING - DIFFERENT STORAGES (ObjectStorageS3)

Test Case ID: RECORDING-2-2-12

Specification Coverage: External targets (ONVIF Recording Control Service Specification), Storage Configuration (ONVIF Core Specification)

Feature Under Test: Recording to Object Storage, ObjectStorageS3

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording to object ObjectStorageS3 storage.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.

- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- ObjectStorageS3 object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **ObjectStorageS3** - object storage type
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSegmentDuration* + 20%.
8. ONVIF Client verifies that data was recorded:
   8.1. ONVIF Client finds the latest span *span1* by checking segments object keys dates.
   8.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.

8.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

9. ONVIF Client restores DUT configuration if required.

10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.13 OBJECT STORAGE RECORDING - DIFFERENT STORAGES (ObjectStorageAzure)

Test Case ID: RECORDING-2-2-13

Specification Coverage: External targets (ONVIF Recording Control Service Specification), Storage Configuration (ONVIF Core Specification)

Feature Under Test: Recording to Object Storage, ObjectStorageAzure

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording to object ObjectStorageAzure storage.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- ObjectStorageAzure object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageAzure.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **ObjectStorageAzure** - object storage type
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. Wait for *recordingSegmentDuration* + 20%.
8. ONVIF Client verifies that data was recorded:
   8.1. ONVIF Client finds the latest span *span1* by checking segments object keys dates.
   8.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
   8.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
9. ONVIF Client restores DUT configuration if required.
10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message(s).

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

5.2.2.1.14 OBJECT STORAGE RECORDING - SYMMETRIC ENCRYPTION (CENC)

Test Case ID: RECORDING-2-2-14

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: Symmetric Encryption, CENC Encryption Mode

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with symmetric encryption.
- To verify recording with encryption with CENC encryption mode.
- To verify ProtectionSystemSpecificHeaderBox PSSH box at recording segments.
- To verify encryption change during recording.
- To verify that key is skipped at GetRecordingConfiguration.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- CENC encryption mode is supported by the DUT as indicated by SupportedEncryptionModes.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID := [kid1 UUID string]
   - *encryption1*.Key := [key1 random 16-byte value]
   - *encryption1*.Track is skipped
   - *encryption1*.AsymmetricEncryption is skipped
4. Set:
   - *encryption2*.Mode := **CENC**
   - *encryption2*.KID := [kid2 UUID string]
   - *encryption2*.Key := [key2 random 16-byte value]
   - *encryption2*.Track is skipped
   - *encryption2*.AsymmetricEncryption is skipped
5. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in *encryption1* - recording encryption
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
6. ONVIF Client deletes all data from object storage.
7. ONVIF Client invokes **SetRecordingJobMode** request with parameters

- JobToken := *recordingJobToken1*
- Mode := **Active**

8. The DUT responds with **SetRecordingJobModeResponse** message.
9. Wait for *recordingSegmentDuration* * 2 + 20%.
10. ONVIF Client verifies that encrypted data was recorded:
    10.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
    10.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
    10.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
    10.4. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:
        - in *segment1* - encrypted segment
        - in **1** - expected KID_count
        - in [*encryption1*.KID] - expected KID list
        - in [(*encryption1*.KID, *encryption1*.Key)] - key list for decryption
    10.5. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.
    10.6. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.
    10.7. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:
        - in *segment2* - encrypted segment
        - in **1** - expected KID_count
        - in [*encryption1*.KID] - expected KID list
        - in [(*encryption1*.KID, *encryption1*.Key)] - key list for decryption
11. ONVIF Client invokes **GetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken1*

12. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration1*
13. If *recordingConfiguration1*.Target.Encryption is not defined, FAIL the test, restore DUT settings and skip other steps.
14. If *recordingConfiguration1*.Target.Encryption.Key is not skipped, FAIL the test, restore DUT settings and skip other steps.
15. Set *recordingConfiguration1*.Target.Encryption := *encryption2*.
16. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *recordingConfiguration1*
17. The DUT responds with **SetRecordingConfigurationResponse** message.
18. Wait for *recordingSegmentDuration* + 20%.
19. ONVIF Client verifies that data was recorded with updated encryption:
    19.1. ONVIF Client finds the last span *span2* by checking segments object keys dates.
    19.2. At *span2* ONVIF Client finds segment *segment3* with maximum **counter** by checking segments object keys dates.
    19.3. If ONVIF Client was not able to find *segment3*, FAIL the test, restore DUT settings and skip other steps.
    19.4. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:
        - in *segment3* - encrypted segment
        - in **1** - expected KID_count
        - in [*encryption2*.KID] - expected KID list
        - in [(*encryption2*.KID, *encryption2*.Key)] - key list for decryption
20. ONVIF Client restores DUT configuration if required.
21. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.

5.2.2.1.15 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (RSA)

Test Case ID: RECORDING-2-2-15

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: RSA Asymmetric Encryption

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with asymmetric encryption with RSA key pair.
- To verify AsymmetricKeySystemHeaderBox PSSH box at recording segments.
- To verify encryption with several certificates defined.
- To verify that a DUT supports OAEP-PSS Digest Algorithms that required by ONVIF Security Baseline.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- Asymmetric encryption is supported by the DUT as indicated by AsymmetricEncryptionSupported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
4. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.15 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in **RSA** - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAprivateKey* - private key for the CA certificate
   - out *CApublicKey* - public key for the CA certificate
5. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec1,C=US" - certificate subject
   - in **RSA** - key pair algorithm
   - in *CAcert* - CA certificate
   - in *CAprivateKey* - private key for the CA certificate
   - in *CApublicKey* - public key for the CA certificate
   - out *cert1* - certificate
   - out *privateKey1* - private key for the certificate

- out *publicKey1* - public key for the certificate

6. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec2,C=US" - certificate subject
   - in **RSA** - key pair algorithm
   - in *CAcert* - CA certificate
   - in *CAprivateKey* - private key for the CA certificate
   - in *CApublicKey* - public key for the CA certificate
   - out *cert2* - certificate
   - out *privateKey2* - private key for the certificate
   - out *publicKey2* - public key for the certificate
7. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:
   - in *cert1* - certificate
   - out *certID1* - certificate ID
   - out *keyID1* - key pair ID
8. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:
   - in *cert2* - certificate
   - out *certID2* - certificate ID
   - out *keyID2* - key pair ID
9. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID is skipped
   - *encryption1*.Key is skipped

- *encryption1*.Track is skipped
- *encryption1*.AsymmetricEncryption.CertificateID[0] = *certID1*

10. Set:
    - *encryption2*.Mode := **CENC**
    - *encryption2*.KID is skipped
    - *encryption2*.Key is skipped
    - *encryption2*.Track is skipped
    - *encryption2*.AsymmetricEncryption.CertificateID[0] = *certID2*
    - *encryption1*.AsymmetricEncryption.CertificateID[1] = *certID1*
11. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - in *encryption1* - recording encryption
    - out *recordingToken1* - recording for test
    - out *recordingConfiguration1* - recording configuration for test
    - out *recordingJobToken1* - recording job for test
    - out *recordingJobConfiguration1* - recording job configuration for test
    - out *recordingSegmentDuration* - recording segment duration
12. ONVIF Client deletes all data from object storage.
13. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Active**
14. The DUT responds with **SetRecordingJobModeResponse** message.
15. Wait for *recordingSegmentDuration* * 2 + 20%.
16. ONVIF Client verifies that encrypted data was recorded:
    16.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.

16.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.

16.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

16.4. ONVIF Client checks segment with asymmetric encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

- in *segment1* - encrypted segment
- in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
- out *KID1* - KID

16.5. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

16.6. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

16.7. ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

- in *segment2* - encrypted segment
- in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
- in *KID1* - expected KID

17. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
18. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration1*
19. Set *recordingConfiguration1*.Target.Encryption := *encryption2*.
20. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *recordingConfiguration1*
21. The DUT responds with **SetRecordingConfigurationResponse** message.

22. Wait for *recordingSegmentDuration* + 20%.
23. ONVIF Client verifies that data was recorded with updated encryption:
    23.1. ONVIF Client finds the last span *span2* by checking segments object keys dates.
    23.2. At *span2* ONVIF Client finds segment *segment3* with maximum **counter** by checking segments object keys dates.
    23.3. If ONVIF Client was not able to find *segment3*, FAIL the test, restore DUT settings and skip other steps.
    23.4. ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:
    - in *segment3* - encrypted segment
    - in [(*cert2*, *privateKey2*), (*cert1*, *privateKey1*)] - certificates list with private keys for decryption
24. ONVIF Client restores DUT configuration if required.
25. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.

5.2.2.1.16 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (ECC)

Test Case ID: RECORDING-2-2-16

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: ECC Asymmetric Encryption

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with asymmetric encryption with ECC key pair.
- To verify AsymmetricKeySystemHeaderBox PSSH box at recording segments.
- To verify encryption with several certificates defined.
- To verify that a DUT supports HPKE KEM Identifiers, HPKE KDF Identifiers, HPKE AEAD Identifiers set that required by ONVIF Security Baseline.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- Asymmetric encryption is supported by the DUT as indicated by AsymmetricEncryptionSupported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
4. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.15 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in **ECC** - key pair algorithm
   - out *CAcert* - CA certificate

- out *CAprivateKey* - private key for the CA certificate
- out *CApublicKey* - public key for the CA certificate

5. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec1,C=US" - certificate subject
   - in **ECC** - key pair algorithm
   - in *CAcert* - CA certificate
   - in *CAprivateKey* - private key for the CA certificate
   - in *CApublicKey* - public key for the CA certificate
   - out *cert1* - certificate
   - out *privateKey1* - private key for the certificate
   - out *publicKey1* - public key for the certificate
6. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec2,C=US" - certificate subject
   - in **ECC** - key pair algorithm
   - in *CAcert* - CA certificate
   - in *CAprivateKey* - private key for the CA certificate
   - in *CApublicKey* - public key for the CA certificate
   - out *cert2* - certificate
   - out *privateKey2* - private key for the certificate
   - out *publicKey2* - public key for the certificate
7. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:

- in *cert1* - certificate
- out *certID1* - certificate ID
- out *keyID1* - key pair ID

8. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:
   - in *cert2* - certificate
   - out *certID2* - certificate ID
   - out *keyID2* - key pair ID
9. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID is skipped
   - *encryption1*.Key is skipped
   - *encryption1*.Track is skipped
   - *encryption1*.AsymmetricEncryption.CertificateID[0] = *certID1*
10. Set:
   - *encryption2*.Mode := **CENC**
   - *encryption2*.KID is skipped
   - *encryption2*.Key is skipped
   - *encryption2*.Track is skipped
   - *encryption2*.AsymmetricEncryption.CertificateID[0] = *certID2*
   - *encryption1*.AsymmetricEncryption.CertificateID[1] = *certID1*
11. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in *encryption1* - recording encryption
   - out *recordingToken1* - recording for test

- out *recordingConfiguration1* - recording configuration for test
- out *recordingJobToken1* - recording job for test
- out *recordingJobConfiguration1* - recording job configuration for test
- out *recordingSegmentDuration* - recording segment duration

12. ONVIF Client deletes all data from object storage.
13. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Active**
14. The DUT responds with **SetRecordingJobModeResponse** message.
15. Wait for *recordingSegmentDuration* * 2 + 20%.
16. ONVIF Client verifies that encrypted data was recorded:
    16.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.
    16.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.
    16.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.
    16.4. ONVIF Client checks segment with asymmetric encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:
        - in *segment1* - encrypted segment
        - in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
        - out *KID1* - KID
    16.5. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.
    16.6. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.
    16.7. ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

- in *segment2* - encrypted segment
- in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
- in *KID1* - expected KID

17. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
18. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration1*
19. Set *recordingConfiguration1*.Target.Encryption := *encryption2*.
20. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *recordingConfiguration1*
21. The DUT responds with **SetRecordingConfigurationResponse** message.
22. Wait for *recordingSegmentDuration* + 20%.
23. ONVIF Client verifies that data was recorded with updated encryption:
    23.1. ONVIF Client finds the last span *span2* by checking segments object keys dates.
    23.2. At *span2* ONVIF Client finds segment *segment3* with maximum **counter** by checking segments object keys dates.
    23.3. If ONVIF Client was not able to find *segment3*, FAIL the test, restore DUT settings and skip other steps.
    23.4. ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:
        - in *segment3* - encrypted segment
        - in [(*cert2*, *privateKey2*), (*cert1*, *privateKey1*)] - certificates list with private keys for decryption
24. ONVIF Client restores DUT configuration if required.
25. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.

5.2.2.1.17 OBJECT STORAGE RECORDING - ASYMMETRIC ENCRYPTION (key rotation)

Test Case ID: RECORDING-2-2-17

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: RSA Asymmetric Encryption

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with asymmetric encryption with key rotation.
- To verify AsymmetricKeySystemHeaderBox PSSH box at recording segments.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- Asymmetric encryption is supported by the DUT as indicated by AsymmetricEncryptionSupported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
4. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *capSecurity* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
5. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.15 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAprivateKey* - private key for the CA certificate
   - out *CApublicKey* - public key for the CA certificate
6. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec1,C=US" - certificate subject
   - in *keyAlgorithm* - key pair algorithm
   - in *CAcert* - CA certificate
   - in *CAprivateKey* - private key for the CA certificate
   - in *CApublicKey* - public key for the CA certificate
   - out *cert1* - certificate
   - out *privateKey1* - private key for the certificate
   - out *publicKey1* - public key for the certificate

7. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:
   - in *cert1* - certificate
   - out *certID1* - certificate ID
   - out *keyID1* - key pair ID
8. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID is skipped
   - *encryption1*.Key is skipped
   - *encryption1*.Track is skipped
   - *encryption1*.AsymmetricEncryption.CertificateID[0] = *certID1*
   - *encryption1*.AsymmetricEncryption.KeyRotationDuration = **PT60S**
9. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in *encryption1* - recording encryption
   - out *recordingToken1* - recording for test
   - out *recordingConfiguration1* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
10. ONVIF Client deletes all data from object storage.
11. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Active**
12. The DUT responds with **SetRecordingJobModeResponse** message.

13. Wait for 60 seconds + *recordingSegmentDuration*.

14. ONVIF Client verifies that encrypted data was recorded:

    14.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.

    14.2. At *span1* ONVIF Client finds segment *segment1* with **counter** = 0 by checking segments object keys dates.

    14.3. If ONVIF Client was not able to find *segment1*, FAIL the test, restore DUT settings and skip other steps.

    14.4. ONVIF Client checks segment with asymmetric encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

    - in *segment1* - encrypted segment
    - in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
    - out *KID1* - KID
    - out *key1* - symmetric key

    14.5. At *span1* ONVIF Client finds segment *segment2* with **counter** = 1 by checking segments object keys dates.

    14.6. If ONVIF Client was not able to find *segment2*, FAIL the test, restore DUT settings and skip other steps.

    14.7. ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

    - in *segment2* - encrypted segment
    - in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
    - in *KID1* - expected KID

    14.8. ONVIF Client finds the last span *span2* by checking segments object keys dates.

    14.9. At *span2* ONVIF Client finds last segment *segment3* with maximum **counter** by checking segments object keys dates.

    14.10 If ONVIF Client was not able to find *segment3*, FAIL the test, restore DUT settings and skip other steps.

14.11 ONVIF Client checks segment encryption by the following the procedure mentioned in Annex A.30 with the following input and output parameters:

- in *segment3* - encrypted segment
- in [(*cert1*, *privateKey1*)] - certificates list with private keys for decryption
- out *KID2* - KID
- out *key2* - symmetric key

14.12 If *KID2* is equal to *KID1*, FAIL the test, restore DUT settings and skip other steps.

14.13 If *key2* is equal to *key1*, FAIL the test, restore DUT settings and skip other steps.

15. ONVIF Client restores DUT configuration if required.
16. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.

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

5.2.2.1.18 OBJECT STORAGE RECORDING - SYMMETRIC ENCRYPTION (CENC, several tracks encryption, H.264 + OPUS/AAC, CMAF)

Test Case ID: RECORDING-2-2-18

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: Symmetric Encryption, CENC Encryption Mode, Different Track Encryption

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with encryption for several tracks.
- To verify recording with different encryption for different tracks.
- To verify ProtectionSystemSpecificHeaderBox PSSH box at recording segments.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- CENC encryption mode is supported by the DUT as indicated by SupportedEncryptionModes.
- Recording of OPUS or AAC encoding is supported by the DUT as indicated by Encoding capability which contains opus or AAC.
- Recording of H.264 encoding is supported by the DUT as indicated by Encoding capability which contains opus or H264.
- Recording to CMAF format is supported by the DUT as indicated by SupportedTargetFormats capability which contains CMAF.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
4. If *cap*.EncryptionEntryLimit < 2, PASS the test and skip other steps.
5. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID := [kid1 UUID string]
   - *encryption1*.Key := [key1 random 16-byte value]
   - *encryption1*.Track is skipped

- *encryption1*.AsymmetricEncryption is skipped

6. Set:
   - *encryption2*.Mode := **CENC**
   - *encryption2*.KID := [kid2 UUID string]
   - *encryption2*.Key := [key2 random 16-byte value]
   - *encryption2*.Track is skipped
   - *encryption2*.AsymmetricEncryption is skipped
7. ONVIF Client prepares recording with recording job for Media2 profile with video and audio only configured by following the procedure mentioned in Annex A.25 with the following input and output parameters:
   - in *encryption1* - recording encryption
   - in **[H264]** - applicable video encoding list
   - in **[opus, AAC]** - applicable audio encoding list
   - in **CMAF** - required format type
   - out *recordingToken1* - recording token
   - out *recordingJobToken1* - recording job for test
   - out *recordingSpanDuration* - recording span duration
   - out *recordingSegmentDuration* - recording segment duration
   - out *audioEncoding* - audio recording used for recording configuration
8. ONVIF Client deletes all data from object storage.
9. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
10. The DUT responds with **SetRecordingJobModeResponse** message.
11. Wait for *recordingSegmentDuration* + 20%.
12. ONVIF Client verifies that encrypted data was recorded:

12.1. ONVIF Client finds the first span *span1* by checking segments object keys dates.

12.2. At *span1* ONVIF Client finds segment *segmentVideo1* with minimal **counter** and video data by checking segments object keys dates.

12.3. If ONVIF Client was not able to find *segmentVideo1*, FAIL the test, restore DUT settings and skip other steps.

12.4. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:

- in *segmentVideo1* - encrypted segment
- in **1** - expected KID_count
- in [*encryption1*.KID] - expected KID list
- in [(*encryption1*.KID, *encryption1*.Key)] - key list for decryption

12.5. At *span1* ONVIF Client finds segment *segmentAudio1* with minimal **counter** and audio data by checking segments object keys dates.

12.6. If ONVIF Client was not able to find *segmentAudio1*, FAIL the test, restore DUT settings and skip other steps.

12.7. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:

- in *segmentAudio1* - encrypted segment
- in **1** - expected KID_count
- in [*encryption1*.KID] - expected KID list
- in [(*encryption1*.KID, *encryption1*.Key)] - key list for decryption

13. ONVIF Client invokes **GetRecordings** request.

14. The DUT responds with **GetRecordingsResponse** message with the following parameters:

- RecordingItem list =: *recordingsList1*

15. Set *encryption1*.Track := [TrackToken of the track with TrackType = Video from RecordingItem with RecordingToken = *recordingToken1*]

16. Set *encryption2*.Track := [TrackToken of the track with TrackType = Audio from RecordingItem with RecordingToken = *recordingToken1*]

17. If *recordingConfiguration1*.Target.Encryption.Key is not skipped, FAIL the test, restore DUT settings and skip other steps.
18. Set *recordingConfiguration1*.Target.Encryption[0] := *encryption1*.
19. Set *recordingConfiguration1*.Target.Encryption[1] := *encryption2*.
20. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *recordingConfiguration1*
21. The DUT responds with **SetRecordingConfigurationResponse** message.
22. Wait for *recordingSegmentDuration**2 + 20%.
23. ONVIF Client verifies that data was recorded with updated encryption:
    23.1. ONVIF Client finds the last span *span2* by checking segments object keys dates.
    23.2. At *span2* ONVIF Client finds segment *segmentVideo2* with minimal **counter** and video data by checking segments object keys dates.
    23.3. If ONVIF Client was not able to find *segmentVideo2*, FAIL the test, restore DUT settings and skip other steps.
    23.4. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:
    - in *segmentVideo2* - encrypted segment
    - in **1** - expected KID_count
    - in [*encryption1*.KID] - expected KID list
    - in [(*encryption1*.KID, *encryption1*.Key)] - key list for decryption

    23.5. At *span2* ONVIF Client finds segment *segmentAudio2* with minimal **counter** and audio data by checking segments object keys dates.
    23.6. If ONVIF Client was not able to find *segmentAudio2*, FAIL the test, restore DUT settings and skip other steps.
    23.7. ONVIF Client checks segment with symmetric encryption by the following the procedure mentioned in Annex A.29 with the following input and output parameters:
    - in *segmentAudio2* - encrypted segment

- in **1** - expected KID_count
- in [*encryption2*.KID] - expected KID list
- in [(*encryption2*.KID, *encryption2*.Key)] - key list for decryption

24. ONVIF Client restores DUT configuration if required.
25. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingsResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.

5.2.2.1.19 OBJECT STORAGE RECORDING - ENCRYPTION (ambiguous configuration)

Test Case ID: RECORDING-2-2-19

Specification Coverage: Encryption (ONVIF Recording Control Service Specification)

Feature Under Test: Ambiguous Configuration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify recording with asymmetric encryption with key rotation.
- To verify AsymmetricKeySystemHeaderBox PSSH box at recording segments.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.

- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Encryption is supported by the DUT as indicated by EncryptionEntryLimit > 0.
- Asymmetric encryption is supported by the DUT as indicated by AsymmetricEncryptionSupported.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
4. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *capSecurity* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
5. ONVIF Client creates an CA certificate by following the procedure mentioned in Annex A.15 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *CAcert* - CA certificate
   - out *CAprivateKey* - private key for the CA certificate
   - out *CApublicKey* - public key for the CA certificate
6. ONVIF Client creates an certificate signed by CA certificate by following the procedure mentioned in Annex A.31 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in "CN=ONVIF TT Rec1,C=US" - certificate subject
   - in *keyAlgorithm* - key pair algorithm

- in *CAcert* - CA certificate
- in *CAprivateKey* - private key for the CA certificate
- in *CApublicKey* - public key for the CA certificate
- out *cert1* - certificate
- out *privateKey1* - private key for the certificate
- out *publicKey1* - public key for the certificate

7. ONVIF Client uploads a certificate by the following the procedure mentioned in Annex A.32 with the following input and output parameters:
   - in *cert1* - certificate
   - out *certID1* - certificate ID
   - out *keyID1* - key pair ID
8. Set:
   - *encryption1*.Mode := **CENC**
   - *encryption1*.KID := [kid1 UUID string]
   - *encryption1*.Key := [key1 random 16-byte value]
   - *encryption1*.Track is skipped
   - *encryption1*.AsymmetricEncryption.CertificateID[0] = *certID1*
   - *encryption1*.AsymmetricEncryption.KeyRotationDuration = **PT60S**
9. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
10. ONVIF Client invokes **GetRecordings** request.
11. The DUT responds with **GetRecordingsResponse** message with parameters
    - RecordingItem list =: *recordingsList1*
12. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:

- in *objectStorageType* - object storage type
- out *externalStorage1* - external storage configuration identifier

13. Set *recordingSegmentDuration* := **PT20S**.
14. Set:
    - *target1*.Storage := *externalStorage1*
    - *target1*.Format := *cap*.SupportedTargetFormats[0]
    - *target1*.SegmentDuration := *recordingSegmentDuration*
    - *target1*.Prefix is skipped
    - *target1*.Postfix is skipped
    - *target1*.SpanDuration is skipped
    - *target1*.Encryption := *encryption1*
    - *target1*.SegmentDurationOverride is skipped
15. If *cap*.DynamicRecording = true:
    15.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 14.5.
    15.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:
        - RecordingToken := *recordingsList1*[0].RecordingToken
    15.3. The DUT responds with **DeleteRecordingResponse** message.
    15.4. Set:
        - *recordingConfiguration1*.Source.Name := "Source1"
        - *recordingConfiguration1*.Source.SourceId := [valid URI]
        - *recordingConfiguration1*.Source.Address := [valid URI]
        - *recordingConfiguration1*.Source.Location := "Source Location"
        - *recordingConfiguration1*.Source.Description := "Source Description"
        - *recordingConfiguration1*.Content := "Content"

- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

15.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

15.6. The DUT returns **env:Sender\ter:InvalidArgVal\ter:AmbiguousConfiguration** SOAP 1.2 fault.

16. If *cap*.DynamicRecording = false:

16.1. If *recordings1* is empty, FAIL the test, restore DUT settings and skip other steps.

16.2. Set:

- *recordingConfiguration*.Target := *target1*
- *recordingToken* := *recordings1*[0].RecordingToken
- *recordingConfiguration* := *recordings1*[0].Configuration

16.3. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

16.4. The DUT returns **env:Sender\ter:InvalidArgVal\ter:AmbiguousConfiguration** SOAP 1.2 fault.

17. ONVIF Client restores DUT configuration if required.

18. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingsResponse** message.
- The DUT did not send **DeleteRecordingResponse** message.

- The DUT did not send **env:Sender\ter:InvalidArgVal\ter:AmbiguousConfiguration** SOAP 1.2 fault.

5.2.2.1.20 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION

Test Case ID: RECORDING-2-2-20

Specification Coverage: OverrideSegmentDuration (ONVIF Recording Control Service Specification)

Feature Under Test: OverrideSegmentDuration, GetRecordingConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that new duration will be applied for new segment after OverrideSegmentDuration was invoked.
- To verify that segment duration set by OverrideSegmentDuration will be active during provided duration.
- To verify that initial segment duration will be active after provided override duration is expired.
- To verify that the DUT returns segment duration override with valid data at GetRecordingConfiguration when it is active.
- To verify that the DUT does not return segment duration override at GetRecordingConfiguration when it is not active.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Override segment duration is supported by the DUT as indicated by OverrideSegmentDuration.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **PT60S** - recording segment duration
   - out *recordingToken1* - recording for test
   - out *recordingConfigurationInitial* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
   - RecordingToken := *recordingToken1*
8. The DUT responds with **GetRecordingConfigurationResponse** with parameters
   - RecordingConfiguration =: *recordingConfiguration1*
9. If *recordingConfiguration1* contains SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
10. ONVIF Client invokes **OverrideSegmentDuration** request with parameters
    - TargetSegmentDuration := **PT20S**
    - Expiration := **PT120S**

- RecordingConfiguration := *recordingToken1*

11. The DUT responds with **OverrideSegmentDurationResponse** message.
12. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
13. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration2*
14. If *recordingConfiguration2*.Target does not contain SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
15. If *recordingConfiguration2*.Target.SegmentDurationOverride.Duration is not equal to **PT20S**, FAIL the test, restore DUT settings and skip other steps.
16. If *recordingConfiguration2*.Target.SegmentDurationOverride.Expiration more than **PT120S**, FAIL the test, restore DUT settings and skip other steps.
17. If *recordingConfiguration2*.Target.SegmentDurationOverride.Expiration less than **PT120S** - [time since **OverrideSegmentDurationResponse** request was received - 10%], FAIL the test, restore DUT settings and skip other steps.
18. Wait for 10 seconds.
19. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
20. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration3*
21. If *recordingConfiguration3*.Target does not contain SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
22. If *recordingConfiguration3*.Target.SegmentDurationOverride.Duration is not equal to **PT20S**, FAIL the test, restore DUT settings and skip other steps.
23. If *recordingConfiguration2*.Target.SegmentDurationOverride.Expiration - *recordingConfiguration3*.Target.SegmentDurationOverride.Expiration is less that 10 seconds, FAIL the test, restore DUT settings and skip other steps.
24. Wait for 74 seconds.

25. ONVIF Client verifies recorded data:
    25.1. ONVIF Client finds the first completed segment *segment1* started after **OverrideSegmentDurationResponse** message was received.
    25.2. If duration of *segment1* exceeds 20 seconds plus one GOP size, FAIL the test, restore DUT settings and skip other steps.
26. Wait for *recordingConfiguration3*.Target.SegmentDurationOverride.Expiration until segment duration override expire.
27. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
28. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration4*
29. If *recordingConfiguration4*.Target contains SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
30. Wait for 60 seconds +10%.
31. ONVIF Client verifies recorded data:
    31.1. ONVIF Client finds the first completed segment *segment2* started after expiration of segment duration override.
    31.2. If duration of *segment2* less than 60 seconds minus one GOP size, FAIL the test, restore DUT settings and skip other steps.
32. ONVIF Client restores DUT configuration if required.
33. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message(s).
- The DUT did not send **OverrideSegmentDurationResponse** message.

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

5.2.2.1.21 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (override current)

Test Case ID: RECORDING-2-2-21

Specification Coverage: OverrideSegmentDuration (ONVIF Recording Control Service Specification)

Feature Under Test: OverrideSegmentDuration, GetRecordingConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that new duration will be applied for new segment after OverrideSegmentDuration was invoked when other override segment duration is still active.
- To verify that segment duration set by OverrideSegmentDuration will be active during provided duration.
- To verify that initial segment duration will be active after provided override duration is expired.
- To verify that the DUT returns segment duration override with valid data at GetRecordingConfiguration when it is active.
- To verify that the DUT does not return segment duration override at GetRecordingConfiguration when it is not active.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Override segment duration is supported by the DUT as indicated by OverrideSegmentDuration.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **PT60S** - recording segment duration
   - out *recordingToken1* - recording for test
   - out *recordingConfigurationInitial* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. ONVIF Client invokes **OverrideSegmentDuration** request with parameters
   - TargetSegmentDuration := **PT40S**
   - Expiration := **PT240S**
   - RecordingConfiguration := *recordingToken1*
8. The DUT responds with **OverrideSegmentDurationResponse** message.
9. ONVIF Client invokes **OverrideSegmentDuration** request with parameters
   - TargetSegmentDuration := **PT20S**
   - Expiration := **PT120S**
   - RecordingConfiguration := *recordingToken1*
10. The DUT responds with **OverrideSegmentDurationResponse** message.

11. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
12. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration1*
13. If *recordingConfiguration1*.Target does not contain SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
14. If *recordingConfiguration1*.Target.SegmentDurationOverride.Duration is not equal to **PT20S**, FAIL the test, restore DUT settings and skip other steps.
15. If *recordingConfiguration1*.Target.SegmentDurationOverride.Expiration more than **PT120S**, FAIL the test, restore DUT settings and skip other steps.
16. If *recordingConfiguration1*.Target.SegmentDurationOverride.Expiration less than **PT120S** - [time since second **OverrideSegmentDurationResponse** request was received - 10%], FAIL the test, restore DUT settings and skip other steps.
17. Wait for 84 seconds.
18. ONVIF Client verifies recorded data:
    18.1. ONVIF Client finds the first completed segment *segment1* started after second **OverrideSegmentDurationResponse** message was received.
    18.2. If duration of *segment1* exceeds 20 seconds plus one GOP size, FAIL the test, restore DUT settings and skip other steps.
19. Wait for *recordingConfiguration1*.Target.SegmentDurationOverride.Expiration until segment duration override expire.
20. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
21. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration2*
22. If *recordingConfiguration2*.Target contains SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
23. Wait for 60 seconds +10% .
24. ONVIF Client verifies recorded data:

24.1. ONVIF Client finds the first completed segment *segment2* started after expiration of segment duration override.

24.2. If duration of *segment2* less than 60 seconds minus one GOP size, FAIL the test, restore DUT settings and skip other steps.

25. ONVIF Client restores DUT configuration if required.

26. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message(s).
- The DUT did not send **OverrideSegmentDurationResponse** message(s).

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

5.2.2.1.22 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (recording configuration update)

Test Case ID: RECORDING-2-2-22

Specification Coverage: OverrideSegmentDuration (ONVIF Recording Control Service Specification)

Feature Under Test: OverrideSegmentDuration, GetRecordingConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that override segment duration will be removed if recording configuration will be updated.
- To verify that initial segment duration will be active after provided override duration is removed.
- To verify that the DUT does not return segment duration override at GetRecordingConfiguration when it is not active.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Override segment duration is supported by the DUT as indicated by OverrideSegmentDuration.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **PT60S** - recording segment duration
   - out *recordingToken1* - recording for test
   - out *recordingConfigurationInitial* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.

7. ONVIF Client invokes **OverrideSegmentDuration** request with parameters
   - TargetSegmentDuration := **PT20S**
   - Expiration := **PT120S**
   - RecordingConfiguration := *recordingToken1*
8. The DUT responds with **OverrideSegmentDurationResponse** message.
9. Set *recordingConfigurationInitial*.Source.Name = "NewName".
10. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *recordingConfigurationInitial*
11. The DUT responds with **SetRecordingConfigurationResponse** message.
12. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
13. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration1*
14. If *recordingConfiguration1*.Target contains SegmentDurationOverride, FAIL the test, restore DUT settings and skip other steps.
15. Wait for 125 seconds.
16. ONVIF Client verifies recorded data:
    16.1. ONVIF Client finds the first completed segment *segment1* started after SetRecordingConfigurationResponse message was received.
    16.2. If duration of *segment1* less than 60 seconds minus one GOP size, FAIL the test, restore DUT settings and skip other steps.
17. ONVIF Client restores DUT configuration if required.
18. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message.
- The DUT did not send **OverrideSegmentDurationResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.

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

5.2.2.1.23 OBJECT STORAGE RECORDING - OVERRIDE SEGMENT DURATION (invalid expiration)

Test Case ID: RECORDING-2-2-23

Specification Coverage: OverrideSegmentDuration (ONVIF Recording Control Service Specification)

Feature Under Test: OverrideSegmentDuration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that the DUT will not allow to set duration above 1 hour.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Override segment duration is supported by the DUT as indicated by OverrideSegmentDuration.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
   - in **PT60S** - recording segment duration
   - out *recordingToken1* - recording for test
   - out *recordingConfigurationInitial* - recording configuration for test
   - out *recordingJobToken1* - recording job for test
   - out *recordingJobConfiguration1* - recording job configuration for test
   - out *recordingSegmentDuration* - recording segment duration
4. ONVIF Client deletes all data from object storage.
5. ONVIF Client invokes **SetRecordingJobMode** request with parameters
   - JobToken := *recordingJobToken1*
   - Mode := **Active**
6. The DUT responds with **SetRecordingJobModeResponse** message.
7. ONVIF Client invokes **OverrideSegmentDuration** request with parameters
   - TargetSegmentDuration := **PT20S**
   - Expiration := **PT61M**
   - RecordingConfiguration := *recordingToken1*
8. The DUT returns **env:Sender/ter:InvalidArgVal/ter:BadDuration** SOAP 1.2 fault.
9. ONVIF Client restores DUT configuration if required.
10. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.

- The DUT did not send SOAP 1.2 fault message.

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

5.2.2.1.24 OBJECT STORAGE RECORDING - RENEWAL

Test Case ID: RECORDING-2-2-24

Specification Coverage: Configuration Renewal (ONVIF Core Specification), Object storage recording format (ONVIF Recording Control Service Specification)

Feature Under Test: Storage Renewal

WSDL Reference: recording.wsdl, device.wsdl

Test Purpose:

- To verify that the DUT will apply configuration after renewal for existing recordings.
- To verify that if renewal request will be repeated after failure.
- To verify that if renewal request was failed the DUT continues to use existing configuration.
- To verify that the DUT immediately contact renewal endpoint after configuration was created with renewal endpoint.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Storage configuration renewal is supported by the DUT as indicated by StorageConfigurationRenewal.
- Test Operator specified Object Storage configuration data (*initialData1*).
- Test Operator specified Object Storage Renewal configuration data (*renewalData1*), which differs from Object Storage configuration data (*storageData1*). It is recommended to use different buckets and users data for proper testing.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. If Test Operator do not specify Object Storage Renewal configuration data (*renewalData1*), FAIL the test and skip other steps.
4. If Object Storage Renewal configuration data (*renewalData1*) specified by Test Operator is matches Object Storage configuration data (*initialData1*) of the same type, FAIL the test and skip other steps.
5. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *cap* - Security Configuration Service Capabilities
6. ONVIF Client configures an authorization server on Device and starts authorization server by following the procedure mentioned in Annex A.33 with the following input and output parameters
   - in *cap* - Security Configuration Service Capabilities
   - out *authServerToken1* - authorization server token
   - out *scope1* - authorization scope
   - out *authServerMetadataEndpoint1* - authentication server metadata endpoint
7. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.36 with the following input and output parameters
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in "C=US,O=ONVIF,CN=ONVIF TT RenewalEndpoint 1" - CA certificate subject
   - in "Test CertPathValidationPolicy RenewalEndpoint Alias" - certification path validation policy alias
   - out *certPathValidationPolicyIDRenewalEndpoint* - certification path validation policy identifier
   - out *certIDRenewalEndpoint* - certificate identifier
   - out *keyIDRenewalEndpoint* - key pair identifier

- out *CACertRenewalEndpoint* - CA certificate
- out *privateKeyCACertRenewalEndpoint* - CA certificate private key

8. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.37 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in "C=US,O=ONVIF,CN=*Renewal Endpoint IP Address*,C=US" - certificate subject
   - in "*Renewal Endpoint IP Address*" - certificate SAN
   - in *CACertRenewalEndpoint* - CA-certificate
   - in *privateKeyCACertRenewalEndpoint* - private key of CA-certificate for certificate signature
   - out *certRenewalEndpoint* - certificate
   - out *publicKeyRenewalEndpoint* - public key of certificate
   - out *privateKeyRenewalEndpoint* - private key of certificate
9. ONVIF Client starts Renewal endpoint with the following parameters:
   - It is configured to use *certRenewalEndpoint* as a server certificate.
   - It is configured to use *authServerMetadataEndpoint1* as a authentication server and accepts authentication tokens for authorization scope (*scope1*).
   - Accessible with *renewalEndpoint1* endpoint address.
   - For the first requests (until the switch) returns *initialData1* for renewal request with [current time + 3 min] as expiration time
10. Set:
    - *renewalConfiguration*.RenewalEndpoint := *renewalEndpoint1*
    - *renewalConfiguration*.AuthorizationServer := *authServerToken1*
    - *renewalConfiguration*.CertPathValidationPolicyID := *certPathValidationPolicyRenewalId*
11. Set:

- *objectStorageConfiguration*.type := [object storage type, provided by Test Operator for Object Storage Renewal]
- *objectStorageConfiguration*.Region := [some random region with valid formatting]
- *objectStorageConfiguration*.LocalPath is skipped
- *objectStorageConfiguration*.StorageUri := [some random uri with valid formatting]
- *objectStorageConfiguration*.User.UserName := "Invalid"
- *objectStorageConfiguration*.User.Password := "Invalid"
- *objectStorageConfiguration*.User.Extension is skipped
- *objectStorageConfiguration*.User.Token := "Invalid"
- *objectStorageConfiguration*.Extension is skipped
- *objectStorageConfiguration*.ConfigurationRenewal := *renewalConfiguration*

12. ONVIF Client prepares recording with recording job for Media2 profile with at least video configured by following the procedure mentioned in Annex A.1 with the following input and output parameters:
    - in *objectStorageConfiguration* - object storage configuration
    - out *recordingToken1* - recording for test
    - out *recordingConfiguration1* - recording configuration for test
    - out *recordingJobToken1* - recording job for test
    - out *recordingJobConfiguration1* - recording job configuration for test
    - out *recordingSegmentDuration* - recording segment duration
13. ONVIF Client deletes all data from object storage.
14. ONVIF Client invokes **SetRecordingJobMode** request with parameters
    - JobToken := *recordingJobToken1*
    - Mode := **Active**
15. The DUT responds with **SetRecordingJobModeResponse** message.
16. Wait for [Operation Delay].

17. If DUT does not invoke renewal request with valid authorization token, FAIL the test, restore DUT settings and skip other steps.

18. Wait for *recordingSegmentDuration* * 2 + 20%.

19. ONVIF Client verifies data was recorded using *initialData1* for access:

    19.1. If no data was recorded to the storage after renewal request, FAIL the test, restore DUT settings and skip other steps.

20. ONVIF Client updates Renewal data:

    - For the following requests Renewal Endpoint returns error for renewal request with [current time + 3 min] as expiration time

21. Wait for 3 min.

22. If DUT does not invoke renewal request with valid authorization token, FAIL the test, restore DUT settings and skip other steps.

23. Wait for *recordingSegmentDuration* * 2 + 20%.

24. ONVIF Client verifies data was recorded using *initialData1* for access:

    24.1. If no data was recorded to the storage after invalid renewal request, FAIL the test, restore DUT settings and skip other steps.

25. ONVIF Client updates Renewal data:

    - For the following requests Renewal Endpoint returns *renewalData1* for renewal request with [current time + 3 min] as expiration time

26. Wait for 3 min.

27. If DUT does not invoke renewal request with valid authorization token, FAIL the test, restore DUT settings and skip other steps.

28. Wait for *recordingSegmentDuration* * 2 + 20%.

29. ONVIF Client verifies data was recorded using *renewalData1* for access:

    29.1. If no data was recorded to the storage after valid renewal request was received again, FAIL the test, restore DUT settings and skip other steps.

30. ONVIF Client restores DUT configuration if required.

31. ONVIF Client deletes all data from object storage.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **SetRecordingJobModeResponse** message.

5.3 General

5.3.1 GET RECORDING CONFIGURATION WITH INVALID TOKEN

Test Case ID: RECORDING-4-1-3

Specification Coverage: GetRecordingConfiguration (ONVIF Recording Control Service Specification)

Feature Under Test: GetRecordingConfiguration

WSDL Reference: recording.wsdl

Test Purpose: To verify Get Recording Configuration with invalid Token.

Pre-Requisite: ONVIF Client gets the entry point for Recording Control Service by GetServices/ GetCapabilities command.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.
4. Verify that the **GetRecordingsResponse** message contains at least one Recording.
5. If there is at least one Recording, then skip steps 6-7 and go to the step 8.
6. ONVIF Client will invoke **CreateRecordingRequest** message to create new Recording.

7. Verify the **CreateRecordingResponse** message from the DUT.
8. ONVIF Client will invoke **GetRecordingConfigurationRequest** message (invalid RecordingToken).
9. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/NoRecording).
10. Restore DUT settings.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingsResponse** message.
- The DUT did not send a valid **CreateRecordingResponse** message.
- The DUT did not send SOAP 1.2 fault message.
- The DUT sent incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

Note: Other faults than specified in the test are acceptable, though the specified are preferable.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

5.3.2 GET RECORDING JOB STATE

Test Case ID: RECORDING-4-1-7

Specification Coverage: GetRecordingJobState (ONVIF Recording Control Service Specification)

Feature Under Test: GetRecordingJobState

WSDL Reference: recording.wsdl

Test Purpose: To verify Get Recording Job State.

Pre-Requisite: ONVIF Client gets the entry point for Recording Control Service by GetServices/ GetCapabilities command.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.
4. Verify the **GetRecordingsResponse** message from the DUT.
5. ONVIF Client will invoke **GetRecordingJobsRequest** message to retrieve a complete recording jobs list.
6. Verify the **GetRecordingJobsResponse** message from the DUT.
7. ONVIF Client will invoke **GetRecordingJobStateRequest** message (JobToken = "Token1", where Token1 is the first JobItem.JobToken from the **GetRecordingJobsResponse** message) to retrieve recording job state.
8. Verify the **GetRecordingJobStateResponse** message from the DUT.
9. Repeat steps 7-8 for all other recording jobs from the **GetRecordingJobsResponse** message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingsResponse** message.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message.
- The DUT returned different parameter values for the same recording job in **GetRecordingJobsResponse** message and in **GetRecordingJobStateResponse** message (see Annex A.60).
- The DUT returned JobItem.JobConfiguration.RecordingToken that does not exist in **GetRecordingsResponse** message.

Note: If the DUT does not return any recording in **GetRecordingsResponse** message, skip steps 5-9 and go to the next test.

Note: If the DUT does not return any recording job in **GetRecordingJobsResponse** message, skip steps 7-9 and go to the next test.

Note: Two recording jobs supposed to be the same, if they have equal tokens.

5.3.3 GET TRACK CONFIGURATION WITH INVALID TOKEN

Test Case ID: RECORDING-4-1-10

Specification Coverage: GetTrackConfiguration (ONVIF Recording Control Service Specification)

Feature Under Test: GetTrackConfiguration

WSDL Reference: recording.wsdl

Test Purpose: To verify Get Track Configuration with invalid Token.

Pre-Requisite: ONVIF Client gets the entry point for Recording Control Service by GetServices/ GetCapabilities command.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.
4. Verify that the **GetRecordingsResponse** message contains at least one Recording.
5. If there is at least one Recording, then skip steps 6-9 and go to the step 10.
6. ONVIF Client will invoke **CreateRecordingRequest** message to create new Recording.
7. Verify the **CreateRecordingResponse** message from the DUT.
8. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve TrackToken of created recording.
9. Verify **GetRecordingsResponse** message from the DUT (TrackToken = TrackToken1).
10. ONVIF Client will invoke **GetTrackConfigurationRequest** message (invalid TrackToken).
11. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/ NoTrack).
12. ONVIF Client will invoke **GetTrackConfigurationRequest** message (invalid RecordingToken).

13. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/ NoRecording).
14. Restore DUT settings.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingsResponse** message.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT did not send a valid **CreateRecordingResponse** message.
- The DUT did not send SOAP 1.2 fault message.
- The DUT sent an incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

Note: Other faults than specified in the test are acceptable, though the specified are preferable.

Note: Skip the test in case of fault in **CreateRecordingResponse** message.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

5.3.4 GET RECORDING JOB STATE WITH INVALID TOKEN

Test Case ID: RECORDING-4-1-14

Specification Coverage: GetRecordingJobState (ONVIF Recording Control Service Specification)

Feature Under Test: GetRecordingJobState

WSDL Reference: recording.wsdl

Test Purpose: To verify Get Recording Job Sate with invalid Token.

Pre-Requisite: ONVIF Client gets the entry point for Recording Control Service by GetServices/ GetCapabilities command. At least one media profile compatible with at least one existing or created recording exists on the DUT. Options are supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingJobStateRequest** message (invalid JobToken).
4. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/NoRecordingJob).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send SOAP 1.2 fault message.
- The DUT sent an incorrect SOAP 1.2 fault message (fault code, namespace, etc.).

Note: Other faults than specified in the test are acceptable, though the specified are preferable.

5.3.5 RECORDING CONFIGURATION (Static Recording)

Test Case ID: RECORDING-4-1-15

Specification Coverage: GetRecordings, SetRecordingConfiguration, GetRecordingConfiguration, Configuration changes (ONVIF Recording Control Specification)

Feature Under Test: GetRecordings, SetRecordingConfiguration, GetRecordingConfiguration, tns1:RecordingConfig/RecordingConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that a DUT can provide a list of recordings by using GetRecordings command.
- To verify that a DUT can provide current recording configuration by using GetRecordingConfiguration command.
- To verify that a DUT can modify existing recording configuration by using SetRecordingConfiguration command.
- To verify that a DUT will provide a tns1:RecordingConfig/RecordingConfiguration notification on recording configuration update.

- To verify applying of recording source information (Name, Address, SourceId) with maximum length by the DUT during Recording update.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Dynamic Recording is not supported by the DUT as indicated by skipped DynamicRecordings capability or DynamicRecordings = false capability.
- Event Service is received from the DUT.
- Pull-Point Notification feature is supported by the DUT as indicated by MaxPullPoints > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities
4. If *cap*.SupportedTargetFormats contains at least one item:
   4.1. ONVIF Client creates an external storage configuration by following the procedure mentioned in Annex A.11 with the following input and output parameters:
      - out *externalStorage1* - external storage configuration identifier
5. ONVIF Client invokes **GetRecordings** request.
6. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
7. If *recordingsList1* is empty, FAIL the test, restore the DUT settings and skip other steps.
8. If *recordingsList1* contains at least two items with the same RecordingToken, FAIL the test, restore the DUT settings and skip other steps.
9. For each recording item *recordingItem1* in *recordingsList1* repeat the following steps:

9.1. ONVIF Client invokes **GetRecordingConfiguration** request with parameters

- RecordingToken := *recordingItem1*.RecordingToken

9.2. The DUT responds with **GetRecordingConfigurationResponse** with parameters

- RecordingConfiguration =: *recordingConfiguration1*

9.3. If *recordingConfiguration1* contains not the same values as *recordingItem1* (see Annex A.44), FAIL the test, restore the DUT settings and skip other steps.

10. If *cap*.SupportedTargetFormats contains at least one item:

10.1. If *cap*.SupportedEncryptionModes contains at least one item:

10.1.1. Set:

- *encryption1*.Mode := *cap*.SupportedEncryptionModes[0]
- *encryption1*.KID := "8c9769c7-55bc-4aa5-a7dc-23acd2f10cf0"
- *encryption1*.Key := [any valid key]
- *encryption1*.Track is skipped
- *encryption1*.AsymmetricEncryption is skipped

10.2. Set:

- *target1*.Storage := *externalStorage1*
- *target1*.Format := *cap*.SupportedTargetFormats[0]
- *target1*.Prefix := "Pre-1"
- *target1*.Postfix := "Post-1"
- *target1*.SegmentDuration := "PT1M"
- *target1*.Encryption := *encryption1* (if it was set, skipped otherwise)
- *target1*.SegmentDurationOverride is skipped

11. Set:

- *recordingToken1* := *recordingsList1*[0].RecordingToken
- *newRecordingConfiguration1* := *recordingsList1*[0].Configuration

- *newRecordingConfiguration1*.Source.Name := [string with length 20 characters, which contains only readable characters and begins with letter]
- *newRecordingConfiguration1*.Source.SourceId := [valid URI with length 128 characters]
- *newRecordingConfiguration1*.Source.Address := [valid URI with length 128 characters]
- *newRecordingConfiguration1*.Source.Location := "New Source Location"
- *newRecordingConfiguration1*.Source.Description := "New Source Description"
- *newRecordingConfiguration1*.Content := "New Content"
- *newRecordingConfiguration1*.MaximumRetentionTime := [acceptable MaximumRetentionTime, will be taken from Recording\Retention Time field of ONVIF Device Test Tool]
- *newRecordingConfiguration1*.Target := *target1* (if it was set, skipped otherwise)

12. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
    - in **tns1:RecordingConfig/RecordingConfiguration** - Notification Topic
    - out *s* - Subscription Reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription Termination time
13. ONVIF Client invokes **SetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *newRecordingConfiguration1*
14. The DUT responds with **SetRecordingConfigurationResponse** message.
15. ONVIF Client retrieves and checks **tns1:RecordingConfig/RecordingConfiguration** event for the specified recording by following the procedure mentioned in Annex A.43 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time

- in *recordingToken1* - expected recording token
- out *recordingConfiguration2* - received recording configuration

16. If *recordingConfiguration2* contains not the same values as *newRecordingConfiguration1* (see Annex A.44), FAIL the test, restore the DUT settings and skip other steps.
17. ONVIF Client invokes **GetRecordingConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
18. The DUT responds with **GetRecordingConfigurationResponse** with parameters
    - RecordingConfiguration =: *recordingConfiguration3*
19. If *recordingConfiguration3* contains not the same values as *newRecordingConfiguration1* (see Annex A.44), FAIL the test, restore the DUT settings and skip other steps.
20. ONVIF Client closes event subscription if required.
21. ONVIF Client restores the DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingsResponse** message.
- The DUT did not send **GetRecordingConfigurationResponse** message(s).
- The DUT did not send **SetRecordingConfigurationResponse** message.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

5.3.6 RECORDING CONFIGURATION (Dynamic Recording)

Test Case ID: RECORDING-4-1-16

Specification Coverage: GetRecordings, CreateRecordingConfiguration, SetRecordingConfiguration, DeleteRecordingConfiguration, GetRecordingConfiguration, Configuration changes, Recording and track creation and deletion (ONVIF Recording Control Specification)

Feature Under Test: GetRecordings, CreateRecordingConfiguration, SetRecordingConfiguration, DeleteRecordingConfiguration, GetRecordingConfiguration, tns1:RecordingConfig/

RecordingConfiguration, tns1:RecordingConfig/CreateRecording, tns1:RecordingConfig/DeleteRecording

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that a DUT can provide list of recordings by using GetRecordings command.
- To verify that a DUT can provide current recording configuration by using GetRecordingConfiguration command.
- To verify that a DUT can create recording configuration by using CreateRecordingConfiguration command.
- To verify that a DUT can modify existing recording configuration by using SetRecordingConfiguration command.
- To verify that a DUT can delete recording configuration by using DeleteRecordingConfiguration command.
- To verify that a DUT will provide tns1:RecordingConfig/RecordingConfiguration notification on recording configuration update.
- To verify that a DUT will provide tns1:RecordingConfig/CreateRecording notification on recording creation.
- To verify that a DUT will provide tns1:RecordingConfig/DeleteRecording notification on recording deletion.
- To verify applying of recording source information (Name, Address, SourceId) with maximum length by the DUT during Recording creation.
- To verify applying of recording source information (Name, Address, SourceId) with maximum length by the DUT during Recording update.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Dynamic Recording is supported by the DUT as indicated by DynamicRecordings = true capability.
- Event Service is received from the DUT.
- Pull-Point Notification feature is supported by the DUT as indicated by MaxPullPoints > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
4. If *cap*.SupportedTargetFormats contains at least one item:
   4.1. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.11 with the following input and output parameters:
      - out *externalStorage1* - external storage configuration identifier
5. Execute Annex A.48 for possibility to create a new recording.
6. Set:
   - *tracksTypes1* := []
   - If *cap*.Encoding contains at least one enumeration value of tt:VideoEncoding, add *Video* to *tracksTypes1*.
   - If *cap*.Encoding contains at least one enumeration value of tt:AudioEncoding, add *Audio* to *tracksTypes1*.
   - If *cap*.MetadaRecording = true, add *Metadata* to *tracksTypes1*.
7. If *cap*.SupportedTargetFormats contains at least one item:
   7.1. Set:
      - *target1*.Storage := *externalStorage1*
      - *target1*.Format := *cap*.SupportedTargetFormats[0]
      - *target1*.Prefix := "Pre-1"
      - *target1*.Postfix := "Post-1"
      - *target1*.SegmentDuration := "PT1M"
      - *target1*.Encryption is skipped

- *target1*.SegmentDurationOverride is skipped

8. Set:
   - *initialRecordingConfiguration1*.Source.Name := [string with length 20 characters, which contains only readable characters and begins with letter]
   - *initialRecordingConfiguration1*.Source.SourceId := [valid URI with length 128 characters]
   - *initialRecordingConfiguration1*.Source.Address := [valid URI with length 128 characters]
   - *initialRecordingConfiguration1*.Source.Location := "Source Location"
   - *initialRecordingConfiguration1*.Source.Description := "Source Description"
   - *initialRecordingConfiguration1*.Content := "Content"
   - *initialRecordingConfiguration1*.MaximumRetentionTime := [acceptable MaximumRetentionTime, will be taken from Recording\Retention Time field of ONVIF Device Test Tool]
   - *initialRecordingConfiguration1*.Target := *target1* (if it was set, skipped otherwise)
9. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters:
   - in **tns1:RecordingConfig//.** - Notification Topic
   - out *s* - Subscription Reference
   - out *currentTime* - current time for the DUT
   - out *terminationTime* - Subscription Termination time
10. ONVIF Client invokes **CreateRecording** request with parameters:
    - RecordingConfiguration := *initialRecordingConfiguration1*
11. The DUT responds with **CreateRecordingResponse** message with parameters:
    - RecordingToken =: *recordingToken1*
12. ONVIF Client retrieves and checks **tns1:RecordingConfig/CreateRecording** event for the specified recording by following the procedure mentioned in Annex A.46 with the following parameters:
    - in *s* - Subscription reference

- in *currentTime* - current time for the DUT
- in *terminationTime* - subscription termination time
- in *recordingToken1* - expected recording token

13. ONVIF Client invokes **GetRecordings** request.
14. The DUT responds with **GetRecordingsResponse** message with parameters:
    - RecordingItem list =: *recordingsList1*
15. If *recordingsList1* contains at least two items with the same RecordingToken, FAIL the test, restore DUT settings and skip other steps.
16. If *recordingsList1* does not contain item with RecordingToken = *recordingToken1*, FAIL the test, restore DUT settings and skip other steps.
17. Set:
    - *recordingItem1* := item from *recordingsList1* with RecordingToken = *recordingToken1*
18. If *recordingItem1*.Configuration contains not the same values as *initialRecordingConfiguration1* (see Annex A.44), FAIL the test, restore DUT settings and skip other steps.
19. If *recordingItem1*.Tracks contains at least two items with the same TrackToken, FAIL the test, restore DUT settings and skip other steps.
20. If *recordingItem1*.Tracks contains at least one item with Description other than "" (empty string), FAIL the test, restore DUT settings and skip other steps.
21. For each track type that should be automatically created *tracksType1* in *tracksTypes1*:
    21.1. If *recordingItem1*.Tracks does not contain exactly one item with TrackType = *tracksType1*, FAIL the test, restore DUT settings and skip other steps.
    21.2. Set *track1* := item from *recordingItem1*.Tracks with TrackType = *tracksType1*.
    21.3. If *tracksType1* is Video and *track1*.TrackToken is not equal 'VIDEO001', FAIL the test, restore DUT settings and skip other steps.
    21.4. If *tracksType1* is Audio and *track1*.TrackToken is not equal 'AUDIO001', FAIL the test, restore DUT settings and skip other steps.
    21.5. If *tracksType1* is Metadata and *track1*.TrackToken is not equal 'META001', FAIL the test, restore DUT settings and skip other steps.

22. ONVIF Client invokes **GetRecordingConfiguration** request with parameters:
    - RecordingToken := *recordingToken1*
23. The DUT responds with **GetRecordingConfigurationResponse** with parameters:
    - RecordingConfiguration =: *recordingConfiguration1*
24. If *recordingConfiguration1* contains not the same values as *recordingItem1* (see Annex A.44), FAIL the test, restore DUT settings and skip other steps.
25. If *cap*.SupportedTargetFormats contains at least one item and *cap*.OnboardStorage = false:
    25.1. If *cap*.SupportedEncryptionModes contains at least one item:
        25.1.1 Set:
        - *encryption2*.Mode := *cap*.SupportedEncryptionModes[0]
        - *encryption2*.KID := "a5a22337-d7b9-450c-a4ed-05cacde597ac"
        - *encryption2*.Key := [any valid key]
        - *encryption2*.Track is skipped
        - *encryption2*.AsymmetricEncryption is skipped

    25.2. Set:
    - *target2*.Storage := *externalStorage1*
    - *target2*.Format := *cap*.SupportedTargetFormats[1], if *cap*.SupportedTargetFormats contains more then one value, otherwise *cap*.SupportedTargetFormats[0]
    - *target2*.Prefix := "Pre-2"
    - *target2*.Postfix := "Post-2"
    - *target2*.SegmentDuration := "PT2M"
    - *target2*.Encryption := *encryption2* (if it was set, skipped otherwise)
    - *target2*.SegmentDurationOverride is skipped
26. Set:
    - *newRecordingConfiguration1* := *recordingsList1*[0].Configuration

- *newRecordingConfiguration1*.Source.Name := [string with length 20 characters, which contains only readable characters and begins with letter, other than at *initialRecordingConfiguration1*]
- *newRecordingConfiguration1*.Source.SourceId := [valid URI with length 128 characters, other than at *initialRecordingConfiguration1*]
- *newRecordingConfiguration1*.Source.Address := [valid URI with length 128 characters, other than at *initialRecordingConfiguration1*]
- *newRecordingConfiguration1*.Source.Location := "New Source Location"
- *newRecordingConfiguration1*.Source.Description := "New Source Description"
- *newRecordingConfiguration1*.Content := "New Content"
- *newRecordingConfiguration1*.MaximumRetentionTime := [acceptable MaximumRetentionTime, will be taken from Recording\Retention Time field of ONVIF Device Test Tool]
- *newRecordingConfiguration1*.Target := *target2* (if it was set and *cap*.OnboardStorage = false, skipped otherwise)

27. ONVIF Client invokes **SetRecordingConfiguration** request with parameters:
    - RecordingToken := *recordingToken1*
    - RecordingConfiguration := *newRecordingConfiguration1*
28. The DUT responds with **SetRecordingConfigurationResponse** message.
29. ONVIF Client retrieves and checks **tns1:RecordingConfig/RecordingConfiguration** event for the specified recording by following the procedure mentioned in Annex A.43 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingToken1* - expected recording token
    - out *recordingConfiguration2* - received recording configuration
30. If *recordingConfiguration2* contains not the same values as *newRecordingConfiguration1* (see Annex A.44), FAIL the test, restore DUT settings and skip other steps.

31. ONVIF Client invokes **GetRecordingConfiguration** request with parameters:
    - RecordingToken := *recordingToken1*
32. The DUT responds with **GetRecordingConfigurationResponse** message with the following parameters:
    - RecordingConfiguration =: *recordingConfiguration3*
33. If *recordingConfiguration3* contains not the same values as *newRecordingConfiguration1* (see Annex A.44), FAIL the test, restore DUT settings and skip other steps.
34. ONVIF Client invokes **DeleteRecording** request with the following parameters:
    - RecordingToken := *recordingToken1*
35. The DUT responds with **DeleteRecordingResponse** message.
36. ONVIF Client retrieves and checks **tns1:RecordingConfig/DeleteRecording** event for the specified recording by following the procedure mentioned in Annex A.47 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingToken1* - expected recording token
37. ONVIF Client invokes **GetRecordings** request.
38. The DUT responds with **GetRecordingsResponse** message with parameters:
    - RecordingItem list =: *recordingsList2*
39. If *recordingsList2* contains item with RecordingToken = *recordingToken1*, FAIL the test, restore DUT settings and skip other steps.
40. ONVIF Client close event subscription if required.
41. ONVIF Client restores DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingsResponse** message(s).
- The DUT did not send **GetRecordingConfigurationResponse** message(s).
- The DUT did not send **CreateRecordingResponse** message.
- The DUT did not send **SetRecordingConfigurationResponse** message.
- The DUT did not send **DeleteRecordingResponse** message.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

5.3.7 TRACK CONFIGURATION (Static Track)

Test Case ID: RECORDING-4-1-17

Specification Coverage: GetRecordings, SetTrackConfiguration, GetTrackConfiguration, Configuration changes (ONVIF Recording Control Specification)

Feature Under Test: GetRecordings, SetTrackConfiguration, GetTrackConfiguration, tns1:RecordingConfig/TrackConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that a DUT can provide a list of tracks for recordings by using GetRecordings command.
- To verify that a DUT can provide current track configuration by using GetTrackConfiguration command and GetRecordings command.
- To verify that a DUT can modify existing track configuration by using SetTrackConfiguration command.
- To verify that a DUT will provide tns1:RecordingConfig/TrackConfiguration notification on track configuration update.
- To verify that a DUT will ignore TrackType for SetTrackConfiguration command.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Dynamic Tracks is not supported by the DUT as indicated by skipped DynamicTracks capability or DynamicTracks = false capability.
- Event Service is received from the DUT.

- Pull-Point Notification feature is supported by the DUT as indicated by MaxPullPoints > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities
4. If *cap*.DynamicRecording = true:
   4.1. ONVIF Client creates new recording by following the procedure mentioned in Annex A.51.
5. ONVIF Client invokes **GetRecordings** request.
6. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
7. If *recordingsList1* is empty, FAIL the test, restore the DUT settings and skip other steps.
8. For each recording item *recordingItem1* in *recordingsList1* repeat the following steps:
   8.1. If *recordingItem1* has empty Tracks, FAIL the test, restore DUT settings and skip other steps.
   8.2. If *recordingItem1* contains at least two items with the same TrackToken, FAIL the test, restore DUT settings and skip other steps.
   8.3. For each track item *trackItem1* in *recordingItem1* repeat the following steps:
      8.3.1. ONVIF Client invokes **GetTrackConfiguration** request with parameters
         - RecordingToken := *recordingItem1*.RecordingToken
         - TrackToken := *trackItem1*.TrackToken
      8.3.2. The DUT responds with **GetTrackConfigurationResponse** with parameters
         - TrackConfiguration =: *trackConfiguration1*

8.3.3.If *trackConfiguration1* contains not the same values as *trackItem1*.Configuration (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.

9. Set:
   - *recordingToken1* := *recordingsList1*[0].RecordingToken
   - *trackToken1* := *recordingsList1*[0].Tracks[0].TrackToken
   - *initialTrackConfiguration1* := *recordingsList1*[0].Tracks[0].Configuration
   - *newTrackConfiguration1*.TrackType := [any other value than *initialTrackConfiguration1*.TrackType from **Video**, **Audio** and **Metadata**]
   - *newTrackConfiguration1*.Description := "New Description"
10. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
    - in **tns1:RecordingConfig/TrackConfiguration** - Notification Topic
    - out *s* - Subscription Reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription Termination time
11. ONVIF Client invokes **SetTrackConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
    - TrackToken := *trackToken1*
    - TrackConfiguration := *newTrackConfiguration1*
12. The DUT responds with **SetTrackConfigurationResponse** message.
13. ONVIF Client retrieves and checks **tns1:RecordingConfig/TrackConfiguration** event for the specified recording by following the procedure mentioned in Annex A.49 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingToken1* - expected recording token

- in *trackToken1* - expected track token
- out *trackConfiguration2* - received track configuration

14. If *trackConfiguration2* contains not the same values as *newTrackConfiguration1*, except TrackType value (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
15. If *trackConfiguration2*.TrackType is not equal to *initialTrackConfiguration1* .TrackType, FAIL the test, restore DUT settings and skip other steps.
16. ONVIF Client invokes **GetTrackConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
    - TrackToken := *trackToken1*
17. The DUT responds with **GetTrackConfigurationResponse** with parameters
    - TrackConfiguration := *trackConfiguration3*
18. If *trackConfiguration3* contains not the same values as *newTrackConfiguration1*, except TrackType value (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
19. If *trackConfiguration3*.TrackType is not equal to *initialTrackConfiguration1* .TrackType, FAIL the test, restore DUT settings and skip other steps.
20. ONVIF Client close event subscription if required.
21. ONVIF Client restores DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingsResponse** message.
- The DUT did not send **GetTrackConfigurationResponse** message(s).
- The DUT did not send **SetTrackConfigurationResponse** message.

5.3.8 TRACK CONFIGURATION (Dynamic Track)

Test Case ID: RECORDING-4-1-18

Specification Coverage: GetRecordings, GetRecordingOptions, CreateTrack, SetTrackConfiguration, DeleteTrack, GetTrackConfiguration, Configuration changes, Recording and track creation and deletion (ONVIF Recording Control Specification)

Feature Under Test: GetRecordings, GetRecordingOptions, CreateTrack, SetTrackConfiguration, DeleteTrack, GetTrackConfiguration, tns1:RecordingConfig/TrackConfiguration, tns1:RecordingConfig/CreateTrack, tns1:RecordingConfig/DeleteTrack

WSDL Reference: recording.wsdl

Test Purpose:

- To verify that a DUT can provide a list of tracks by using GetRecordings command.
- To verify that a DUT can provide current track configuration by using GetTrackConfiguration command.
- To verify that a DUT can provide track options by using GetRecordingOptions command.
- To verify that a DUT can create a track by using CreateTrack command.
- To verify that a DUT can modify existing track by using SetTrackConfiguration command.
- To verify that a DUT can delete track by using DeleteTrack command.
- To verify that a DUT will provide tns1:RecordingConfig/TrackConfiguration notification on track configuration update.
- To verify that a DUT will provide tns1:RecordingConfig/CreateTrack notification on track creation.
- To verify that a DUT will provide tns1:RecordingConfig/DeleteTrack notification on track deletion.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Dynamic Tracks is supported by the DUT as indicated by DynamicTracks = true capability.
- Event Service is received from the DUT.
- Pull-Point Notification feature is supported by the DUT as indicated by MaxPullPoints > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
4. ONVIF Client selects recording for test by following the procedure mentioned in Annex A.54 with the following input and output parameters:
   - out *recordingToken1* - recording token to be used for test
5. ONVIF Client invokes **GetRecordingOptions** request with parameters:
   - RecordingToken := *recordingToken1*
6. The DUT responds with **GetRecordingOptionsResponse** message with parameters:
   - Options =: *recordingOptions1*
7. If *recordingOptions1* does not contain Track, FAIL the test, restore DUT settings and skip other steps.
8. If *recordingOptions1*.Track.SpareTotal is not specified or less than 1, FAIL the test, restore DUT settings and skip other steps.
9. If *recordingOptions1*.Track.SpareVideo is specified and more than 0, set *trackType* := Video and go to the step 13.
10. If *recordingOptions1*.Track.SpareAudio is specified and more than 0, set *trackType* := Audio and go to the step 13.
11. If *recordingOptions1*.Track.SpareMetadata is specified and more than 0, set *trackType* := Metadata and go to the step 13.
12. Set *trackType* := Video.
13. Set:
    - *initialTrackConfiguration1*.TrackType := *trackType*
    - *initialTrackConfiguration1*.Description := "Description 1"
14. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters

- in **tns1:RecordingConfig//.** - Notification Topic
- out *s* - Subscription Reference
- out *currentTime* - current time for the DUT
- out *terminationTime* - Subscription Termination time

15. ONVIF Client invokes **CreateTrack** request with parameters:
    - RecordingToken := *recordingToken1*
    - TrackConfiguration := *initialTrackConfiguration1*
16. The DUT responds with **CreateTrackResponse** message with parameters:
    - TrackToken =: *trackToken1*
17. ONVIF Client retrieves and checks **tns1:RecordingConfig/CreateTrack** event for the specified track by following the procedure mentioned in Annex A.52 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingToken1* - expected recording token
    - in *trackToken1* - expected track token
18. ONVIF Client invokes **GetRecordings** request.
19. The DUT responds with **GetRecordingsResponse** message with parameters:
    - RecordingItem list =: *recordingsList1*
20. If *recordingsList1* does not contain item with RecordingToken = *recordingToken1*, FAIL the test, restore DUT settings and skip other steps.
21. Set:
    - *recordingItem1* := item from *recordingsList1* with RecordingToken = *recordingToken1*
22. If *recordingItem1*.Tracks contains at least two items with the same TrackToken, FAIL the test, restore DUT settings and skip other steps.

23. If *recordingItem1*.Tracks does not contain an item with TrackToken = *trackToken1*, FAIL the test, restore DUT settings and skip other steps.
24. Set:
    - *trackItem1* := item from *recordingItem1*.Tracks with TrackToken = *trackToken1*
25. If *trackItem1*.Configuration contains not the same values as *initialTrackConfiguration1* (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
26. ONVIF Client invokes **GetTrackConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
    - TrackToken := *trackToken1*
27. The DUT responds with **GetTrackConfigurationResponse** message with parameters:
    - TrackConfiguration =: *trackConfiguration1*
28. If *trackConfiguration1* contains not the same values as *trackItem1* (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
29. Set:
    - *recordingToken1* := *recordingsList1*[0].RecordingToken
    - *trackToken1* := *recordingsList1*[0].Tracks[0].TrackToken
    - *initialTrackConfiguration1* := *recordingsList1*[0].Tracks[0].Configuration
    - *newTrackConfiguration1*.TrackType := [any value other than *initialTrackConfiguration1*.TrackType from **Video**, **Audio** and **Metadata**]
    - *newTrackConfiguration1*.Description := "New Description"
30. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
    - in **tns1:RecordingConfig/TrackConfiguration** - Notification Topic
    - out *s2* - Subscription Reference
    - out *currentTime2* - current time for the DUT
    - out *terminationTime2* - Subscription Termination time
31. ONVIF Client invokes **SetTrackConfiguration** request with parameters:

- RecordingToken := *recordingToken1*
- TrackToken := *trackToken1*
- TrackConfiguration := *newTrackConfiguration1*

32. The DUT responds with **SetTrackConfigurationResponse** message.
33. ONVIF Client retrieves and checks **tns1:RecordingConfig/TrackConfiguration** event for the specified recording and track by following the procedure mentioned in Annex A.49 with the following parameters:
    - in *s2* - Subscription reference
    - in *currentTime2* - current time for the DUT
    - in *terminationTime2* - subscription termination time
    - in *recordingToken1* - expected recording token
    - in *trackToken1* - expected track token
    - out *trackConfiguration2* - received track configuration
34. If *trackConfiguration2* contains not the same values as *newTrackConfiguration1*, except TrackType value (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
35. If *trackConfiguration2*.TrackType is not equal to *initialTrackConfiguration1*.TrackType, FAIL the test, restore DUT settings and skip other steps.
36. ONVIF Client invokes **GetTrackConfiguration** request with parameters
    - RecordingToken := *recordingToken1*
    - TrackToken := *trackToken1*
37. The DUT responds with **GetTrackConfigurationResponse** message with parameters:
    - TrackConfiguration := *trackConfiguration3*
38. If *trackConfiguration3* contains not the same values as *newTrackConfiguration1*, except TrackType value (see Annex A.50), FAIL the test, restore DUT settings and skip other steps.
39. If *trackConfiguration3*.TrackType is not equal to *initialTrackConfiguration1*.TrackType, FAIL the test, restore DUT settings and skip other steps.
40. ONVIF Client invokes **DeleteTrack** request with the following parameters:

- RecordingToken := *recordingToken1*
- TrackToken := *trackToken1*

41. The DUT responds with **DeleteTrackResponse** message.
42. ONVIF Client retrieves and checks **tns1:RecordingConfig/DeleteTrack** event for the specified recording by following the procedure mentioned in Annex A.53 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingToken1* - expected recording token
    - in *trackToken1* - expected track token
43. ONVIF Client invokes **GetRecordings** request.
44. The DUT responds with **GetRecordingsResponse** message with parameters:
    - RecordingItem list =: *recordingsList2*
45. Set:
    - *recordingItem2* := item from *recordingsList2* with RecordingToken = *recordingToken1*
46. If *recordingItem2*.Tracks contains item with TrackToken = *trackToken1*, FAIL the test, restore DUT settings and skip other steps.
47. ONVIF Client closes event subscription if required.
48. ONVIF Client restores DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingsResponse** message(s).
- The DUT did not send **GetTrackConfigurationResponse** message(s).

- The DUT did not send **CreateTrackResponse** message.
- The DUT did not send **SetTrackConfigurationResponse** message.
- The DUT did not send **DeleteTrackResponse** message.

5.3.9 RECORDING JOB CONFIGURATION

Test Case ID: RECORDING-4-1-19

Specification Coverage: GetRecordingJobs, CreateRecordingJob, DeleteRecordingJob, GetRecordingOptions, GetRecordingJobConfiguration, SetRecordingJobConfiguration (ONVIF Recording Control Specification)

Feature Under Test: GetRecordingJobs, CreateRecordingJob, DeleteRecordingJob, GetRecordingOptions, GetRecordingJobConfiguration, SetRecordingJobConfiguration, tns1:RecordingConfig/RecordingJobConfiguration

WSDL Reference: recording.wsdl

Test Purpose:

- To verify creation of recording jobs using CreateRecordingJob command.
- To verify that a DUT can provide spare jobs and compatible sources for recording using GetRecordingOptions command.
- To verify that a DUT can provide current list of recording jobs by using GetRecordingJobs command.
- To verify that a DUT can provide current recording job configuration by using GetRecordingJobConfiguration command.
- To verify that a DUT can modify recording job by using SetRecordingJobConfiguration command.
- To verify that a DUT can delete recording job by using DeleteRecordingJob command.
- To verify that a DUT return env:Sender/ter:InvalidArgVal/ter:NoRecordingJob SOAP 1.2 Fault for GetRecordingJobConfiguration command for deleted recording job.
- To verify that a DUT will provide tns1:RecordingConfig/RecordingJobConfiguration notification on recording job configuration update.

Pre-Requisite:

- Recording Control Service is received from the DUT.

- Event Service is received from the DUT.
- Pull-Point Notification feature is supported by the DUT as indicated by MaxPullPoints > 0 capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client prepares recording for which recording job could be created by following the procedure mentioned in Annex A.57 with the following input and output parameters:
   - out *recordingToken1* - recording for test
4. ONVIF Client invokes **GetRecordingOptions** request with parameters:
   - RecordingToken := *recordingToken1*
5. The DUT responds with **GetRecordingOptionsResponse** message with parameters:
   - Options =: *recordingOptions1*
6. If *recordingOptions1*.Job.Spare is skipped or less than 1, FAIL the test, restore DUT settings and skip other steps.
7. If Receiver Service is supported by the DUT:
   7.1. Set:
      - *initialSource1*.SourceToken is skipped
      - *initialSource1*.AutoCreateReceiver := true
8. If Receiver Service is not supported by the DUT:
   8.1. If *recordingOptions1*.Job.CompatibleSources is skipped or empty, FAIL the test, restore DUT settings and skip other steps.
   8.2. Set:
      - *initialSource1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
      - *initialSource1*.SourceToken.Token = *recordingOptions1*.Job.CompatibleSources[0]
      - *initialSource1*.AutoCreateReceiver is skipped

9. Set:
   - *initialSource1*.Tracks is skipped
   - *initialSource1*.Extension is skipped
10. Set:
   - *initialRecordingJobConfiguration1*.ScheduleToken is skipped
   - *initialRecordingJobConfiguration1*.RecordingToken := *recordingToken1*
   - *initialRecordingJobConfiguration1*.Mode := **Idle**
   - *initialRecordingJobConfiguration1*.Priority := 0
   - *initialRecordingJobConfiguration1*.Source[0] := *initialSource1*
   - *initialRecordingJobConfiguration1*.Extension is skipped
   - *initialRecordingJobConfiguration1*.EventFilter is skipped
11. ONVIF Client invokes **CreateRecordingJob** request with parameters:
   - JobConfiguration := *initialRecordingJobConfiguration1*
12. The DUT responds with **CreateRecordingJobResponse** message with parameters:
   - JobToken =: *recordingJobToken1*
   - JobConfiguration =: *createdRecordingJobConfiguration1*
13. If *createdRecordingJobConfiguration1* contains not the same values as *initialRecordingJobConfiguration1* except Tracks, AutoCreateReceiver, and SourceToken (if was omitted at creation request) field (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
14. ONVIF Client invokes **GetRecordingJobs** request.
15. The DUT responds with **GetRecordingJobsResponse** message with parameters:
   - JobItem list =: *recordingJobsList1*
16. If *recordingJobsList1* contains at least two items with the same JobToken, FAIL the test, restore DUT settings and skip other steps.
17. If *recordingJobsList1* does not contain item with JobToken = *recordingJobToken1*, FAIL the test, restore DUT settings and skip other steps.

18. Set:
    - *recordingJobItem1* := item from *recordingJobsList1* with JobToken = *recordingJobToken1*
19. If *recordingJobItem1*.JobConfiguration contains not the same values as *createdRecordingJobConfiguration1* (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
20. ONVIF Client invokes **GetRecordingJobConfiguration** request with parameters:
    - JobToken := *recordingJobToken1*
21. The DUT responds with **GetRecordingJobConfigurationResponse** with parameters
    - JobConfiguration =: *recordingJobConfiguration1*
22. If *recordingJobConfiguration1* contains not the same values as *createdRecordingJobConfiguration1* (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
23. Set:
    - *newRecordingJobConfiguration1*.ScheduleToken is skipped
    - *newRecordingJobConfiguration1*.RecordingToken := *recordingToken1*
    - *newRecordingJobConfiguration1*.Mode := **Active**
    - *newRecordingJobConfiguration1*.Priority := 10
    - *newRecordingJobConfiguration1*.Source[0] := *recordingJobConfiguration1*.Source[0]
    - *newRecordingJobConfiguration1*.Extension is skipped
    - *newRecordingJobConfiguration1*.EventFilter is skipped
24. ONVIF Client creates PullPoint subscription for the specified topic by following the procedure mentioned in Annex A.8 with the following input and output parameters
    - in **tns1:RecordingConfig/RecordingJobConfiguration** - Notification Topic
    - out *s* - Subscription Reference
    - out *currentTime* - current time for the DUT
    - out *terminationTime* - Subscription Termination time
25. ONVIF Client invokes **SetRecordingJobConfiguration** request with parameters:

- JobToken := *recordingJobToken1*
- JobConfiguration := *newRecordingJobConfiguration1*

26. The DUT responds with **SetRecordingJobConfigurationResponse** message with parameters:
    - JobConfiguration =: *updatedRecordingJobConfiguration1*
27. If *updatedRecordingJobConfiguration1* contains not the same values as *newRecordingJobConfiguration1* (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
28. ONVIF Client retrieves and checks **tns1:RecordingConfig/RecordingJobConfiguration** event for the specified recording by following the procedure mentioned in Annex A.55 with the following parameters:
    - in *s* - Subscription reference
    - in *currentTime* - current time for the DUT
    - in *terminationTime* - subscription termination time
    - in *recordingJobToken1* - expected recording job token
    - out *recordingJobConfiguration2* - received recording job configuration
29. If *recordingJobConfiguration2* contains not the same values as *updatedRecordingJobConfiguration1* (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
30. ONVIF Client invokes **GetRecordingJobConfiguration** request with parameters:
    - JobToken := *recordingJobToken1*
31. The DUT responds with **GetRecordingJobConfigurationResponse** message with the following parameters:
    - JobConfiguration =: *recordingJobConfiguration3*
32. If *recordingJobConfiguration3* contains not the same values as *updatedRecordingJobConfiguration1* (see Annex A.56), FAIL the test, restore DUT settings and skip other steps.
33. ONVIF Client invokes **DeleteRecordingJob** request with the following parameters:
    - JobToken := *recordingJobToken1*

34. The DUT responds with **DeleteRecordingJobResponse** message.
35. ONVIF Client invokes **GetRecordingJobs** request.
36. The DUT responds with **GetRecordingJobsResponse** message with parameters:
    - JobItem list =: *recordingJobsList2*
37. If *recordingJobsList2* contains item with JobToken = *recordingJobToken1*, FAIL the test, restore DUT settings and skip other steps.
38. ONVIF Client invokes **GetRecordingJobConfiguration** request with parameters:
    - JobToken := *recordingJobToken1*
39. The DUT returns **env:Sender\ter:InvalidArgVal\ter:NoRecordingJob** SOAP 1.2 fault.
40. ONVIF Client closes event subscription if required.
41. ONVIF Client restores DUT configuration if required.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **CreateRecordingJobResponse** message.
- The DUT did not send **GetRecordingJobsResponse** message.
- The DUT did not send **GetRecordingJobConfigurationResponse** message.
- The DUT did not send **SetRecordingJobConfigurationResponse** message.
- The DUT did not send **DeleteRecordingJobResponse** message.

5.4 Events

5.4.1 RECORDING CONTROL – CREATE TRACK EVENT

Test Case ID: RECORDING-5-1-8

Specification Coverage: Recording and track creation and deletion (ONVIF Recording Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/CreateTrack event format in TopicSet.

Pre-Requisite: Event Service is supported by the DUT. Dynamic Recording or dynamic track functionality is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventPropertiesRequest** message to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:RecordingConfig/CreateTrack. If there is no event with such Topic, skip other steps, fail the test and go to the next test.
6. Check that this event is not a Property event (MessageDescription.IsProperty = false).
7. Check that this event contains Source.SimpleItemDescription item with Name = "RecordingToken" and Type = "tt:RecordingReference".
8. Check that this event contains Source.SimpleItemDescription item with Name = "TrackToken" and Type = "tt:TrackReference".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse** message.
- The DUT does not return valid tns1:RecordingConfig/CreateTrack Topic in **GetEventPropertiesResponse**.

5.4.2 RECORDING CONTROL – DELETE TRACK EVENT

Test Case ID: RECORDING-5-1-9

Specification Coverage: Recording and track creation and deletion (ONVIF Recording Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/DeleteTrack event format in TopicSet.

Pre-Requisite: Event Service is supported by the DUT. Dynamic Recording or dynamic track functionality is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventPropertiesRequest** message to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:RecordingConfig/DeleteTrack. If there is no event with such Topic skip other steps, fail the test and go to the next test.
6. Check that this event is not a Property event (MessageDescription.IsProperty = false).
7. Check that this event contains Source.SimpleItemDescription item with Name = "RecordingToken" and Type = "tt:RecordingReference".
8. Check that this event contains Source.SimpleItemDescription item with Name = "TrackToken" and Type = "tt:TrackReference".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse** message.
- The DUT does not return a valid tns1:RecordingConfig/DeleteTrack Topic in **GetEventPropertiesResponse**.

5.4.3 RECORDING CONTROL – DELETE TRACK DATA EVENT

Test Case ID: RECORDING-5-1-14

Specification Coverage: Data deletion (ONVIF Recording Control Service Specification)

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/DeleteTrackData event format in TopicSet.

Pre-Requisite: Event Service was received from the DUT. Dynamic Recording or dynamic track functionality is supported by the DUT. tns1:RecordingConfig/DeleteTrackData event is supported by the DUT. Device supports Pull-Point Notification feature.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventPropertiesRequest** message to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:RecordingConfig/DeleteTrackData. If there is no event with such Topic, skip other steps, fail the test and go to the next test.
6. Check that this event is not a Property event (MessageDescription.IsProperty = false).
7. Check that this event contains Source.SimpleItemDescription item with Name = "RecordingToken" and Type = "tt:RecordingReference".
8. Check that this event contains Source.SimpleItemDescription item with Name = "TrackToken" and Type = "tt:TrackReference".
9. Check that this event contains Data.SimpleItemDescription item with Name = "StartTime" and Type = "xs:dateTime".
10. Check that this event contains Data.SimpleItemDescription item with Name = "EndTime" and Type = "xs:dateTime".

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse** message.
- The DUT does not return a valid tns1:RecordingConfig/DeleteTrackData Topic in **GetEventPropertiesResponse**.

5.4.4 RECORDING CONTROL – JOB STATE CHANGE EVENT

Test Case ID: RECORDING-5-1-19

Specification Coverage: Recording job state changes (ONVIF Recording Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetRecordingJobState

WSDL Reference: event.wsdl, recording.wsdl

Test Purpose: To verify tns1:RecordingConfig/JobState event generation after property was changed and to verify tns1:RecordingConfig/JobState event format. Options are supported by the DUT.

Pre-Requisite:

- Event Service was received from the DUT.
- ONVIF Client gets the entry point for Recording Control Service by GetServices command.
- Media Service or Receiver Service was received from the DUT. At least one recording exists.
- All recording jobs were stopped. At least one media profile compatible with at least one existing or created recording exists on the DUT.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT.

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingJobsRequest** message to retrieve a complete recording jobs list.
4. If there is at least one RecordingJob, then skip steps 5-10 and go to the step 11.
5. If Receiver service is supported, then ONVIF Client execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
6. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = "RecordingToken1", JobConfiguration.Mode = "Idle", JobConfiguration.Priority = 1, no JobConfiguration.Source.SourceToken.Token, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Receiver", JobConfiguration.Source.AutoCreateReceiver = true) to create a recording job and auto create receiver.
7. Verify **CreateRecordingJobResponse** from the DUT and go to step 11.
8. If Receiver service is not supported ONVIF Client execute Annex A.66 with RequiredSpareJobs = 0 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 1 recording job can be created and to get the compatible media profile tokens list.
9. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Idle, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job with Idle mode.
10. Verify the **CreateRecordingJobResponse** message from the DUT.
11. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime of timeout1.
12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

13. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
14. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 14 until Notification for selected Recording Job is received.
15. ONVIF Client will invoke **SetRecordingJobModeRequest** message (JobToken = [selected job token], Mode = [other than current]) to change Recording Job state.
16. Verify **SetRecordingJobModeResponse** message from the DUT.
17. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
18. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 17 until Notification for selected Recording Job is received.
19. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).
20. Verify that TopicExpression is equal to tns1:RecordingConfig/JobState for received Notification message.
21. Verify that notification contains Source.SimpleItem item with Name = "RecordingJobToken" and Value is equal to one of existing Recording Job Tokens (e.g. complete list of Recording Jobs contains Recording Job with the same token). Verify that there are Notification messages for selected Recording Job.
22. Verify that notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to xs:string.
23. If notification contains Data.ElementItem item with Name = "Information", verify that its Value type is equal to tt:RecordingJobStateInformation (validation with XML Schema complex type).
24. Verify that Notify PropertyOperation = "Changed".
25. ONVIF Client will invoke **GetRecordingJobStateRequest** message for selected Recording Job with corresponding tokens.
26. Verify the **GetRecordingJobStateResponse** messages from the DUT.
27. Verify that Data.ElementItem item with Name = "State" from Notification message has the same value with State.State element from corresponding **GetRecordingJobStateResponse** messages for selected Recording Job.

28. If Information element present in notification

    28.1 Verify that Data.ElementItem item with Name = "Information" from Notification message has the same value with State element from corresponding **GetRecordingJobStateResponse** messages for selected Recording Job.

29. Restore DUT settings.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **SetRecordingJobModeResponse** message.
- The DUT did not send a valid SubscriptionReference.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message
- The DUT did not send a Notification message that contains a property event tns1:RecordingConfig/JobState for selected Recording Job.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem, Data.SimpleItem, or Data.ElementItem wrong type of Value fields, invalid RecordingJobToken, State or Information values, PropertyOperation is not equal to "Changed").

Note: The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

Note: ONVIF Client at steps 15 and 18 will wait for Notification messages until notification for selected Recording Job is received or Operation Delay after last notification expires.

Note: The Renew has to be used for renew subscription during test, if InitialTerminationTime expires. If DUT returns UnacceptableTerminationTimeFault, resend Renew request with acceptable InitialTerminationTime from UnacceptableTerminationTimeFault.

Note: If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

Note: timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

Note: All Notification messages for Recording Job other than selected will be ignored during this test.

5.4.5 RECORDING CONTROL - RECORDING CONFIGURATION EVENT PROPERTIES

Test Case ID: RECORDING-5-1-21

Specification Coverage: Configuration changes (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/RecordingConfiguration

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/RecordingConfiguration event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*

- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:RecordingConfig/RecordingConfiguration topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/RecordingConfiguration topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is equal to true, FAIL the test and skip other steps.
   - If *topic1* does not contain an MessageDescription.Source.SimpleItemDescription item with Name = "RecordingToken", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingToken" does not have Type = "tt:RecordingReference", FAIL the test and skip other steps.
   - If *topic1* does not contain an MessageDescription.Data.ElementItemDescription item with Name = "Configuration", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Data.ElementItemDescription with Name = "Configuration" does not have Type = "tt:RecordingConfiguration", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.6 RECORDING CONTROL - CREATE RECORDING EVENT PROPERTIES

Test Case ID: RECORDING-5-1-22

Specification Coverage: Recording and track creation and deletion (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/CreateRecording

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/CreateRecording event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.
- Dynamic Recording is supported by the DUT as indicated by DynamicRecordings = true capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:RecordingConfig/CreateRecording topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/CreateRecording topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is equal to true, FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "RecordingToken", FAIL the test and skip other steps.

- If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingToken" does not have Type = "tt:RecordingReference", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.7 RECORDING CONTROL - DELETE RECORDING EVENT PROPERTIES

Test Case ID: RECORDING-5-1-23

Specification Coverage: Recording and track creation and deletion (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/DeleteRecording

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/DeleteRecording event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.
- Dynamic Recording is supported by the DUT as indicated by DynamicRecordings = true capability.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.

4. The DUT responds with a **GetEventPropertiesResponse** message with parameters:
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:RecordingConfig/DeleteRecording topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/DeleteRecording topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is equal to true, FAIL the test and skip other steps.
   - If *topic1* does not contain a MessageDescription.Source.SimpleItemDescription item with Name = "RecordingToken", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingToken" does not have Type = "tt:RecordingReference", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.8 RECORDING CONTROL - TRACK CONFIGURATION EVENT PROPERTIES

Test Case ID: RECORDING-5-1-24

Specification Coverage: Configuration changes (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/TrackConfiguration

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/TrackConfiguration event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters:
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:RecordingConfig/TrackConfiguration topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/TrackConfiguration topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is equal to true, FAIL the test and skip other steps.
   - If *topic1* does not contain a MessageDescription.Source.SimpleItemDescription item with Name = "RecordingToken", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingToken" does not have Type = "tt:RecordingReference", FAIL the test and skip other steps.

- If *topic1* does not contain a MessageDescription.Source.SimpleItemDescription item with Name = "TrackToken", FAIL the test and skip other steps.
- If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "TrackToken" does not have Type = "tt:TrackReference", FAIL the test and skip other steps.
- If *topic1* does not contain a MessageDescription.Data.ElementItemDescription item with Name = "Configuration", FAIL the test and skip other steps.
- If *topic1*.MessageDescription.Data.ElementItemDescription with Name = "Configuration" does not have Type = "tt:TrackConfiguration", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.9 RECORDING CONTROL - RECORDING JOB CONFIGURATION EVENT PROPERTIES

Test Case ID: RECORDING-5-1-25

Specification Coverage: Configuration changes (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/RecordingJobConfiguration

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/RecordingJobConfiguration event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain tns1:RecordingConfig/RecordingJobConfiguration topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/RecordingJobConfiguration topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is equal to true, FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "RecordingJobToken", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingJobToken" does not have Type = "tt:RecordingJobReference", FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Data.ElementItemDescription item with Name = "Configuration", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Data.ElementItemDescription with Name = "Configuration" does not have Type = "tt:RecordingJobConfiguration", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.10 RECORDING CONTROL - RECORDING JOB STATE EVENT PROPERTIES

Test Case ID: RECORDING-5-1-26

Specification Coverage: Recording job state changes (ONVIF Recording Control Specification)

Feature Under Test: GetEventProperties, tns1:RecordingConfig/JobState

WSDL Reference: event.wsdl

Test Purpose: To verify tns1:RecordingConfig/JobState event format in TopicSet.

Pre-Requisite:

- Event service is supported by DUT.
- Recording Control service is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list

5. If *topicSet* does not contain tns1:RecordingConfig/JobState topic, FAIL the test and skip other steps.
6. ONVIF Client verifies tns1:RecordingConfig/JobState topic (*topic1*) from *topicSet*:
   - If *topic1*.MessageDescription.IsProperty is skipped or equal to false, FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Source.SimpleItemDescription item with Name = "RecordingJobToken", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Source.SimpleItemDescription with Name = "RecordingJobToken" does not have Type = "tt:RecordingJobReference", FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Data.SimpleItemDescription item with Name = "State", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Data.SimpleItemDescription with Name = "State" does not have Type = "xs:String", FAIL the test and skip other steps.
   - If *topic1* does not contain MessageDescription.Data.ElementItemDescription item with Name = "Information", FAIL the test and skip other steps.
   - If *topic1*.MessageDescription.Data.ElementItemDescription with Name = "Information" does not have Type = "tt:RecordingJobStateInformation", FAIL the test and skip other steps.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.

5.4.11 RECORDING CONTROL – JOB STATE EVENT

Test Case ID: RECORDING-5-1-27

Specification Coverage: Recording job state changes (ONVIF Recording Control Service Specification), Properties (ONVIF Core Specification)

Feature Under Test: GetRecordingJobState

WSDL Reference: event.wsdl, recording.wsdl

Test Purpose: To verify tns1:RecordingConfig/JobState event generation after subscription.

Pre-Requisite:

- Event Service was received from the DUT.
- ONVIF Client gets the entry point for Recording Control Service by GetServices.
- Media Service or Receiver Service was received from the DUT.
- At least one recording exists. All recording jobs were stopped.
- At least one media profile compatible with at least one existing or created recording exists on the DUT.
- Options are supported by the DUT.
- Device supports Pull-Point Notification feature.
- Onboard storage is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingJobsRequest** message to retrieve a complete recording jobs list.
4. If there is at least one RecordingJob, then skip steps 5-10 and go to the step 11.
5. If Receiver service is supported, then ONVIF Client execute Annex A.64 with RequiredSpareJobs = 0 to create or select Recording to make sure that 1 recording job can be created.
6. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = "RecordingToken1", JobConfiguration.Mode = "Idle", JobConfiguration.Priority = 1, no JobConfiguration.Source.SourceToken.Token, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Receiver", JobConfiguration.Source.AutoCreateReceiver = true) to create a recording job and auto create receiver.
7. Verify **CreateRecordingJobResponse** from the DUT and go to step 11.

8. If Receiver service is not supported, ONVIF Client execute Annex A.66 with RequiredSpareJobs = 0 to create or select Recording (RecordingToken = RecordingToken1) to make sure that 1 recording job can be created and to get the compatible media profile tokens list.

9. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = RecordingToken1, JobConfiguration.Mode = Idle, JobConfiguration.Priority = 1, JobConfiguration.Source.SourceToken.Token = "ProfileToken1", where ProfileToken1 is token of MediaProfile from Compatible Sources list, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Profile", JobConfiguration.Source.AutoCreateReceiver is not present) to create a recording job with Idle mode.

10. Verify the **CreateRecordingJobResponse** message from the DUT.

11. ONVIF Client will invoke **CreatePullPointSubscriptionRequest** message with tns1:RecordingConfig/JobState Topic as Filter and an InitialTerminationTime of timeout1.

12. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.

13. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 2.

14. Verify that the DUT sends a **PullMessagesResponse** that contains NotificationMessages. Repeat step 20 until Notifications for all Jobs are received.

15. Verify received Notification messages (correct value for UTC time, TopicExpression and wsnt:Message).

16. Verify that TopicExpression is equal to tns1:RecordingConfig/JobState for all received Notify messages.

17. Verify that each notification contains Source.SimpleItem item with Name = "RecordingJobToken" and Value is equal to one of existing Recording Job Tokens (e.g. complete list of Recording Jobs contains Recording Job with the same token). Verify that there are Notification messages for each Recording Job.

18. Verify that each notification contains Data.SimpleItem item with Name = "State" and Value with type is equal to xs:string.

19. If notification contains Data.ElementItem item with Name = "Information", verify that its Value type is equal to tt:RecordingJobStateInformation (validation with XML Schema complex type).

20. Verify that Notify PropertyOperation = "Initialized".

21. ONVIF Client will invoke **GetRecordingJobStateRequest** message for each Recording Job with corresponding tokens.
22. Verify the **GetRecordingJobStateResponse** messages from the DUT. Verify that Data.ElementItem item with Name = "Information" from Notification message has the same value with State element from corresponding **GetRecordingJobStateResponse** messages for each Recording Job. Verify that Data.SimpleItem item with Name = "State" from Notification message has the same value with State.State element from corresponding **GetRecordingJobStateResponse** messages for each Recording Job.
23. Restore DUT settings.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send a valid SubscriptionReference.
- The DUT did not send a valid **GetRecordingJobsResponse** message.
- The DUT did not send a valid **CreateRecordingJobResponse** message.
- The DUT did not send a valid **GetRecordingJobStateResponse** message
- The DUT did not send a Notification message that contains a property event tns1:RecordingConfig/JobState at least for one Recording Job.
- The DUT sent an invalid Notification message (no corresponding Source.SimpleItem, Data.SimpleItem, or Data.ElementItem wrong type of Value fields, invalid RecordingJobToken, State, or Information values, PropertyOperation is not equal to "Initialized").

Note: The Subscription Manager has to be deleted at the end of the test either by calling unsubscribe or through a timeout.

Note: ONVIF Client at step 20 will wait for Notification messages until notification for all Recording Job is received or Operation Delay after last notification expires.

Note: The Renew has to be used for renew subscription during the test, if InitialTerminationTime expires. If DUT returns UnacceptableTerminationTimeFault, resend Renew request with acceptable InitialTerminationTime from UnacceptableTerminationTimeFault.

Note: If DUT cannot accept the set value to Timeout or MessageLimit, ONVIF Client retries to send the PullMessage message with Timeout and MessageLimit which is contained in PullMessagesFaultResponse.

Note: timeout1 will be taken from Subscription Timeout field of ONVIF Device Test Tool.

Annex A Helper Procedures and Additional Notes

A.1 Object Storage Recording Prerequisite

Name: HelperMedia2ExternalStorageRecordingPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare recording for object storage recording.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Input:

- Object storage type (*objectStorageType*) (optional).
- Object storage configuration (*objectStorageConfiguration*) (optional). Could contains no CertPathValidationPolicyID for object storage. In this case it will be created automatically by the procedure.
- Recording encryption (*encryption*) (optional).
- Recording segment duration (*recordingSegmentDuration*) (optional, if skipped default value is **PT20S**).
- **Note:** if *objectStorageConfiguration* is specified, *objectStorageType* will be ignored.

Returns:

- Recording token (*recordingToken*).
- Recording configuration (*recordingConfiguration*).
- Recording job token (*recordingJobToken*).

- Recording job configuration (*recordingJobConfiguration*).
- Recording segment duration (*recordingSegmentDuration*).
- External storage configuration identifier (*externalStorageId*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
2. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.
3. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := AudioEncoder
   - Type[1] := VideoEncoder
4. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
5. Set *listOfSuitableProfiles* := tokens of items of *profileList* which contains VideoEncoder or AudioEncoder with encoding listed in *cap*.Encoding.
6. ONVIF Client invokes **GetRecordings** request.
7. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
8. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:
   - in *objectStorageType* - object storage type
   - in *objectStorageConfiguration* - storage configuration
   - out *externalStorage1* - external storage configuration identifier
9. Set:
   - *target1*.Storage := *externalStorage1*

- *target1*.Format := *cap*.SupportedTargetFormats[0]
- *target1*.SegmentDuration := *recordingSegmentDuration*
- *target1*.Prefix is skipped
- *target1*.Postfix is skipped
- *target1*.SpanDuration is skipped
- *target1*.Encryption := *encryption* if defined, skipped otherwise
- *target1*.SegmentDurationOverride is skipped

10. If *cap*.DynamicRecording = true:

10.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 10.5.

10.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:

- RecordingToken := *recordingsList1*[0].RecordingToken

10.3. The DUT responds with **DeleteRecordingResponse** message.

10.4. Set:

- *recordingConfiguration1*.Source.Name := "Source1"
- *recordingConfiguration1*.Source.SourceId := [valid URI]
- *recordingConfiguration1*.Source.Address := [valid URI]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

10.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

10.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

10.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

10.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

10.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

10.10 If *recordingOptions1*.Job.CompatibleSources contains no profiles tokens from *listOfSuitableProfiles*, FAIL the test, restore DUT settings and skip other steps.

10.11 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11. If *cap*.DynamicRecording = false:

11.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

11.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

11.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

11.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

11.1.4 If *recordingOptions1*.Job.CompatibleSources contains at least one profile tokens from *listOfSuitableProfiles*:

11.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

11.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

11.1.4.3 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11.1.4.4Go to the step 11.3.

11.2. If no recording was found, FAIL the test, restore DUT settings and skip other steps.

11.3. Set:

- *recordingConfiguration*.Target := *target1*

11.4. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

11.5. The DUT responds with **SetRecordingConfigurationResponse** message.

12. Set:

- *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
- *source1*.SourceToken.Token = *sourceToken*
- *source1*.AutoCreateReceiver is skipped
- *source1*.Tracks is skipped
- *source1*.Extension is skipped

13. Set:

- *recordingJobConfiguration1*.ScheduleToken is skipped
- *recordingJobConfiguration1*.RecordingToken := *recordingToken*
- *recordingJobConfiguration1*.Mode := **Idle**
- *recordingJobConfiguration1*.Priority := 0
- *recordingJobConfiguration1*.Source[0] := *source1*
- *recordingJobConfiguration1*.Extension is skipped
- *recordingJobConfiguration1*.EventFilter is skipped

14. ONVIF Client invokes **CreateRecordingJob** request with parameters:

- JobConfiguration := *recordingJobConfiguration1*

15. The DUT responds with **CreateRecordingJobResponse** message with parameters:

- JobToken =: *recordingJobToken*
- JobConfiguration =: *recordingJobConfiguration*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.2 Delete all Recording Jobs

Name: HelperDeleteAllRecordingJobs

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to delete all recording jobs.

Pre-Requisite:

- Recording Control Service is received from the DUT.

Input: None

Returns: None

Procedure:

1. ONVIF Client invokes **GetRecordingJobs** request.
2. The DUT responds with **GetRecordingJobsResponse** message with parameters
   - JobItem list =: *recordingJobsList1*
3. For each recording job *recordingJobsItem1* in *recordingJobsList1* repeat the following steps:

   3.1. ONVIF Client invokes **DeleteRecordingJob** request with the following parameters:

- JobToken := *recordingJobsItem1*.JobToken

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

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetRecordingJobsResponse** message.
- DUT did not send **DeleteRecordingJobResponse** message.

A.3 Get Recording Control Service Capabilities

Name: HelperGetServiceCapabilities_Recording

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

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

Pre-Requisite: Recording Control Service is received from the DUT.

Input: None

Returns: The service capabilities (*cap*).

Procedure:

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

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

A.4 Object Storage Configuration Creation For Recording Tests

Name: HelperStorageCreationForRecordingTests

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create object storage configuration for recording test cases.

Pre-Requisite:

- Storage configuration is supported by the DUT as indicated by StorageConfiguration capability.
- Security Configuration Service is received from the DUT.
- Certification path validation policy is supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability.
- UploadCertificate is supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA or PKCS10 capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- The DUT shall have enough free storage capacity for one additional certification path validation policy.
- The DUT shall have enough free storage capacity for one additional certification path.
- The DUT shall have enough free storage capacity for one additional certificate.
- The DUT shall have enough free storage capacity for one additional key pair.

Input:

- Object storage type (*objectStorageType*) (optional).
- Object storage configuration (*objectStorageConfiguration*) (optional). Could contains no CertPathValidationPolicyID for object storage. In this case it will be created automatically by the procedure.

- **Note:** if *objectStorageConfiguration* is specified, *objectStorageType* will be ignored.

Returns:

- External storage configuration token (*externalStorage1*).

Procedure:

1. If *objectStorageConfiguration* were defined, go to step 6.
2. If *objectStorageType* were defined, go to step 5.
3. ONVIF Client gets the device management service capabilities by following the procedure mentioned in Annex A.5 with the following input and output parameters:
   - out *capDevice* - Device Management Service Capabilities
4. Set *objectStorageType* := **ObjectStorageS3**, if *capDevice*.System.StorageTypesSupported contains **ObjectStorageS3**, otherwise **ObjectStorageAzure**.
5. Set:
   - *objectStorageConfiguration*.type := *objectStorageType*
   - *objectStorageConfiguration*.Region := [region provided by user for *objectStorageType*]
   - *objectStorageConfiguration*.LocalPath is skipped
   - *objectStorageConfiguration*.StorageUri := [storage uri provided by user for *objectStorageType*]
   - *objectStorageConfiguration*.User.UserName := [user name provided by user for *objectStorageType*]
   - *objectStorageConfiguration*.User.Password := [password provided by user for *objectStorageType*]
   - *objectStorageConfiguration*.User.Extension is skipped
   - *objectStorageConfiguration*.User.Token := [token provided by user for *objectStorageType*]
   - *objectStorageConfiguration*.Extension is skipped
   - *objectStorageConfiguration*.ConfigurationRenewal is skipped
6. If *objectStorageConfiguration*.CertPathValidationPolicyID were defined, go to step 11.
7. ONVIF Client invokes **UploadCertificate** with parameters

- Certificate := [certificate provided for certification path validation policy for external storage by user]
- Alias := "ONVIF Test Cert"
- PrivateKeyRequired := false

8. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*
9. ONVIF Client creates certification path validation policy identifier with specified alias and the certificate identifier for trust anchor by following the procedure mentioned in Annex A.6 with the following input and output parameters:
   - in *certID* - certificate identifier for trust anchor
   - in "ONVIF Test Cert Val Policy" - certification path validation policy alias
   - out *certPathValidationPolicyID* - certification path validation policy identifier
10. Set:
    - *objectStorageConfiguration*.CertPathValidationPolicyID := *certPathValidationPolicyID*
11. ONVIF Client invokes **CreateStorageConfiguration** with parameters
    - StorageConfiguration := *objectStorageConfiguration*
12. The DUT responds with a **CreateStorageConfigurationResponse** message with parameters:
    - Token =: *externalStorage1*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **UploadCertificateResponse** message.
- DUT did not send **CreateStorageConfigurationResponse** message.

A.5 Get Service Capabilities (Device Management)

Name: HelperGetServiceCapabilities

Procedure Purpose: Helper procedure to retrieve Device Management Service Capabilities.

Pre-Requisite: GetServices command is supported by the DUT.

Input: None.

Returns: Device Management Service Capabilities (*cap*).

Procedure:

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.6 Create a certification path validation policy with provided certificate identifier

Name: HelperCreateCertPathValidationPolicyWithCertID

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create a certification path validation policy with provided certificate identifier.

Pre-Requisite: Security Configuration Service is received from the DUT. Certification path validation policy is supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. The DUT shall have enough free storage capacity for one additional certification path validation policy.

Input: The certification path validation policy alias (*alias*) and the certificate identifier (*certID*) for trust anchor.

Returns: The certification path validation policy identifier (*certPathValidationPolicyID*).

Procedure:

1. ONVIF Client invokes **CreateCertPathValidationPolicy** with parameters
   - Alias := *alias*
   - Parameters.RequireTLSWWWClientAuthExtendedKeyUsage skipped
   - Parameters.UseDeltaCRLs = true
   - Parameters.anyParameters skipped
   - TrustAnchor[0].CertificateID := *certID*
   - anyParameters skipped
2. The DUT responds with **CreateCertPathValidationPolicyResponse** message with parameters:
   - CertPathValidationPolicyID =: *certPathValidationPolicyID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.7 Retrieve Recording Job State Event by PullPoint

Name: HelperPullJobState

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/JobState** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*currentTime*).
- Subscription termination time (*terminationTime*).
- Recording Job token (*recordingJobToken*).
- Expected Job State (*expectedJobState*).

Returns:

- Job State (*jobState*).

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*terminationTime-currentTime*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime =: *currentTime*
      - TerminationTime =: *terminationTime*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains an item with Topic = "tns1:RecordingConfig/JobState" and Message.Source.SimpleItem with Name = "RecordingJobToken" and Value = *recordingJobToken* and Message.Data.SimpleItem with Name = "State" and Value = *expectedJobState*:
      1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* with Topic = "tns1:RecordingConfig/JobState" and

Message.Source.SimpleItem with Name = "RecordingJobToken" and Value = *recordingJobToken* and Message.Data.SimpleItem with Name = "State" and Value = *expectedJobState*].

1.4.2. Go to step 3.

2. If *operationDelay* timeout expires for step 1 without at least one of expected notifications, FAIL the test and skip other steps.
3. If *notificationMessage*.Message.Data does not contain ElementItem with Name = "Information" and Value of tt:RecordingJobStateInformation type, FAIL the test, restore the DUT state, and skip other steps.
4. Set *jobState* := [value of Value attribute from *notificationMessage*.Message.Data.ElementItem[Name = "Information"]].

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.8 Create Pull Point Subscription

Name: HelperCreatePullPointSubscription

Notes: Annex is used at:

- ONVIF Base Device Test Specification
- ONVIF Device IO Device Test Specification
- ONVIF Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to create PullPoint Subscription with specified Topic.

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

Input: Notification Topic (*topic*).

Returns: Subscription reference (*s*), current time for the DUT (*currentTime*), subscription termination time (*terminationTime*).

Procedure:

1. ONVIF Client invokes **CreatePullPointSubscription** request with parameters:
   - Filter.TopicExpression := *topic*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
2. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters:
   - SubscriptionReference =: *s*
   - CurrentTime =: *currentTime*
   - TerminationTime =: *terminationTime*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.9 Object Storage Recording With Different Priorities Prerequisite

Name: HelperMedia2ExternalStorageRecordingWithPrioritiesPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare recording for object storage recording.

Pre-Requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.

- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Input:

- Recording prefix (*prefix*).
- Recording postfix (*postfix*).

Returns:

- Recording token (*recordingToken*).
- Recording configuration (*recordingConfiguration*).
- Spare jobs (*spareJobs*).
- Recording job source (*source*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities
2. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.
3. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := AudioEncoder
   - Type[1] := VideoEncoder
4. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
5. Set *listOfSuitableProfiles* := tokens of items of *profileList* which contains VideoEncoder or AudioEncoder with encoding listed in *cap*.Encoding.
6. ONVIF Client invokes **GetRecordings** request.
7. The DUT responds with **GetRecordingsResponse** message with parameters

- RecordingItem list =: *recordingsList1*

8. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:
   - out *externalStorage1* - external storage configuration identifier
9. Set:
   - *target1*.Storage := *externalStorage1*
   - *target1*.Format := *cap*.SupportedTargetFormats[0]
   - *target1*.SegmentDuration := **PT20S**
   - *target1*.Prefix := *prefix*
   - *target1*.Postfix := *postfix*
   - *target1*.SpanDuration is skipped
   - *target1*.Encryption is skipped
   - *target1*.SegmentDurationOverride is skipped
10. If *cap*.DynamicRecording = true:

    10.1. If number of items at *recordingsList1* is less than *cap*.MaxRecordings, go to step 10.5.

    10.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:
    - RecordingToken := *recordingsList1*[0].RecordingToken

    10.3. The DUT responds with **DeleteRecordingResponse** message.

    10.4. Set:
    - *recordingConfiguration1*.Source.Name := "Source1"
    - *recordingConfiguration1*.Source.SourceId := [valid URI]
    - *recordingConfiguration1*.Source.Address := [valid URI]
    - *recordingConfiguration1*.Source.Location := "Source Location"
    - *recordingConfiguration1*.Source.Description := "Source Description"
    - *recordingConfiguration1*.Content := "Content"

- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

10.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

10.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

10.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

10.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

10.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

10.10 Set *spareJobs* := *recordingOptions1*.Job.Spare.

10.11 If *recordingOptions1*.Job.CompatibleSources contains at least one profile token from *listOfSuitableProfiles*, continue test. Otherwise FAIL the test, restore DUT settings and skip other steps.

10.12 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11. If *cap*.DynamicRecording = false:

11.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

11.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

11.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

11.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

11.1.4 If *recordingOptions1*.Job.CompatibleSources contains at least one profile token from *listOfSuitableProfiles*:

11.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

11.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

11.1.4.3 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11.1.4.4 Set *spareJobs* := *recordingOptions1*.Job.Spare.

11.1.4.5 If *spareJobs* > 1:

11.1.4.5.1 Go to the step 11.3.

11.2. If no recording for which media2 profile recording could be started was found (*recordingToken* was not set at step 11.1.4.1), FAIL the test, restore DUT settings and skip other steps.

11.3. Set:

- *recordingConfiguration*.Target := *target1*

11.4. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

11.5. The DUT responds with **SetRecordingConfigurationResponse** message.

12. Set:

- *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
- *source1*.SourceToken.Token = *sourceToken*
- *source1*.AutoCreateReceiver is skipped
- *source1*.Tracks is skipped
- *source1*.Extension is skipped

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.10 Object Storage Recording With No Storage Access Prerequisite

Name: HelperMedia2ExternalStorageRecordingWithStorageErrorPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare recording for object storage recording with no storage access.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.

Input: None

Returns:

- Recording job token (*recordingJobToken*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities

2. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.
3. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := AudioEncoder
   - Type[1] := VideoEncoder
4. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
5. Set *listOfSuitableProfiles* := tokens of items of *profileList* which contains VideoEncoder or AudioEncoder with encoding listed in *cap*.Encoding.
6. ONVIF Client invokes **GetRecordings** request.
7. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
8. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.11 with the following input and output parameters:
   - out *externalStorage1* - external storage configuration identifier
9. Set:
   - *target1*.Storage := *externalStorage1*
   - *target1*.Format := *cap*.SupportedTargetFormats[0]
   - *target1*.SegmentDuration := **PT20S**
   - *target1*.Prefix is skipped
   - *target1*.Postfix is skipped
   - *target1*.SpanDuration is skipped
   - *target1*.Encryption is skipped
   - *target1*.SegmentDurationOverride is skipped
10. If *cap*.DynamicRecording = true:

    10.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 10.5.

10.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:

- RecordingToken := *recordingsList1*[0].RecordingToken

10.3. The DUT responds with **DeleteRecordingResponse** message.

10.4. Set:

- *recordingConfiguration1*.Source.Name := "Source1"
- *recordingConfiguration1*.Source.SourceId := [valid URI]
- *recordingConfiguration1*.Source.Address := [valid URI]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

10.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

10.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

10.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

10.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

10.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore the DUT settings and skip other steps.

10.10 If *recordingOptions1*.Job.CompatibleSources contains no profile tokens from *listOfSuitableProfiles*, FAIL the test, restore the DUT settings and skip other steps.

10.11.3 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11. If *cap*.DynamicRecording = false:

11.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

11.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

11.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

11.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore the DUT settings and skip other steps.

11.1.4 If *recordingOptions1*.Job.CompatibleSources contains at least one profile tokens from *listOfSuitableProfiles*:

11.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

11.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

11.1.4.3 Set *sourceToken* := item from *recordingOptions1*.Job.CompatibleSources listed at *listOfSuitableProfiles*.

11.1.4.4 Go to the step 11.3.

11.2. If no recording was found, FAIL the test, restore DUT settings and skip other steps.

11.3. Set:

- *recordingConfiguration*.Target := *target1*

11.4. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

11.5. The DUT responds with **SetRecordingConfigurationResponse** message.

12. Set:

- *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
- *source1*.SourceToken.Token = *sourceToken*
- *source1*.AutoCreateReceiver is skipped
- *source1*.Tracks is skipped
- *source1*.Extension is skipped

13. Set:
    - *recordingJobConfiguration1*.ScheduleToken is skipped
    - *recordingJobConfiguration1*.RecordingToken := *recordingToken*
    - *recordingJobConfiguration1*.Mode := **Active**
    - *recordingJobConfiguration1*.Priority := 0
    - *recordingJobConfiguration1*.Source[0] := *source1*
    - *recordingJobConfiguration1*.Extension is skipped
    - *recordingJobConfiguration1*.EventFilter is skipped
14. ONVIF Client invokes **CreateRecordingJob** request with parameters:
    - JobConfiguration := *recordingJobConfiguration1*
15. The DUT responds with **CreateRecordingJobResponse** message with parameters:
    - JobToken =: *recordingJobToken*
    - JobConfiguration =: *recordingJobConfiguration*
16. Wait for *operationDelay*.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

- DUT did not send **DeleteRecordingResponse** message at least once.

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

A.11 Object Storage Configuration Creation For Recording Configuration Tests

Name: HelperStorageCreationForRecordingConfigurationTests

Procedure Purpose: Helper procedure to create object storage configuration for recording configuration test cases.

Pre-requisite:

- Storage configuration is supported by the DUT as indicated by StorageConfiguration capability.
- Security Configuration Service is received from the DUT.
- Certification path validation policy is supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability.
- UploadCertificate is supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA or PKCS10 capability.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- The DUT shall have enough free storage capacity for one additional certification path validation policy.
- The DUT shall have enough free storage capacity for one additional certification path.
- The DUT shall have enough free storage capacity for one additional certificate.
- The DUT shall have enough free storage capacity for one additional key pair.

Input: None

Returns: External storage configuration token (*externalStorage1*).

Procedure:

1. ONVIF Client gets the device service capabilities by following the procedure mentioned in Annex A.12 with the following input and output parameters:
   - out *capDevice* - Device Service Capabilities

2. ONVIF Client gets the security configuration service capabilities by following the procedure mentioned in Annex A.13 with the following input and output parameters:
   - out *capSecurity* - Security Configuration Service Capabilities
3. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *capSecurity* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
4. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.14 with the following input and output parameters:
   - in *capSecurity* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - in "CN=ONVIF TT Storage,C=US" - CA certificate subject
   - in "Test CertPathValidationPolicy Storage Alias" - certification path validation policy alias
   - out *certPathValidPolID1* - certification path validation policy identifier
   - out *certID1* - certificate identifier
   - out *keyID1* - key pair identifier
   - out *CACert1* - CA certificate
   - out *privateKeyCACert1* - CA certificate private key
5. ONVIF Client invokes **CreateStorageConfiguration** request with parameters
   - type := **ObjectStorageS3**, if *capDevice*.System.StorageTypesSupported contains **ObjectStorageS3**, otherwise **ObjectStorageAzure**
   - Region := "us-east-1"
   - LocalPath := "test1/test1"
   - StorageUri := "s3://some-bucket1", if **ObjectStorageS3** was selected, otherwise "abfs://some-container1@user.dfs.core.windows.net"
   - User.UserName := "User1"
   - User.Password := "Password1"

- User.Extension is skipped
- User.Token = "UserToken1"
- Extension is skipped
- CertPathValidationPolicyID := *certPathValidPolID1*
- ConfigurationRenewal is skipped

6. The DUT responds with a **CreateStorageConfigurationResponse** message with parameters:
   - Token =: *externalStorage1*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

A.12 Get Capabilities (Device Management)

Name: HelperGetDeviceCapabilities

Procedure Purpose: Helper procedure to retrieve Device Management Capabilities.

Pre-requisite: GetCapabilities command is supported by the DUT.

Input: None.

Returns: Device Management Capabilities (*deviceManagementCap*).

Procedure:

1. ONVIF Client invokes **GetCapabilities** request.
2. The DUT responds with **GetCapabilitiesResponse** message with parameters:
   - Capabilities =: *cap*
3. Set *deviceManagementCap* := *cap*.Device.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.13 Get service capabilities for Advanced Security service

Name: HelperGetServiceCapabilities_AdvancedSecurity

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Media2 Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to get service capabilities.

Pre-requisite: Security Configuration Service is received from the DUT.

Input: None

Returns: The service capabilities (*cap*).

Procedure:

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.14 Create a certification path validation policy for external storage configuration

Name: HelperCreateCertPathValidationPolicyForExternalStorage

Notes: Annex is used at:

- ONVIF Base Test Specification
- ONVIF Recording Control Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to create a certification path validation policy for external storage configuration.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- Certification path validation policy is supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability.
- UploadCertificate is supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA or PKCS10 capability.
- The DUT shall have enough free storage capacity for one additional certification path validation policy.
- The DUT shall have enough free storage capacity for one additional certification path.
- The DUT shall have enough free storage capacity for one additional certificate.
- The DUT shall have enough free storage capacity for one additional key pair.

Input:

- The service capabilities (*cap*).
- The key pair algorithm (*keyAlgorithm*).
- The certification path validation policy alias (*certPathValidationPolicyAlias*).
- The subject (*subject*) of CA certificate (optional input parameter, could be skipped).

Returns:

- Certification path validation policy identifier (*certPathValidationPolicyID*).
- Certificate used in certification path validation policy (*certID*).
- Key pair of the certificate (*keyID*).
- CA certificate (*CAcert*).
- CA certificate private key (*privateKey*).

Procedure:

1. ONVIF Client creates a CA certificate and a corresponding private key by following the procedure described in Annex A.15 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *CAcert* - CA certificate
   - out *privateKey* - private key
2. ONVIF Client invokes **UploadCertificate** request with parameters:
   - Certificate := *CAcert*
   - Alias := "ONVIF Test"
   - PrivateKeyRequired := false
3. The DUT responds with an **UploadCertificateResponse** message with parameters:
   - CertificateID =: *certID*
   - KeyID =: *keyID*
4. ONVIF Client creates certification path validation policy identifier with specified alias and the certificate identifier for trust anchor by following the procedure mentioned in Annex A.6 with the following input and output parameters:
   - in *certID* - certificate identifier for trust anchor
   - in *certPathValidationPolicyAlias* - certification path validation policy alias
   - out *certPathValidationPolicyID* - certification path validation policy identifier

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.15 Provide CA certificate

Name: HelperCreateCACertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to create an X.509 CA certificate.

Pre-requisite: None.

Input: The subject (*subject*) of certificate (optional input parameter, could be skipped). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: An X.509 CA certificate (*CAcert*) that is compliant to [RFC 5280] and a corresponding key pair (*keyPair*) with private key and public key.

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.16 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
2. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm

- out *keyPair* - key pair

3. If *subject* is skipped set:
   - *subject* := "CN=ONVIF TT,C=US"
4. ONVIF Client creates an X.509 self-signed CA certificate that is compliant to [RFC 5280] and has the following properties:
   - version := v3
   - signature := *signatureAlgorithm*
   - validity := not before 19700101000000Z and not after 99991231235959Z
   - subject := *subject*
   - public key := *keyPair* .publicKey
   - private key to be used := *keyPair*.privateKey

Note: ONVIF Client may return the same CA certificate in subsequent invocations of this procedure for the same subject.

A.16 Signature Algorithm Selection

Name: HelperSignatureAlgorithmSelection

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to select signature algorithm which will be used for tests based on Device capabilities.

Pre-requisite: Security Configuration Service is received from the DUT.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The signature algorithm (*signatureAlgorithm*).

Procedure:

1. If *keyAlgorithm* = RSA: ONVIF Client selects signature algorithm (*signatureAlgorithm*) that will be used for the test from the list provided by DUT at *cap*.KeystoreCapabilities.SignatureAlgorithms. Selection is done among the following list of signature algorithms supported by the Client by priority from first to last:
   - 1.2.840.113549.1.1.13 (OID of SHA-512 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.12 (OID of SHA-384 with RSA Encryption algorithm)
   - 1.2.840.113549.1.1.11 (OID of SHA-256 with RSA Encryption algorithm)
2. If *keyAlgorithm* = ECC: ONVIF Client selects signature algorithm (*signatureAlgorithm*) that will be used for the test from the list provided by DUT at *cap*.KeystoreCapabilities.SignatureAlgorithms. Selection is done among the following list of signature algorithms supported by the Client by priority from first to last:
   - 1.2.840.10045.4.3.4 (OID of SHA-512 with ECC Encryption algorithm)
   - 1.2.840.10045.4.3.3 (OID of SHA-384 with ECC Encryption algorithm)
   - 1.2.840.10045.4.3.2 (OID of SHA-256 with ECC Encryption algorithm)
3. If the previous steps is done with empty signature algorithm (*signatureAlgorithm*): FAIL the procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not return any of signature algorithms listed at step 1 or 2.

A.17 Generate a key pair

Name: HelperGenerateKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to generate a key pair.

Pre-requisite: None.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: A [RFC 3447] compliant RSA or [RFC 5480, RFC 5915] compliant ECC key pair (*keyPair*) with new public key and private key.

Procedure:

1. ONVIF Client determines the key pair generation parameters by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation parameters
2. If *keyGenParams*.algorithm = RSA:
   a. Create an [RFC 3447] compliant RSA key pair (out *keyPair*) with new public key and private key with the following properties:
      - KeyLength := *keyGenParams*.keyLength
3. If *keyGenParams*.algorithm = ECC:
   a. Create an [RFC 5480, RFC 5915] compliant ECC key pair (out *keyPair*) with new public key and private key with the following properties:
      - EllipticCurve := *keyGenParams*.ellipticCurve

A.18 Determine key pair generation parameters

Name: HelperDetermineKeyPairGenerationParams

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification

- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to determine the key pair generation parameters to use during testing.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- On-board ECC or RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*).

Returns: The key pair generation parameters (*keyGenParams*).

Procedure:

1. If *keyAlgorithm* = RSA:
   a. ONVIF Client loops through the supported Key length list (*cap*.KeystoreCapabilities .RSAKeyLengths) and selects the smallest supported key length (*keyLength*).
   b. ONVIF Client creates the RSA key generation parameters (*keyGenParams*) with the key length (*keyLength*).
2. If *keyAlgorithm* = ECC:
   a. ONVIF Client loops through the supported elliptic curves list (*cap*.KeystoreCapabilities.EllipticCurves) and selects the simplest elliptic curve (*ellipticCurve*).
   b. ONVIF Client creates the ECC key generation parameters (*keyGenParams*) with the elliptic curve (*ellipticCurve*).
3. If previous steps is done with empty key pair generation parameters (*keyGenParams*): FAIL the procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- No supported RSA key length was found at step 1.1 or no supported ECC elliptic curves was found at step 2.1.

A.19 Selection of key pair algorithm

Name: HelperKeyAlgorithmSelection

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to select key pair algorithm for the test depending on device capabilities.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- On-board key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability.

Input: Advanced security capabilities (*cap*).

Returns: key pair algorithm (*keyAlgorithm*).

Procedure:

1. If *cap*.KeystoreCapabilities.ECCKeyPairGeneration = true or *cap*.KeystoreCapabilities.KeyPairGeneration contains **ECC**:
   - Set *keyAlgorithm* := **ECC**.
   - Skip other steps of procedure and return to the test.
2. If *cap*.KeystoreCapabilities.RSAKeyPairGeneration = true or *cap*.KeystoreCapabilities.KeyPairGeneration contains **RSA**:

- Set *keyAlgorithm* := **RSA**.
- Skip other steps of procedure and return to the test.

3. FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT does not pass all assertions.

A.20 Video Recording to Object Storage Recording Prerequisite

Name: HelperVideoRecordingExternalStoragePrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare video recording for object storage recording.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.
- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of video is supported by the DUT as indicated by Encoding capability which contains video encoding.

Input:

- Required video encoding (*videoEncoding*).
- Required format type (*formatType*).

Returns:

- Recording job token (*recordingJobToken*).
- Recording span duration (*recordingSpanDuration*).
- Recording segment duration (*recordingSegmentDuration*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
2. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.
3. ONVIF Client selects a Media Profile with required video encoding support by following the procedure mentioned in Annex A.21 with the following input and output parameters
   - in *videoEncoding* - required video encoding
   - out *profile* - Media Profile with Video Source Configuration and Video Encoder Configuration with the required video encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list (or skipped if *vecOptions*.@GovLengthRange skipped)
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline (or skipped if *vecOptions*.@ProfilesSupported skipped)
   - Configuration.Encoding := *requiredVideoEncoding*
   - Configuration.Resolution := resolution closest to 640x480 from *vecOptions*.ResolutionsAvailable list

- if *vecOptions*.@FrameRatesSupported skipped and *profile*.Configurations.VideoEncoder.RateControl skipped:
  - Configuration.RateControl skipped
- if *vecOptions*.@FrameRatesSupported or *profile*.Configurations.VideoEncoder.RateControl is not skipped:
  - Configuration.RateControl.@ConstantBitRate := *vecOptions*.@ConstantBitRateSupported
  - Configuration.RateControl.FrameRateLimit := value closest to 25 but greater than 1 from *vecOptions*.@FrameRatesSupported list (or *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit if *vecOptions*.@FrameRatesSupported skipped)
  - Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl.BitrateLimit, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}
- Configuration.Quality := *vecOptions*.QualityRange.Min

5. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
6. Set *confTypeList* := (AudioSource, AudioEncoder, AudioOutput, AudioDecoder, Metadata, Analytics, PTZ)
7. ONVIF Client removes all configurations except VideoSource and VideoEncoder from the Media Profile by following the procedure mentioned in Annex A.22 with the following input and output parameters
   - in *confTypeList* - list of configuration types to remove from Media Profile
   - in *profile* - Media Profile to update
8. ONVIF Client invokes **GetRecordings** request.
9. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
10. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:
    - out *externalStorage1* - external storage configuration identifier
11. Set *recordingSegmentDuration* := **PT20S**.

12. Set *recordingSpanDuration* := **PT60S**.

13. Set:

- *target1*.Storage := *externalStorage1*
- *target1*.Format := *formatType*
- *target1*.SpanDuration := *recordingSpanDuration*
- *target1*.SegmentDuration := *recordingSegmentDuration*
- *target1*.Prefix is skipped
- *target1*.Postfix is skipped
- *target1*.Encryption is skipped
- *target1*.SegmentDurationOverride is skipped

14. If *cap*.DynamicRecording = true:

14.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 14.5.

14.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:

- RecordingToken := *recordingsList1*[0].RecordingToken

14.3. The DUT responds with **DeleteRecordingResponse** message.

14.4. Set:

- *recordingConfiguration1*.Source.Name := "Source1"
- *recordingConfiguration1*.Source.SourceId := [valid URI]
- *recordingConfiguration1*.Source.Address := [valid URI]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

14.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

14.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

14.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

14.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

14.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

14.10 If *recordingOptions1*.Job.CompatibleSources contains no *profile*.@token, FAIL the test, restore DUT settings and skip other steps.

14.11 Set *sourceToken* := *profile*.@token.

15. If *cap*.DynamicRecording = false:

15.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

15.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

15.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

15.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

15.1.4 If *recordingOptions1*.Job.CompatibleSources contains *profile*.@token:

15.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

15.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

15.1.4.3 Set *sourceToken* := *profile*.@token.

15.1.4.4 Go to the step 15.3.

15.2. If no recording was found, FAIL the test, restore DUT settings and skip other steps.

15.3. Set:

- *recordingConfiguration*.Target := *target1*

15.4. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

15.5. The DUT responds with **SetRecordingConfigurationResponse** message.

16. Set:

- *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
- *source1*.SourceToken.Token = *sourceToken*
- *source1*.AutoCreateReceiver is skipped
- *source1*.Tracks is skipped
- *source1*.Extension is skipped

17. Set:

- *recordingJobConfiguration1*.ScheduleToken is skipped
- *recordingJobConfiguration1*.RecordingToken := *recordingToken*
- *recordingJobConfiguration1*.Mode := **Idle**
- *recordingJobConfiguration1*.Priority := 0
- *recordingJobConfiguration1*.Source[0] := *source1*
- *recordingJobConfiguration1*.Extension is skipped
- *recordingJobConfiguration1*.EventFilter is skipped

18. ONVIF Client invokes **CreateRecordingJob** request with parameters:

- JobConfiguration := *recordingJobConfiguration1*

19. The DUT responds with **CreateRecordingJobResponse** message with parameters:

- JobToken =: *recordingJobToken*

- JobConfiguration =: *recordingJobConfiguration*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.21 Media2 Service Profile Configuration for Video Streaming

Name: HelperFindMediaProfileForVideoStreaming

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification

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

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

Input:

- Required video encoding (*requiredVideoEncoding*)
- Signed video flag (*signedVideoFlag*, false by default).

Returns:

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

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped

- Type[0] := VideoSource
- Type[1] := VideoEncoder

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

4.1.2. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters

- Configurations list =: *videoEncoderConfList*

4.1.3. For each Video Encoder Configuration *videoEncoderConfiguration1* in *videoEncoderConfList* repeat the following steps:

4.1.3.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *videoEncoderConfiguration1*.@token
- ProfileToken := *profile1*.@token

4.1.3.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

4.1.3.3. If *optionsList* list contains an item with Encoding = *requiredVideoEncoding* and, if *signedVideoFlag* = **true**, with SigningSupported = **true**:

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

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

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

4.1.3.3.3. Set *profile* := *profile1*.

4.1.3.3.4. Set *vecOptions* := item with Encoding = *requiredVideoEncoding* and, if *signedVideoFlag* = **true**, with SigningSupported = **true** from *optionsList* list.

4.1.3.3.5. Skip other steps in procedure.

5. Set *profile1* := *profileList*[0]
6. Set *confTypeList* := (configurations that are contained in profile *profile1*)
7. ONVIF Client removes all configurations from the Media Profile by following the procedure mentioned in Annex A.22 with the following input and output parameters
   - in *confTypeList* - list of configuration type to remove from Media Profile
   - in *profile1* - Media Profile to update
8. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
   - ConfigurationToken skipped
   - ProfileToken := *profile1*.@token
9. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
   - Configurations list =: *videoSourceConfList*
10. For each Video Source Configuration *videoSourceConfiguration1* in *videoSourceConfList* repeat the following steps:
    10.1. ONVIF Client invokes **AddConfiguration** request with parameters
      - ProfileToken := *profile1*.@token
      - Name skipped
      - Configuration[0].Type := VideoSource
      - Configuration[0].Token := *videoSourceConfiguration1*.@token

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

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

    10.4. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters
      - Configurations list =: *videoEncoderConfList*

10.5. For each Video Encoder Configuration *videoEncoderConfiguration1* in *videoEncoderConfList* repeat the following steps:

10.5.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *videoEncoderConfiguration1*.@token
- ProfileToken := *profile1*.@token

10.5.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

10.5.3. If *optionsList* list contains an item with Encoding = *requiredVideoEncoding* and, if *signedVideoFlag* = **true**, with SigningSupported = **true**:

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

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

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

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

10.5.3.4. Set *vecOptions* := item with Encoding = *requiredVideoEncoding* and, if *signedVideoFlag* = **true**, with SigningSupported = **true** from *optionsList* list.

10.5.3.5. Skip other steps in procedure.

11. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **GetProfilesResponse** message.
- DUT did not send **GetVideoEncoderConfigurationOptionsResponse** message.
- DUT did not send **GetVideoEncoderConfigurationsResponse** message.
- DUT did not send **AddConfigurationResponse** message.
- DUT did not send **GetVideoSourceConfigurationsResponse** message.

A.22 Removing Configurations from Media Profile

Name: HelperRemoveConfigurationsFromMediaProfile

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper Procedure to remove configuartions from Media Profile.

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

Input: Media Profile (*profile*). List of configuration types to remove from profile (*confTypeList*).

Returns: None.

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token := *profile*.@token
   - Type[0] := All
2. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
3. If *profileList*[0] contains at least one Configuration with type equals to configuration type from *confTypeList*:
   3.1. ONVIF Client invokes **RemoveConfiguration** request with parameters
   - ProfileToken := *profile*.@token

- If *profileList*[0] contains Configuration.VideoSource and *confTypeList* contains VideoSource:
  - Configuration[0].Type := VideoSource
  - Configuration[0].Token skipped
- If *profileList*[0] contains Configuration.VideoEncoder and *confTypeList* contains VideoEncoder:
  - Configuration[1].Type := VideoEncoder
  - Configuration[1].Token skipped
- If *profileList*[0] contains Configuration.AudioSource and *confTypeList* contains AudioSource:
  - Configuration[2].Type := AudioSource
  - Configuration[2].Token skipped
- If *profileList*[0] contains Configuration.AudioEncoder and *confTypeList* contains AudioEncoder:
  - Configuration[3].Type := AudioEncoder
  - Configuration[3].Token skipped
- If *profileList*[0] contains Configuration.AudioOutput and *confTypeList* contains AudioOutput:
  - Configuration[4].Type := AudioOutput
  - Configuration[4].Token skipped
- If *profileList*[0] contains Configuration.AudioDecoder and *confTypeList* contains AudioDecoder:
  - Configuration[5].Type := AudioDecoder
  - Configuration[5].Token skipped
- If *profileList*[0] contains Configuration.Metadata and *confTypeList* contains Metadata:
  - Configuration[6].Type := Metadata

- Configuration[6].Token skipped

- If *profileList*[0] contains Configuration.Analytics and *confTypeList* contains Analytics:
  - Configuration[7].Type := Analytics
  - Configuration[7].Token skipped
- If *profileList*[0] contains Configuration.PTZ and *confTypeList* contains PTZ:
  - Configuration[8].Type := PTZ
  - Configuration[8].Token skipped

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.23 Audio Recording to Object Storage Recording Prerequisite

Name: HelperAudioRecordingExternalStoragePrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare audio recording for object storage recording.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by the SupportedTargetFormats capability.

- Object storage is supported as indicated by the System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of audio is supported by the DUT as indicated by the Encoding capability which contains an audio encoding.

Input:

- Required audio encoding (*audioEncoding*).
- Required format type (*formatType*).

Returns:

- Recording job token (*recordingJobToken*).
- Recording span duration (*recordingSpanDuration*).
- Recording segment duration (*recordingSegmentDuration*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
2. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.
3. ONVIF Client selects a Media Profile with required audio encoding support by following the procedure mentioned in Annex A.24 with the following input and output parameters:
   - in *audioEncoding* - required audio encoding
   - out *profile* - Media Profile with Audio Source Configuration and Audio Encoder Configuration with the required audio encoding
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
4. ONVIF Client invokes **SetAudioEncoderConfiguration** request with the following parameters:
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := *audioEncoding*

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

5. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
6. Set *confTypeList* := (VideoSource, VideoEncoder, AudioOutput, AudioDecoder, Metadata, Analytics, PTZ)
7. ONVIF Client removes all configurations except AudioSource and AudioEncoder from the Media Profile by following the procedure mentioned in Annex A.22 with the following input and output parameters:
   - in *confTypeList* - list of configuration types to remove from Media Profile
   - in *profile* - Media Profile to update
8. ONVIF Client invokes **GetRecordings** request.
9. The DUT responds with **GetRecordingsResponse** message with the following parameters:
   - RecordingItem list =: *recordingsList1*
10. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:
    - out *externalStorage1* - external storage configuration identifier
11. Set *recordingSegmentDuration* := **PT20S**.
12. Set *recordingSpanDuration* := **PT60S**.
13. Set:
    - *target1*.Storage := *externalStorage1*
    - *target1*.Format := *formatType*
    - *target1*.SpanDuration := *recordingSpanDuration*
    - *target1*.SegmentDuration := *recordingSegmentDuration*
    - *target1*.Prefix is skipped

- *target1*.Postfix is skipped
- *target1*.Encryption is skipped
- *target1*.SegmentDurationOverride is skipped

14. If *cap*.DynamicRecording = true:

14.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 14.5.

14.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:

- RecordingToken := *recordingsList1*[0].RecordingToken

14.3. The DUT responds with **DeleteRecordingResponse** message.

14.4. Set:

- *recordingConfiguration1*.Source.Name := "Source1"
- *recordingConfiguration1*.Source.SourceId := [valid URI]
- *recordingConfiguration1*.Source.Address := [valid URI]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

14.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

14.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

14.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

14.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

14.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

14.10 If *recordingOptions1*.Job.CompatibleSources does not contain *profile*.@token, FAIL the test, restore DUT settings and skip other steps.

14.11 Set *sourceToken* := *profile*.@token.

15. If *cap*.DynamicRecording = false:

15.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

15.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

15.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

15.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

15.1.4 If *recordingOptions1*.Job.CompatibleSources contains *profile*.@token:

15.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

15.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

15.1.4.3 Set *sourceToken* := *profile*.@token.

15.1.4.4 Go to the step 15.3.

15.2. If no recording was found, FAIL the test, restore DUT settings and skip other steps.

15.3. Set:

- *recordingConfiguration*.Target := *target1*

15.4. ONVIF Client invokes **SetRecordingConfiguration** request with the following parameters:

- RecordingToken := *recordingToken*

- RecordingConfiguration := *recordingConfiguration*

15.5. The DUT responds with **SetRecordingConfigurationResponse** message.

16. Set:
    - *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
    - *source1*.SourceToken.Token = *sourceToken*
    - *source1*.AutoCreateReceiver is skipped
    - *source1*.Tracks is skipped
    - *source1*.Extension is skipped
17. Set:
    - *recordingJobConfiguration1*.ScheduleToken is skipped
    - *recordingJobConfiguration1*.RecordingToken := *recordingToken*
    - *recordingJobConfiguration1*.Mode := **Idle**
    - *recordingJobConfiguration1*.Priority := 0
    - *recordingJobConfiguration1*.Source[0] := *source1*
    - *recordingJobConfiguration1*.Extension is skipped
    - *recordingJobConfiguration1*.EventFilter is skipped
18. ONVIF Client invokes **CreateRecordingJob** request with parameters:
    - JobConfiguration := *recordingJobConfiguration1*
19. The DUT responds with **CreateRecordingJobResponse** message with parameters:
    - JobToken =: *recordingJobToken*
    - JobConfiguration =: *recordingJobConfiguration*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.24 Media2 Service – Media Profile Configuration for Audio Streaming

Name: HelperConfigureMediaProfileForAudioStreaming

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification

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

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

Input: Required audio encoding (*requiredAudioEncoding*)

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

Procedure:

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

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

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

- Options list =: *optionsList*

3.3. If *requiredAudioEncoding* = AAC:

3.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":

3.3.1.1. Set *profile* := *profile1*.

3.3.1.2. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if it exists, otherwise item with Encoding = "MPEG4-GENERIC".

3.3.1.3. Skip other steps in procedure.

3.4. If *requiredAudioEncoding* = !AAC:

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

3.4.1.1. Set *profile* := *profile1*.

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

3.4.1.3. Skip other steps in procedure.

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

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

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

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

- Configurations list =: *audioSourceConfList*

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

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

- ProfileToken := *profile1*.@token
- Name skipped
- Configuration[0].Type := AudioSource
- Configuration[0].Token := *audioSourceConfiguration1*.@token

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

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

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

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

- Configurations list =: *audioEncoderConfList*

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

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

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

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

- Options list =: *optionsList*

4.3.5.3. If *requiredAudioEncoding* = AAC:

4.3.5.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":

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

- ProfileToken := *profile1*.@token
- Name skipped
- Configuration[0].Type := AudioEncoder
- Configuration[0].Token := *audioEncoderConfiguration1*.@token

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

4.3.5.3.1.3. Set *profile* := *profile1*.

4.3.5.3.1.4. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if it exists, otherwise item with Encoding = "MPEG4-GENERIC".

4.3.5.3.1.5. Skip other steps in procedure.

4.3.5.4. If *requiredAudioEncoding* = !AAC:

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

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

- ProfileToken := *profile1*.@token
- Name skipped
- Configuration[0].Type := AudioEncoder
- Configuration[0].Token := *audioEncoderConfiguration1*.@token

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

4.3.5.4.1.3. Set *profile* := *profile1*.

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

4.3.5.4.1.5. Skip other steps in procedure.

5. FAIL the test and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.25 Video And Audio Recording to Object Storage Recording Prerequisite

Name: HelperVideoAndAudioRecordingExternalStoragePrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to prepare video and audio recording for object storage recording.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Media2 Service is received from the DUT.
- Recording to external storage is supported by the DUT as indicated by SupportedTargetFormats.

- Object storage is supported as indicated by System.StorageTypesSupported Device Management capability which contains ObjectStorageS3 or ObjectStorageAzure.
- Recording of video is supported by the DUT as indicated by Encoding capability which contains video encoding.
- Recording of audio is supported by the DUT as indicated by Encoding capability which contains video encoding.

Input:

- Applicable video encoding list (*videoEncodingList*).
- Applicable audio encoding list (*audioEncodingList*).
- Required format type (*formatType*).
- Encryption (*encryption*) (optional).

Returns:

- Recording token (*recordingToken*).
- Recording job token (*recordingJobToken*).
- Recording span duration (*recordingSpanDuration*).
- Recording segment duration (*recordingSegmentDuration*).
- Audio encoding (*audioEncoding*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters:
   - out *cap* - Recording Control Service Capabilities
2. Remove all items from *videoEncodingList* which are not the part of *cap*.Encoding.
3. If *audioEncodingList* contains AAC, replace it with two values "MP4A-LATM" and "MPEG4-GENERIC".
4. Remove all items from *audioEncodingList* which are not the part of *cap*.Encoding.
5. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.

6. ONVIF Client selects a Media Profile with required video and audio encoding support by following the procedure mentioned in Annex A.26 with the following input and output parameters:
   - in *videoEncodingList* - applicable video encoding
   - in *audioEncodingList* - applicable audio encoding
   - out *profile* - Media Profile with Video Source Configuration, Video Encoder Configuration, Audio Source Configuration, Audio Encoder Configuration, with the required video and audio encoding
   - out *vecOptions* - Video Encoder Configuration Options for the Media Profile
   - out *aecOptions* - Audio Encoder Configuration Options for the Media Profile
7. ONVIF Client invokes **SetVideoEncoderConfiguration** request with parameters
   - Configuration.Multicast := *profile*.Configurations.VideoEncoder.Multicast
   - Configuration.@token := *profile*.Configurations.VideoEncoder.@token
   - Configuration.Name := *profile*.Configurations.VideoEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.VideoEncoder.UseCount
   - Configuration.@GovLength := minimum item from *vecOptions*.@GovLengthRange list (or skipped if *vecOptions*.@GovLengthRange skipped)
   - Configuration.@Profile := highest value from *vecOptions*.@ProfilesSupported list as the order is High/Extended/Main/Baseline (or skipped if *vecOptions*.@ProfilesSupported skipped)
   - Configuration.Encoding := *vecOptions*.Encoding
   - Configuration.Resolution := resolution closest to 640x480 from *vecOptions*.ResolutionsAvailable list
   - if *vecOptions*.@FrameRatesSupported skipped and *profile*.Configurations.VideoEncoder.RateControl skipped:
     - Configuration.RateControl skipped
   - if *vecOptions*.@FrameRatesSupported or *profile*.Configurations.VideoEncoder.RateControl is not skipped:

- Configuration.RateControl.@ConstantBitRate := *vecOptions*.@ConstantBitRateSupported
- Configuration.RateControl.FrameRateLimit := value closest to 25 but greater than 1 from *vecOptions*.@FrameRatesSupported list (or *profile*.Configurations.VideoEncoder.RateControl.FrameRateLimit if *vecOptions*.@FrameRatesSupported skipped)
- Configuration.RateControl.BitrateLimit := min {max {*profile*.Configurations.VideoEncoder.RateControl.BitrateLimit, *vecOptions*.BitrateRange.Min}, *vecOptions*.BitrateRange.Max}

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

8. The DUT responds with **SetVideoEncoderConfigurationResponse** message.
9. ONVIF Client invokes **SetAudioEncoderConfiguration** request with parameters
   - Configuration.@token := *profile*.Configurations.AudioEncoder.@token
   - Configuration.Name := *profile*.Configurations.AudioEncoder.Name
   - Configuration.UseCount := *profile*.Configurations.AudioEncoder.UseCount
   - Configuration.Encoding := *aecOptions*.Encoding
   - Configuration.Multicast := *profile*.Configurations.AudioEncoder.Multicast
   - Configuration.Bitrate := the nearest value to *profile*.Configurations.AudioEncoder.Bitrate from *aecOptions*BitrateList.Items list
   - Configuration.SampleRate := the nearest value to *profile*.Configurations.AudioEncoder.SampleRate from *aecOptions*SampleRateList.Items list
10. The DUT responds with **SetAudioEncoderConfigurationResponse** message.
11. Set *audioEncoding* := *aecOptions*.Encoding
12. Set *confTypeList* := (AudioOutput, AudioDecoder, Metadata, Analytics, PTZ)
13. ONVIF Client removes all configurations except VideoSource and VideoEncoder from the Media Profile by following the procedure mentioned in Annex A.22 with the following input and output parameters
    - in *confTypeList* - list of configuration types to remove from Media Profile

- in *profile* - Media Profile to update

14. ONVIF Client invokes **GetRecordings** request.
15. The DUT responds with **GetRecordingsResponse** message with parameters
    - RecordingItem list =: *recordingsList1*
16. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.4 with the following input and output parameters:
    - out *externalStorage1* - external storage configuration identifier
17. Set *recordingSegmentDuration* := **PT20S**.
18. Set *recordingSpanDuration* := **PT60S**.
19. Set:
    - *target1*.Storage := *externalStorage1*
    - *target1*.Format := *formatType*
    - *target1*.SpanDuration := *recordingSpanDuration*
    - *target1*.SegmentDuration := *recordingSegmentDuration*
    - *target1*.Prefix is skipped
    - *target1*.Postfix is skipped
    - *target1*.Encryption := *encryption* if defined, otherwise is skipped
    - *target1*.SegmentDurationOverride is skipped
20. If *cap*.DynamicRecording = true:

    20.1. If number of items in *recordingsList1* is less than *cap*.MaxRecordings, go to step 20.5.

    20.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:
    - RecordingToken := *recordingsList1*[0].RecordingToken

    20.3. The DUT responds with **DeleteRecordingResponse** message.

    20.4. Set:
    - *recordingConfiguration1*.Source.Name := "Source1"

- *recordingConfiguration1*.Source.SourceId := [valid URI]
- *recordingConfiguration1*.Source.Address := [valid URI]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := PT0S
- *recordingConfiguration1*.Target := *target1*

20.5. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

20.6. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

20.7. ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordingToken*

20.8. The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

20.9. If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

20.10 If *recordingOptions1*.Job.CompatibleSources contains no *profile*.@token, FAIL the test, restore DUT settings and skip other steps.

20.11 Set *sourceToken* := *profile*.@token.

21. If *cap*.DynamicRecording = false:

21.1. For each recording *recordings1* in *recordingsList1* repeat the following steps:

21.1.1 ONVIF Client invokes **GetRecordingOptions** request with parameters:

- RecordingToken := *recordings1*.RecordingToken

21.1.2 The DUT responds with **GetRecordingOptionsResponse** message with parameters:

- Options =: *recordingOptions1*

21.1.3 If *recordingOptions1*.Job.Spare is less than 1, FAIL the test, restore DUT settings and skip other steps.

21.1.4 If *recordingOptions1*.Job.CompatibleSources contains *profile*.@token:

21.1.4.1 Set *recordingToken* := *recordings1*.RecordingToken.

21.1.4.2 Set *recordingConfiguration* := *recordings1*.Configuration.

21.1.4.3 Set *sourceToken* := *profile*.@token.

21.1.4.4 Go to the step 21.3.

21.2. If no recording was found, FAIL the test, restore DUT settings and skip other steps.

21.3. Set:

- *recordingConfiguration*.Target := *target1*

21.4. ONVIF Client invokes **SetRecordingConfiguration** request with parameters

- RecordingToken := *recordingToken*
- RecordingConfiguration := *recordingConfiguration*

21.5. The DUT responds with **SetRecordingConfigurationResponse** message.

22. Set:

- *source1*.SourceToken.Type = **http://www.onvif.org/ver10/schema/Profile**
- *source1*.SourceToken.Token = *sourceToken*
- *source1*.AutoCreateReceiver is skipped
- *source1*.Tracks is skipped
- *source1*.Extension is skipped

23. Set:

- *recordingJobConfiguration1*.ScheduleToken is skipped
- *recordingJobConfiguration1*.RecordingToken := *recordingToken*
- *recordingJobConfiguration1*.Mode := **Idle**

- *recordingJobConfiguration1*.Priority := 0
- *recordingJobConfiguration1*.Source[0] := *source1*
- *recordingJobConfiguration1*.Extension is skipped
- *recordingJobConfiguration1*.EventFilter is skipped

24. ONVIF Client invokes **CreateRecordingJob** request with parameters:
    - JobConfiguration := *recordingJobConfiguration1*
25. The DUT responds with **CreateRecordingJobResponse** message with parameters:
    - JobToken =: *recordingJobToken*
    - JobConfiguration =: *recordingJobConfiguration*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.26 Configure Media Profile For Video and Audio Streaming

Name: HelperConfigureMediaProfileForVideoAndAudioStreaming

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to configure Media profile, Video Encoder Configuration, Audio Encoder Configuration for video and audio streaming.

Pre-requisite:

- Media2 Service is received from the DUT.

Input:

- Required video encoding list (*requiredVideoEncoding*).

- Required audio encoding list (*requiredAudioEncoding*).

Returns:

- Media profile with required configurations (*profile*).
- Video Encoder Configuration Options for the Media Profile (*vecOptions*).
- Audio Encoder Configuration Options for the Media Profile (*aecOptions*).

Procedure:

1. ONVIF Client invokes **GetProfiles** request with parameters
   - Token skipped
   - Type[0] := All
2. The DUT responds with **GetProfilesResponse** message with parameters
   - Profiles list =: *profileList*
3. For each Media Profile *profile* in *profileList* with not empty Configuration.VideoSource, Configuration.VideoEncoder, Configuration.AudioSource, and Configuration.AudioEncoder repeat the following steps:

   3.1. If *profile*.Configuration.VideoEncoder is listed at *requiredVideoEncoding*

   3.1.1. ONVIF Client invokes **GetAudioEncoderConfigurationOptions** request with parameters
   - ConfigurationToken := *profile*.Configuration.AudioEncoder.@token
   - ProfileToken := *profile*.@token

   3.1.2. DUT responds with **GetAudioEncoderConfigurationOptionsResponse** message with parameters
   - Options list =: *optionsList*

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

   3.1.3.1. Set *aecOptions* := item from *optionsList* with Encoding which is listed at *requiredAudioEncoding*.

   3.1.4. If *aecOptions* != NULL

   3.1.4.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *profile*.Configuration.VideoEncoder.@token
- ProfileToken := *profile*.@token

3.1.4.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

3.1.4.3. Set *vecOptions* := item from *optionsList* with Encoding which is listed at *requiredVideoEncoding*.

3.1.4.4. Go to step 11.

4. Set *profile* = *profileList*[0].
5. Set *confTypeList* := (configurations that are contained in profile *profile*)
6. ONVIF Client removes all configurations from the Media Profile by following the procedure mentioned in Annex A.22 with the following input and output parameters
   - in *confTypeList* - list of configuration type to remove from Media Profile
   - in *profile* - Media Profile to update
7. ONVIF Client invokes **GetVideoSourceConfigurations** request with parameters
   - ConfigurationToken skipped
   - ProfileToken := *profile*.@token
8. The DUT responds with **GetVideoSourceConfigurationsResponse** with parameters
   - Configurations list =: *videoSourceConfList*
9. For each Video Source Configuration *videoSourceConfiguration* in *videoSourceConfList* repeat the following steps:

   9.1. ONVIF Client invokes **AddConfiguration** request with parameters
   - ProfileToken := *profile*.@token
   - Name skipped
   - Configuration[0].Type := VideoSource
   - Configuration[0].Token := *videoSourceConfiguration*.@token

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

9.3. ONVIF Client invokes **GetVideoEncoderConfigurations** request with parameters

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

9.4. The DUT responds with **GetVideoEncoderConfigurationsResponse** with parameters

- Configurations list =: *videoEncoderConfList*

9.5. For each Video Encoder Configuration *videoEncoderConfiguration* in *videoEncoderConfList* repeat the following steps:

9.5.1. ONVIF Client invokes **GetVideoEncoderConfigurationOptions** request with parameters

- ConfigurationToken := *videoEncoderConfiguration*.@token
- ProfileToken := *profile*.@token

9.5.2. DUT responds with **GetVideoEncoderConfigurationOptionsResponse** message with parameters

- Options list =: *optionsList*

9.5.3. If *optionsList* list contains item which is listed at *requiredVideoEncoding*:

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

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

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

9.5.3.3. Set *vecOptions* := item from *optionsList* with Encoding which is listed at *requiredVideoEncoding*.

9.5.3.4. ONVIF Client tries to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding support to the Media Profile by following the procedure mentioned in Annex A.27 with the following input and output parameters

- in *requiredAudioEncoding* - required audio encoding
- in *profile* - Media profile
- out (optional) *aecOptions* - Audio Encoder Configuration Options for the Media Profile

9.5.3.5. If *aecOptions* != NULL

9.5.3.5.1 Go to step 11.

9.5.3.6. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile*.@token
- Configuration[0].Type = VideoEncoder
- Configuration[0].Token skipped

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

10. If step 9 ends with *aecOptions* = NULL, FAIL test, restore DUT settings, and skip other steps.

11. Return to the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: See Annex A.28 for Name and Token Parameters Length limitations.

A.27 Media2 Service – Adding AudioSource and AudioEncoder with Specified Audio Encoder Value to Media Profile

Name: HelperAddAudioConfigurationsWithSpecificEncoderToMediaProfile

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to add AudioSource Configuration and AudioEncoder Configuration with required audio encoding to the Media Profile if corresponding AudioEncoder Configuration found.

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

Input: Required audio encoding (*requiredAudioEncoding*). Media Profile (*profile*)

Returns: Audio Encoder Configuration Options for the Media Profile (*aecOptions*) (optional, returned in case profile was configured with audio).

Procedure:

1. ONVIF Client invokes **GetAudioSourceConfigurations** request with parameters
   - ConfigurationToken skipped
   - ProfileToken := *profile*.@token
2. The DUT responds with **GetAudioSourceConfigurationsResponse** with parameters
   - Configurations list =: *audioSourceConfList*
3. For each Audio Source Configuration *audioSourceConfiguration* in *audioSourceConfList* repeat the following steps:

   3.1. ONVIF Client invokes **AddConfiguration** request with parameters
   - ProfileToken := *profile*.@token
   - Name skipped
   - Configuration[0].Type := AudioSource
   - Configuration[0].Token := *audioSourceConfiguration*.@token

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

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

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

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

- Configurations list =: *audioEncoderConfList*

3.5. For each Audio Encoder Configuration *audioEncoderConfiguration* in *audioEncoderConfList* repeat the following steps:

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

- ConfigurationToken := *audioEncoderConfiguration*.@token
- ProfileToken := *profile*.@token

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

- Options list =: *optionsList*

3.5.3. If *requiredAudioEncoding* = AAC:

3.5.3.1. If *optionsList* list contains item with Encoding = "MP4A-LATM" or "MPEG4-GENERIC":

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

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

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

3.5.3.1.3. Set *aecOptions* := item with Encoding = "MP4A-LATM" from *optionsList* list if exists, otherwise item with Encoding = "MPEG4-GENERIC".

3.5.3.1.4. Skip other steps in procedure.

3.5.4. If *requiredAudioEncoding* != AAC:

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

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

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

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

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

3.5.4.1.4. Skip other steps in procedure.

3.6. ONVIF Client invokes **RemoveConfiguration** request with parameters

- ProfileToken = *profile*.@token
- Configuration[0].Type = AudioSource
- Configuration[0].Token skipped

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.28 Name and Token Parameters

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

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

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

A.29 Segment With Symmetric CENC Encryption Check

Name: HelperCENCSegmentCheck

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to check segment with symmetric encryption with CENC format.

Pre-requisite:

- None.

Input:

- Segment (*segment*).
- Expected KID_count (*kidCount*).

- Expected KID list (*kidList*).
- Keys list (*keysList*).

Returns:

- None.

Procedure:

1. If *segment* does no contains ProtectionSystemSpecificHeaderBox PSSH box (SystemID is equal to **1077efec-c0b2-4d02-ace3-3c1e52e2fb4b**), FAIL the test, restore DUT settings and skip other steps.
2. If *segment* ProtectionSystemSpecificHeaderBox PSSH box does not have the following values of the fields, FAIL the test, restore DUT settings and skip other steps:
   - Version is equal to **1**.
   - Flags is equal to **0**.
   - KID_count is equal to *kidCount*.
   - KID list is equal to *kidList*.
   - DataSize is equal to **0**.
3. If *segment* could not be decrypted using *keysList*, FAIL the test, restore DUT settings and skip other steps.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not pass all assertions.

A.30 Segment With Asymmetric Encryption Check

Name: HelperAsymmetricSegmentCheck

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to check segment with asymmetric encryption.

Pre-requisite:

- None.

Input:

- Segment (*segment*).
- Certificates list with private keys (*certificatesList*).
- Expected KID (*expectedKid*) (optional).

Returns:

- KID (*kid*).
- Symmetric key (*key*).

Procedure:

1. If *segment* does no contains AsymmetricKeySystemHeaderBox PSSH box (SystemID is equal to **a4852bd0-80fc-484e-b9e1-78a74d49f5ce**), FAIL the test, restore DUT settings and skip other steps.
2. ONVIF Client checks which OAEP-PSS Digest algorithms (*digestAlgorithms*) the DUT shall support depending on required Security Baseline versions by the rules defined in section Recording Asymmetric Encryption Version 1 Requirements of [ONVIF Security Baseline Device Test Spec].
3. ONVIF Client checks which HPKE algorithms (*hpkeAlgorithms*) the DUT shall support depending on required Security Baseline versions by the rules defined in section Recording Asymmetric Encryption Version 2 Requirements of [ONVIF Security Baseline Device Test Spec].
4. If *segment* AsymmetricKeySystemHeaderBox PSSH box does not have the following values of the fields, FAIL the test, restore DUT settings and skip other steps:
   - Version is equal to **1**.
   - Flags is equal to **0**.
   - KID_count is equal to **1**.
   - KID list is contain only one item with UUID.
   - If *expectedKid* is specified, KID[0] is equal to *expectedKid*.

- DataSize is equal to size in bytes of all the other fields present in this box following this field.
- Number of encrypted key entries is equal to number of items in *certificatesList*.
- For each *certificate* at *certificatesList*:
  - Check if there is CertificateThumbprint which corresponding to *certificate* (CertificateThumbprintAlgorithm and CertificateThumbprintSize to be used for calculation).
  - EncryptionVersion is **1** if CertificateThumbprint corresponds to RSA based certificate.
  - If EncryptionVersion is **1**, one of *digestAlgorithms* OAEP-PSS Digest Algorithms is used for encryption.
  - EncryptionVersion is **2** if CertificateThumbprint corresponds to ECC based certificate.
  - If EncryptionVersion is **2**, HpkeKem, HpkeHkdf, EncapsulatedSharedSecretSize and EncapsulatedSharedSecret is defined.
  - If EncryptionVersion is **2**, one of the *hpkeAlgorithms* sets is used for encryption.
  - If EncapsulatedSharedSecret is defined EncapsulatedSharedSecretSize contains size of EncapsulatedSharedSecret.
  - EncryptedSymmetricKeySize contains size of EncryptedSymmetricKey.
  - EncryptedSymmetricKey could be decrypted by private key of certificate corresponds to CertificateThumbprint.

5. If decrypted symmetric keys are different for different certificates, FAIL the test, restore DUT settings and skip other steps.
6. Set *kid* := KID[0].
7. Set *key* := [decrypted symmetric key].
8. If *segment* does no contains ProtectionSystemSpecificHeaderBox PSSH box (SystemID is equal to **1077efec-c0b2-4d02-ace3-3c1e52e2fb4b**), FAIL the test, restore DUT settings and skip other steps.
9. If *segment* ProtectionSystemSpecificHeaderBox PSSH box does not have the following values of the fields, FAIL the test, restore DUT settings and skip other steps:
   - Version is equal to **1**.

- Flags is equal to **0**.
- KID_count is equal to **1**.
- KID list is equal to [*kid*].
- DataSize is equal to **0**.

10. If *segment* could not be decrypted using decrypted symmetric key, FAIL the test, restore DUT settings and skip other steps.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not pass all assertions.

A.31 Provide Certificate Signed By CA Certificate

Name: HelperCreateCertificateSignedByCA

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to create an X.509 certificate signed by CA certificate.

Pre-requisite: None.

Input:

- The subject (*subject*) of certificate (optional input parameter, could be skipped).
- The service capabilities (*cap*).
- The key pair algorithm (*keyAlgorithm*).
- An X.509 CA certificate (*CAcert*) that is compliant to [RFC 5280] and a corresponding key pair (*CAkeyPair*) with private key and public key.

Returns:

- An X.509 certificate (*cert*) that is compliant to [RFC 5280] and a corresponding key pair (*keyPair*) with private key and public key.

Procedure:

1. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.16 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
2. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
3. If *subject* is skipped set:
   - *subject* := "CN=ONVIF TT Child,C=US"
4. ONVIF Client creates an X.509 certificate signed with *CAcert* that is compliant to [RFC 5280] and has the following properties:
   - version := v3
   - signature := *signatureAlgorithm*
   - validity := not before 19700101000000Z and not after 99991231235959Z
   - subject := *subject*
   - public key := *keyPair* .publicKey
   - private key to be used := *keyPair*.privateKey

Note: ONVIF Client may return the same CA certificate in subsequent invocations of this procedure for the same subject.

A.32 Upload a certificate without Private Key Assignment

Name: HelperUploadCertificate

Procedure Purpose: Helper procedure to upload a certificate without private key assignment.

Pre-requisite: Security Configuration Service is received from the DUT. Create PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability. The DUT shall have enough free storage capacity for one additional key pair. The DUT shall have enough free storage capacity for one additional certificate.

Input: Certificate (*cert*).

Returns: The identifier of the new key pair (*keyID*) and a certificate identifier (*certID*).

Procedure:

1. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *cert*
   - Alias := "ONVIF_Test"
   - PrivateKeyRequired : = false
2. DUT responds with a **UploadCertificateResponse** message.
   - Certificate := *certID*
   - KeyID := *keyID*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.33 Configure Authorization Server On Device and Start It

Name: HelperConfigureAndStartAuthServer

Notes: Annex is used at:

- ONVIF Media2 Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification

Procedure Purpose: Helper procedure configures connection Authorization Server on Device and starts Authorization Server.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.
- OAuthClientCredentials authentication method is supported by the DUT as indicated by the AuthorizationServer.ConfigurationTypesSupported capability.
- client_secret_basic of private_key_jwt authentication is supported by the DUT as indicated by the AuthorizationServer.ClientAuthenticationMethodsSupported capability.

Input:

- Security Configuration Service Capabilities (*cap*).

Returns:

- Authentication server configuration token (*authServerConfToken*).
- Authentication scope (*scope1*).
- Authentication server metadata endpoint (*authServerMetadataEndpoint1*).

Procedure:

1. ONVIF Client deletes one authorization server configuration if maximum is reached by following the procedure described in Annex A.34 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *itemToRestore1* - deleted authorization server configuration if any
2. ONVIF Client selects key pair algorithm by following the procedure mentioned in Annex A.19 with the following input and output parameters:
   - in *cap* - security configuration service capabilities
   - out *keyAlgorithm* - key pair algorithm
3. ONVIF Client selects authorization server type and authentication method by following the procedure mentioned in Annex A.38 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *authServerType* - authorization server configuration type

- out *authMethod* - authentication method

4. ONVIF Client configures authentication server connection using the following steps:

4.1. ONVIF Client creates certification path validation policy by following the procedure mentioned in Annex A.36 with the following input and output parameters

- in *cap* - DUT capabilities
- in *keyAlgorithm* - key pair algorithm
- in "C=US,O=ONVIF,CN=ONVIF TT AuthServer 1" - CA certificate subject
- in "Test CertPathValidationPolicy AuthServer Alias" - certification path validation policy alias
- out *certPathValidationPolicyIDAuthServer* - certification path validation policy identifier
- out *certIDAuthServer* - certificate identifier
- out *keyIDAuthServer* - key pair identifier
- out *CACertAuthServer* - CA certificate
- out *privateKeyCACertAuthServer* - CA certificate private key

4.2. ONVIF Client creates a certificate signed by private key of the CA-certificate with subject and a corresponding public key in the certificate along with the corresponding private key by following the procedure described in Annex A.37 with the following input and output parameters:

- in *cap* - DUT capabilities
- in "C=US,O=ONVIF,CN=*Authorization Server IP Address*" - certificate subject
- in "*Authorization Server IP Address*" - certificate SAN
- in *CACertAuthServer* - CA-certificate
- in *privateKeyCACertAuthServer* - private key of CA-certificate for certificate signature
- out *certAuthServer* - certificate
- out *publicKeyAuthServer* - public key of certificate

- out *privateKeyAuthServer* - private key of certificate

4.3. ONVIF Client starts Authorization server with metadata endpoint *authServerMetadataEndpoint1* conforming to RFC8414 with following settings:

- It is configured to *certAuthServer* as a server certificate.
- It is configured *authServerType* authorization server type.
- It is configured to accept *authMethod* authentication method.

4.4. If *authMethod* is private_key_jwt ONVIF Client configures key pair using the following steps:

4.4.1. ONVIF Client creates a key pair and get public key for JWT authentication from device by following the procedure described in Annex A.39 with the following input and output parameters:

- in *keyAlgorithm* - CSR key pair algorithm
- out *keyID1Auth1* - key pair
- out *publicKeyAuth1* - public key

4.5. ONVIF Client configures Authorization Server with the following credentials to be authorized:

- client identifier *clientID1*
- client secret *clientSecret1*
- scope *scope1*
- public key *publicKeyAuth1* assigned if *authMethod* is private_key_jwt

4.6. ONVIF Client invokes **CreateAuthorizationServerConfiguration** request with parameters

- Type := *authServerType*
- ClientAuth := *authMethod*
- ServerUri := *authServerMetadataEndpoint1*
- ClientID := *clientID1*
- ClientSecret := *clientSecret1*

- Scope := *scope1*
- KeyID := *keyID1Auth1* if *authMethod* is private_key_jwt, otherwise is skipped
- CertificateID is skipped
- CertPathValidationPolicyID := *certPathValidationPolicyIDAuthServer*

4.7. The DUT responds with **CreateAuthorizationServerConfigurationResponse** message with parameters

- Token =: *authServerConfToken*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.34 Make Sure That At Least One Authorization Server Configuration Could Be Created

Name: HelperEmptySpaceForOneAuthorizationServerConfiguration

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to remove one authorization server configuration if maximum is reached.

Pre-requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.

Input: The service capabilities (*cap*).

Returns: Removed authorization server configuration (*removedAuthServerConfiguration*), could be empty.

Procedure:

1. ONVIF Client gets current authorization server configurations by following the procedure mentioned in Annex A.35 with the following input and output parameters:
   - out *authServerConfigurations1* list - authorization server configurations
2. If *authServerConfigurations1* items number is equal to *cap*.AuthorizationServer.MaxConfigurations:
   2.1. Set the following:
      - *removedAuthServerConfiguration* := *authServerConfigurations1*[0]
   2.2. ONVIF Client invokes **DeleteAuthorizationServerConfiguration** with parameter
      - Token := *removedAuthServerConfiguration*.token
   2.3. The DUT responds with **DeleteAuthorizationServerConfigurationResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.35 Get Authorization Server Configurations List

Name: HelperGetAuthorizationServerConfigurations

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to get authorization server configurations list.

Pre-requisite: Security Configuration Service is received from the DUT. Authorization Server Configuration is supported by the DUT as indicated by the AuthorizationServer.MaxConfigurations capability.

Input: None

Returns: Authorization server configurations list (*authServerConfigurations*).

Procedure:

1. ONVIF Client invokes **GetAuthorizationServerConfigurations** request with parameters
   - Token is skipped
2. The DUT responds with **GetAuthorizationServerConfigurationsResponse** message with parameters
   - Configuration list =: *authServerConfigurations*

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.36 Create a certification path validation policy for authentication server configuration

Name: HelperCreateCertPathValidationPolicyForAuthServer

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Media2 Configuration Device Test Specification

- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create a certification path validation policy for authentication server configuration.

Pre-requisite: Security Configuration Service is received from the DUT. Certification path validation policy supported by the DUT as indicated by the MaximumNumberOfCertificationPathValidationPolicies capability. UploadCertificate is supported by the DUT as indicated by the PKCS10ExternalCertificationWithRSA or PKCS10 capability. The DUT shall have enough free storage capacity for one additional certification path validation policy. The DUT shall have enough free storage capacity for one additional certification path. The DUT shall have enough free storage capacity for one additional certificate. The DUT shall have enough free storage capacity for one additional key pair.

Input: The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). The certification path validation policy alias (*certPathValidationPolicyAlias*). The subject (*subject*) of CA certificate (optional input parameter, could be skipped).

Returns: The certification path validation policy identifier (*certPathValidationPolicyID*), related certificate (*certID*), key pair (*keyID*), CA certificate (out *CAcert*) CA certificate private key (out *privateKey*).

Procedure:

1. ONVIF Client creates a CA certificate and a corresponding private key by following the procedure described in Annex A.15 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - out *CAcert* - CA certificate
   - out *privateKey* - private key
2. ONVIF Client invokes **UploadCertificate** with parameters
   - Certificate := *CAcert*
   - Alias := "ONVIF Test"
   - PrivateKeyRequired : = false
3. The DUT responds with a **UploadCertificateResponse** message with parameters
   - CertificateID =: *certID*
   - KeyID =: *keyID*

4. ONVIF Client creates certification path validation policy identifier with specified alias and the certificate identifier for trust anchor by following the procedure mentioned in Annex A.6 with the following input and output parameters:.
   - in *certID* - certificate identifier for trust anchor
   - in *certPathValidationPolicyAlias* - certification path validation policy alias
   - out *certPathValidationPolicyID* - certification path validation policy identifier

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.37 Provide certificate signed by private key of other certificate

Name: HelperCreateSignedCertificate

Notes: Annex is used at:

- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Base Device Test Specification

Procedure Purpose: Helper procedure to create an X.509 certificate signed by private key of other certificate.

Pre-requisite: None.

Input: The subject (*subject*) of certificate and private key (*inputPrivateKey*) of the CA-certificate (*cert*). The service capabilities (*cap*). The key pair algorithm (*keyAlgorithm*). Certificate SAN (*certSAN*, optional).

Returns: An X.509 certificate (*cert*) signed by input private key that is compliant to [RFC 5280] and a corresponding key pair (*keyPair*) with the corresponding private key and public key.

Procedure:

1. ONVIF Client generates a key pair by following the procedure mentioned in Annex A.17 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyPair* - key pair
2. ONVIF Client selects signature algorithm by following the procedure mentioned in Annex A.16 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *inputPrivateKey*.algorithm - key pair algorithm
   - out *signatureAlgorithm* - signature algorithm
3. ONVIF Client creates an X.509 certificate signed by *inputPrivateKey* that is compliant to [RFC 5280] and has the following properties:
   - version:= v3
   - signature := *signatureAlgorithm*
   - publicKey := *keyPair*.publicKey
   - validity := validity from *cert*
   - subject := *subject*
   - SAN := *certSAN*
   - issuerDN := subjectDN from *cert*

Note: ONVIF Client may return the same certificate in subsequent invocations of this procedure for the same subject and private key.

A.38 Authorization Server Configuration Type And Authentication Method Selection

Name: HelperAuthServerTypeAndMethodSelection

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification

Procedure Purpose: Helper procedure to select type and authentication method for authorization server configuration which will be used for tests based on Security Configuration Service capabilities.

Pre-requisite: Security Configuration Service is received from the DUT.

Input: The service capabilities (*cap*).

Returns: The authorization server configuration type (*authServerType*). The authentication method (*authMethod*).

Procedure:

1. ONVIF Client selects authorization server configuration type (*authServerType*) that will be used for the test from the list provided by DUT at *cap*.AuthorizationServer.ConfigurationTypesSupported. Selection is done among the following list of server configuration type supported by the ONVIF Client by priority from first to last:

   - OAuthClientCredentials

2. ONVIF Client selects authentication method (*authMethod*) that will be used for the test from the list provided by DUT at *cap*.AuthorizationServer.ClientAuthenticationMethodsSupported. Selection is done among the following list of server configuration type supported by the ONVIF Client by priority from first to last:

   - client_secret_basic

   - private_key_jwt

3. If after the previous steps execution authorization server configuration type (*authServerType*) is empty, FAIL the procedure.

4. If after the previous steps execution authentication method (*authMethod*) is empty, FAIL the procedure.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not return any of signature algorithms listed at step 1 or 2.

A.39 Create Key Pair and Receive Public Key

Name: HelperCreateJWTKeyPairAndReceivePublicKey

Notes: Annex is used at:

- ONVIF Base Device Test Specification
- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Media2 Configuration Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Uplink Test Specification

Procedure Purpose: Helper procedure to create a key pair and get public key from the device.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- Create PCKS#10 supported by the DUT as indicated by the PKCS10 or PKCS10ExternalCertificationWithRSA capability.
- On-board ECC or RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- The DUT shall have enough free storage capacity for one additional key pair.
- Current time of the DUT shall be at least Jan 01, 1970.

Input:

- The JWT algorithm (*jwtAlgorithm*) (optional).
- The CSR key pair algorithm (*keyAlgorithm*) (optional).
- **Note:** At least one of *keyAlgorithm* and *jwtAlgorithm* to be defined.

Returns:

- The identifier of the new key pair (*keyID*).

- Th public key (*publicKey*).

Procedure:

1. If *jwtAlgorithm* is not defined, set
   - *jwtAlgorithm* := [key pair algorithm as defined for *keyAlgorithm* for latest [ONVIF Security Baseline] version supported by the DUT in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]]
2. ONVIF Client creates a key pair by following the procedure mentioned in Annex A.40 with the following input and output parameters:
   - in [key pair algorithm as defined for *jwtAlgorithm* in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]] - key pair algorithm
   - in [key pair parameters as defined for *jwtAlgorithm* in section Private Key JWT Algorithms for Baseline Versions of [ONVIF Security Baseline Device Test Spec]] - key pair algorithm parameters
   - out *csrKeyID* - key pair ID
3. ONVIF Client invokes **CreatePKCS10CSR** with parameters
   - Subject := *subject* (see Annex A.42)
   - KeyID := *csrKeyID*
   - CSRAttribute skipped
   - SignatureAlgorithm.algorithm := signature algorithm, used for *jwtAlgorithm*
4. The DUT responds with **CreatePKCS10CSRResponse** message with parameters
   - PKCS10CSR =: *PKCS10request*
5. ONVIF Client extracts public key *publicKey* from *PKCS10request*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreatePKCS10CSRResponse** message.

A.40 Create a key pair

Name: HelperCreateKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification

Procedure Purpose: Helper procedure to create ECC or RSA key pair

Pre-requisite:

- Security Configuration Service is received from the DUT.
- On-board ECC or RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.
- The DUT shall have enough free storage capacity for one additional key pair.

Input:

- The service capabilities (*cap*) (optional, required if *keyPairParameters* are not defined).
- The key pair algorithm (*keyAlgorithm*).
- The key pair parameters (key length for RSA; curve for ECC) (*keyGenParams*) (optional).

Returns:

- The identifier of the new key pair (*keyID*).

Procedure:

1. If *keyGenParams* is not defined, ONVIF Client determines the key pair generation parameters by following the procedure mentioned in Annex A.18 with the following input and output parameters:
   - in *cap* - DUT capabilities
   - in *keyAlgorithm* - key pair algorithm
   - out *keyGenParams* - key pair generation parameters

2. If *keyAlgorithm* = RSA:
   a. ONVIF Client invokes **CreateRSAKeyPair** with parameter
      - KeyLength := *keyGenParams*.keyLength
   b. The DUT responds with **CreateRSAKeyPairResponse** message with parameters
      - KeyID =: *keyID*
      - EstimatedCreationTime =: *duration*
3. If *keyAlgorithm* = ECC:
   a. ONVIF Client invokes **CreateECCKeyPair** with parameter
      - EllipticCurve := *keyGenParams*.ellipticCurve
   b. The DUT responds with **CreateECCKeyPairResponse** message with parameters
      - KeyID =: *keyID*
      - EstimatedCreationTime =: *duration*
4. Until *operationDelay* + *duration* expires repeat the following steps:
   4.1. ONVIF Client waits for 5 seconds.
   4.2. ONVIF Client invokes **GetKeyStatus** with parameters
      - KeyID := *keyID*
   4.3. The DUT responds with **GetKeyStatusResponse** message with parameters
      - KeyStatus =: *keyStatus*
   4.4. If *keyStatus* is equal to "ok", *keyID* will be return as a result of the procedure, other steps will be skipped.
   4.5. If *keyStatus* is equal to "corrupt", FAIL the procedure and deletes the key pair (*keyID*) by following the procedure mentioned in Annex A.41.
5. If *operationDelay* + *duration* expires for step 4 and the last *keyStatus* is other than "ok", FAIL the procedure and deletes the key pair (*keyID*) by following the procedure mentioned in Annex A.41.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- DUT did not send **CreateRSAKeyPairResponse** or **CreateECCKeyPairResponse** message.
- DUT did not send **GetKeyStatusResponse** message(s).

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

A.41 Delete a key pair

Name: HelperDeleteKeyPair

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification

Procedure Purpose: Helper procedure to delete a key pair.

Pre-requisite:

- Security Configuration Service is received from the DUT.
- On-board ECC or RSA key pair generation is supported by the DUT as indicated by the RSAKeyPairGeneration capability, or by the ECCKeyPairGeneration capability, or by the KeyPairGeneration capability, which contains RSA or ECC.

Input: The identifier of the key pair (*keyID*) to delete.

Returns: None

Procedure:

1. ONVIF Client invokes **DeleteKey** with parameters
   - KeyID := *keyID*
2. DUT responds with a **DeleteKeyResponse** message.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.42 Subject for a server certificate

Name: HelperSubjectForServerCertificate

Notes: Annex is used at:

- ONVIF Real Time Streaming using Media2 Device Test Specification
- ONVIF Security Configuration Device Test Specification
- ONVIF Base Device Test Specification
- ONVIF Uplink Device Test Specification

Use the following subject for test cases:

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

A.43 Retrieve Recording Configuration Event by PullPoint

Name: HelperPullRecordingConfiguration

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/RecordingConfiguration** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).

- Subscription termination time (*tt*).
- Recording Configuration token (*recordingToken*).

Returns:

- Recording Configuration (*recordingConfiguration*).

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters:
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters:
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains item with Topic = "tns1:RecordingConfig/RecordingConfiguration" and Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*:
      1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* which contains item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*].
      1.4.2. Go to step 2.
   1.5. If *operationDelay* timeout expires for step 1 without Notification with source with RecordingToken = *recordingToken*, FAIL the test and skip other steps.
2. If *notificationMessage*.Message.Data does not contain an ElementItem with Name = "Configuration" and Value of tt:RecordingConfiguration type, FAIL the test, restore the DUT state, and skip other steps.
3. Set *recordingConfiguration* := [value of Value attribute from *notificationMessage*.Message.Data.ElementItem[Name = "Configuration"]].

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.44 Recording Information Comparison

Name: HelperComparisonOfRecordings

Procedure Purpose: Helper procedure to compare of parameter values for the same recording in different messages.

The following fields to be compared for the recording information received/sent in different messages:

- Source
  - SourceId
  - Name
  - Location
  - Description
  - Address
- Content
- MaximumRetentionTime
- Target
  - Storage
  - Format
  - Prefix
  - Postfix

- SpanDuration
- SegmentDuration
- Encryption list (order is not important for validation, Track list to be used as unique identifier for comparison)
  - Mode
  - KID
  - Key
  - Track list (order is not important for validation)
  - AsymmetricEncryption
    - CertificateID list (order is not important for validation)
    - KeyRotationDuration
- SegmentDurationOverride (to be compared only in case of both set of compared were received from the DUT, because field is read only)
  - Duration

Recording information will be assumed as the same if information in listed fields is the same. The absence of field in one structure and present in other to be assumed as different values.

A.45 Recording Source Information Parameters Maximum Length

Name: HelperRecordingSourceInformationParametersMaximumLength

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

1. SourceId shall be less than or equal to 128 characters.
2. Name shall be less than or equal to 20 characters.
3. Address shall be less than or equal to 128 characters.

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

A.46 Retrieve Create Recording Event by PullPoint

Name: HelperPullCreateRecording

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/CreateRecording** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).
- Subscription termination time (*tt*).
- Recording Configuration token (*recordingToken*).

Returns: None.

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters:
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters:
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains an item with Topic = "tns1:RecordingConfig/CreateRecording" and Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*:

1.4.1. Set *notificationMessage_CreateRecording* := [NotificationMessage from *notificationMessageList* with Topic = "tns1:RecordingConfig/CreateRecording" and which contains item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*].

1.5. If all notifications listed below were received, go to step 3:

- *notificationMessage_CreateRecording*

2. If *operationDelay* timeout expires for step 1 without at least one of expected notifications, FAIL the test and skip other steps.

3. Return to the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.47 Retrieve Delete Recording Event by PullPoint

Name: HelperPullDeleteRecording

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/DeleteRecording** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).
- Subscription termination time (*tt*).

- Recording Configuration token (*recordingToken*).

Returns: None.

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters:
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters:
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains an item with Topic = "tns1:RecordingConfig/DeleteRecording" and Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*:
      1.4.1. Set *notificationMessage_DeleteRecording* := [NotificationMessage from *notificationMessageList* with Topic = "tns1:RecordingConfig/DeleteRecording" and which contains item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*].
   1.5. If all notifications listed below were received, go to step 3:
      - *notificationMessage_DeleteRecording*
2. If *operationDelay* timeout expires for step 1 without at least one of expected notifications, FAIL the test and skip other steps.
3. Return to the test.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.48 Creation of Recording Prerequisite

Name: HelperRecordingCreationPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to delete recording if maximum number of recordings supported was reached.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Dynamic Recording is supported by the DUT as indicated by DynamicRecordings = true capability.

Input: None

Returns: None

Procedure:

1. ONVIF Client invokes **GetServiceCapabilitiesRequest** message to retrieve Capabilities for the recording service.
2. Verify **GetServiceCapabilitiesResponse** message.
3. ONVIF Client invokes **GetRecordingsRequest** message to retrieve a complete recordings list.
4. Verify the **GetRecordingsResponse** message from the DUT.
5. If total number of recordings is less than MaxRecordings Capabilities element, skip other steps and run test. If MaxRecordings Capabilities is skipped, return to test procedure steps and try performing CreateRecording step. In error case perform 6-8 steps from this Annex.
6. ONVIF Client invokes **DeleteRecordingRequest** message (RecordingToken = Token1, where Token1 is the first recording token from the **GetRecordingsResponse**) to delete recording.

7. Verify the **DeleteRecordingResponse** message or SOAP 1.2 fault message (Action/CannotDelete or others) from the DUT. If **DeleteRecordingResponse** message received skip other steps and run test.

8. Repeat steps 6-7 for the next recording token. If there is no next Recording Token then skip the test.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.49 Pull Track Configuration

Name: HelperPullTrackConfiguration

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/TrackConfiguration** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).
- Subscription termination time (*tt*).
- Recording Configuration token (*recordingToken*).
- Track Configuration token (*trackToken*).

Returns:

- Track Configuration (*trackConfiguration*).

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains an item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken* and with Message.Source.SimpleItem with Name = "TrackToken" and Value = *trackToken*:
      1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* which contains an item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken* and with Message.Source.SimpleItem with Name = "TrackToken" and Value = *trackToken*].
      1.4.2. Go to step 2.
   1.5. If *operationDelay* timeout expires for step 1 without a notification with source with RecordingToken = *recordingToken* and TrackToken = *trackToken*, FAIL the test and skip other steps.
2. If *notificationMessage*.Message.Data does not contain ElementItem with Name = "Configuration" and Value of tt:TrackConfiguration type, FAIL the test, restore the DUT state, and skip other steps.
3. Set *trackConfiguration* := [value of Value attribute from *notificationMessage*.Message.Data.ElementItem[Name = "Configuration"]].

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.50 Track Information Comparison

Name: HelperComparisonOfTracks

Procedure Purpose: Helper procedure to compare of parameter values for the same track in different messages.

The following fields to be compared for the track information received/sent in different messages:

- TrackType (to be ignored for SetTrackConfiguration operation, because field to be ignored by the DUT)
- Description

Track information will be assumed as the same if information in listed fields is the same. The absence of field in one structure and present in other to be assumed as different values.

A.51 Creation of Recording

Name: HelperRecordingCreation

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to create recording if no recordings exist.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Dynamic Recording is supported by the DUT as indicated by DynamicRecordings = true capability.

Input: None

Returns: None

Procedure:

1. ONVIF Client invokes **GetRecordings** request.
2. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
3. If *recordingsList1* is not empty, skip other steps and return to the test.
4. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities
5. If *cap*.OnboardStorage = false and *cap*.SupportedTargetFormats contains at least one item:
   5.1. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.11 with the following input and output parameters:
      - out *externalStorage1* - external storage configuration identifier
   5.2. Set:
      - *target1*.Storage := *externalStorage1*
      - *target1*.Format := *cap*.SupportedTargetFormats[0]
      - *target1*.Prefix := "Pre-1"
      - *target1*.Postfix := "Post-1"
      - *target1*.SegmentDuration := "PT1M"
      - *target1*.Encryption is skipped
      - *target1*.SegmentDurationOverride is skipped
6. Set:
   - *recordingConfiguration1*.Source.Name := "Test Source"
   - *recordingConfiguration1*.Source.SourceId := "Test SourceId"
   - *recordingConfiguration1*.Source.Address := [valid URI]
   - *recordingConfiguration1*.Source.Location := "Source Location"

- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := "PT0S"
- *recordingConfiguration1*.Target := *target1* (if it was set, skipped otherwise)

7. ONVIF Client invokes **CreateRecording** request with parameters:
   - RecordingConfiguration := *recordingConfiguration1*
8. The DUT responds with **CreateRecordingResponse** message with parameters:
   - RecordingToken =: *recordingToken1*

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

A.52 Retrieve Create Track Event by PullPoint

Name: HelperPullCreateTrack

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/CreateTrack** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).

- Subscription termination time (*tt*).
- Recording Configuration token (*recordingToken*).
- Track Configuration token (*trackToken*).

Returns: None.

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   1.3. The DUT responds with **PullMessagesResponse** message with parameters
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   1.4. If *notificationMessageList* is not empty and contains an item with Topic = "tns1:RecordingConfig/CreateTrack" and Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*:
      1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* with Topic = "tns1:RecordingConfig/CreateTrack" and which contains item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*].
   1.5. If *notificationMessage* was received, go to step 3.
2. If *operationDelay* timeout expires for step 1 without at least one of expected notifications, FAIL the test and skip other steps.
3. If *notificationMessage*.Message.Source does not contain SimpleItem with Name = "TrackToken" and Value = *trackToken*, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.53 Retrieve Delete Track Event by PullPoint

Name: HelperPullDeleteTrack

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/DeleteTrack** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).
- Subscription termination time (*tt*).
- Recording Configuration token (*recordingToken*).
- Track Configuration token (*trackToken*).

Returns: None.

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1

1.3. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*
- TerminationTime =: *tt*
- NotificationMessage list =: *notificationMessageList*

1.4. If *notificationMessageList* is not empty and contains an item with Topic = "tns1:RecordingConfig/DeleteTrack" and Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*:

1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* with Topic = "tns1:RecordingConfig/DeleteTrack" and which contains item with Message.Source.SimpleItem with Name = "RecordingToken" and Value = *recordingToken*].

1.5. If *notificationMessage* was received, go to step 3.

2. If *operationDelay* timeout expires for step 1 without at least one of expected notifications, FAIL the test and skip other steps.

3. If *notificationMessage*.Message.Source does not contain SimpleItem with Name = "TrackToken" and Value = *trackToken*, FAIL the test, restore the DUT state, and skip other steps.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.54 Creation of Track Prerequisite

Name: HelperTrackCreationPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to delete recording if maximum number of recordings supported was reached.

Pre-requisite:

- Recording Control Service is received from the DUT.
- Dynamic Tracks is supported by the DUT as indicated by DynamicTracks = true capability.

Input: None

Returns: Recording token (*recordingToken*)

- Recording token (*recordingToken*).
- Track type (*trackType*).

Procedure:

1. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.
2. Verify the **GetRecordingsResponse** message from the DUT.
3. ONVIF Client will invoke **GetRecordingOptionsRequest** message (RecordingToken as Token of the first recording in the **GetRecordingsResponse** message) to get total spare number of tracks that can be added to this recording.
4. If Options.Track.SpareTotal>0, then return to the test and use this Recording for creation of track. Type of creation track (*trackType*) shall correspond to value of SpareVideo, SpareAudio or SpareMetadata greater than 0.
5. Repeat steps 3-4 for the next RecordingToken from the **GetRecordingsResponse** message.
6. If there are no the next Recording then try do delete Track from recording with track and repeat 4-5 steps.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

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

- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteTrackResponse** message.

Note: If list of recording items is empty, ONVIF Client will retrieve DynamicRecording capability from the DUT. In case the Dynamic capability is true ONVIF Client will create recording with procedure defined in Annex A.51. Otherwise ONVIF Client fails the test.

Note: If DUT returns SOAP fault (Action/CannotDelete) message to DeleteTrack request, then ONVIF will try deleting other tracks.

A.55 Retrieve Recording Job Configuration Event by PullPoint

Name: HelperPullRecordingJobConfiguration

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to retrieve and check **tns1:RecordingConfig/RecordingJobConfiguration** event with PullMessages.

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

Input:

- Subscription reference (*s*).
- Current time for the DUT (*ct*).
- Subscription termination time (*tt*).
- Recording Job Configuration token (*recordingJobToken*).

Returns:

- Recording Job Configuration (*recordingJobConfiguration*).

Procedure:

1. Until *operationDelay* timeout expires, repeat the following steps:
   1.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   1.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S

- MessageLimit := 1

1.3. The DUT responds with **PullMessagesResponse** message with parameters

- CurrentTime =: *ct*
- TerminationTime =: *tt*
- NotificationMessage list =: *notificationMessageList*

1.4. If *notificationMessageList* is not empty and contains item with Message.Source.SimpleItem with Name = "RecordingJobToken" and Value = *recordingJobToken*:

1.4.1. Set *notificationMessage* := [NotificationMessage from *notificationMessageList* which contains item with Message.Source.SimpleItem with Name = "RecordingJobToken" and Value = *recordingJobToken*].

1.4.2. Go to step 2.

1.5. If *operationDelay* timeout expires for step 1 without Notification with source with RecordingJobToken = *recordingJobToken*, FAIL the test and skip other steps.

2. If *notificationMessage*.Message.Data does not contain ElementItem with Name = "Configuration" and Value of tt:RecordingJobConfiguration type, FAIL the test, restore the DUT state, and skip other steps.

3. Set *recordingJobConfiguration* := [value of Value attribute from *notificationMessage*.Message.Data.ElementItem[Name = "Configuration"]].

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

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

A.56 Recording Job Information Comparison

Name: HelperComparisonOfRecordingJobs

Procedure Purpose: Helper procedure to compare of parameter values for the same recording job in different messages.

The following fields to be compared for the recording job information received/sent in different messages:

- ScheduleToken
- RecordingToken
- Mode
- Priority
- Source
  - SourceToken (this field should not be compared if AutoCreateReceiver = true was specified in one of the comparable items)
    - Type
    - Token
  - AutoCreateReceiver (this field should not be compared)
  - Tracks (order is not important for comparison, field shall not be compared if data from CreateRecordingJob is compared)
    - SourceTag
    - Destination

Recording job information will be assume as the same if information in listed fields is the same. The absence of a field in one structure and its presence in the other shall be assumed as different values.

A.57 Creation of Recording Job Prerequisite

Name: HelperRecordingJobCreationPrerequisite

Notes: Annex is used at:

- ONVIF Recording Control Device Test Specification

Procedure Purpose: Helper procedure to select or create recording for recording job creation test.

Pre-requisite:

- Recording Control Service is received from the DUT.

Input: None

Returns:

- Recording token (*recordingToken*).

Procedure:

1. ONVIF Client gets the service capabilities by following the procedure mentioned in Annex A.3 with the following input and output parameters
   - out *cap* - Recording Control Service Capabilities
2. ONVIF Client invokes **GetRecordings** request.
3. The DUT responds with **GetRecordingsResponse** message with parameters
   - RecordingItem list =: *recordingsList1*
4. If *cap*.DynamicRecording = true:
   4.1. If number of items at *recordingsList1* is less than *cap*.MaxRecordings, go to step 4.6.
   4.2. ONVIF Client invokes **DeleteRecording** request with the following parameters:
      - RecordingToken := *recordingsList1*[0].RecordingToken
   4.3. The DUT responds with **DeleteRecordingResponse** message.
   4.4. If *cap*.SupportedTargetFormats contains at least one item and *cap*.OnboardStorage = false:
      4.4.1. ONVIF Client creates external storage configuration by following the procedure mentioned in Annex A.11 with the following input and output parameters:
         - out *externalStorage1* - external storage configuration identifier
      4.4.2. Set:
         - *target1*.Storage := *externalStorage1*
         - *target1*.Format := *cap*.SupportedTargetFormats[0]
         - *target1*.Prefix := "Pre-1"
         - *target1*.Postfix := "Post-1"
         - *target1*.SpanDuration := "PT1M"

- *target1*.Encryption is skipped
- *target1*.SegmentDurationOverride is skipped

4.5. Set:

- *recordingConfiguration1*.Source.Name := [string with length 20 characters, which contains only readable characters and begins with letter]
- *recordingConfiguration1*.Source.SourceId := [valid URI with length 128 characters]
- *recordingConfiguration1*.Source.Address := [valid URI with length 128 characters]
- *recordingConfiguration1*.Source.Location := "Source Location"
- *recordingConfiguration1*.Source.Description := "Source Description"
- *recordingConfiguration1*.Content := "Content"
- *recordingConfiguration1*.MaximumRetentionTime := [acceptable MaximumRetentionTime, will be taken from Recording\Retention Time field of ONVIF Device Test Tool]
- *recordingConfiguration1*.Target := *target1* (if it was set, skipped otherwise)

4.6. ONVIF Client invokes **CreateRecording** request with parameters:

- RecordingConfiguration := *recordingConfiguration1*

4.7. The DUT responds with **CreateRecordingResponse** message with parameters:

- RecordingToken =: *recordingToken*

5. If *cap*.DynamicRecording = false:

5.1. If *recordingsList1* is empty, FAIL the test, restore the DUT settings and skip other steps.

5.2. Set *recordingToken* := *recordingsList1*[0].RecordingToken.

6. ONVIF Client deletes all recording jobs by following the procedure mentioned in Annex A.2.

Procedure Result:

PASS –

- DUT passed all assertions.

FAIL –

- DUT did not send **GetServiceCapabilitiesResponse** message.
- DUT did not send **GetRecordingsResponse** message.
- DUT did not send **DeleteRecordingResponse** message at least once.

A.58 Comparison of parameter values for the same recording in GetRecordingsResponse message and in GetRecordingConfigurationResponse message

Compare RecordingItem.Configuration item from GetRecordings response and RecordingConfiguration item from GetRecordingConfiguration response. Parameter values will be assumed as the same, if they have the same values for:

- Source.SourceId
- Source.Name
- Source.Location
- Source.Description
- Source.Address
- Content
- MaximumRetentionTime

A.59 Comparison of parameter values for the same recording job in GetRecordingJobsResponse message and in GetRecordingJobConfigurationResponse message

Compare JobConfiguration items from GetRecordingJobs response and from GetRecordingJobConfiguration response. Parameter values will be assumed as the same, if they have the same values for:

- JobConfiguration.RecordingToken
- JobConfiguration.Mode
- JobConfiguration.Priority
- JobConfiguration.Source.SourceToken.Token

- JobConfiguration.Source.SourceToken.Type
- JobConfiguration.Source.Tracks.SourceTag
- JobConfiguration.Source.Tracks.Destination

A.60 Comparison of parameter values for the same recording job in GetRecordingJobsResponse message and in GetRecordingJobStateResponse message

For comparing parameter values for the same recording job item from **GetRecordingJobsResponse** message and **GetRecordingJobStateResponse** message the following steps will be done:

1. Verify that for each RecordingJobSource from **GetRecordingJobsResponse** there is a corresponding item with the same SourceToken in RecordingJobStateInformation from **GetRecordingJobStateResponse**.
2. Verify that for each RecordingJobStateSource from **GetRecordingJobStateResponse** there is a corresponding item with the same SourceToken in RecordingJobConfiguration from **GetRecordingJobsResponse**.
3. For each pair of corresponding RecordingJobSource and RecordingJobStateSource check that for any RecordingJobTrack with some SourceTag and Destination there is a corresponding RecordingJobStateTrack with the same SourceTag and Destination in RecordingJobStateSource.

A.61 Comparison of parameter values for the same track in GetTrackConfigurationResponse message and in GetRecordingsResponse message

Compare RecordingItem.Tracks.Track.TrackToken.Configuration item from GetRecordings response and TrackConfiguration item from GetRecordingConfiguration response. Parameter values will be assumed as the same, if they have the same values for TrackType and Description.

A.62 PullMessages algorithm for check Recording Job State changing

If the state field of the RecordingJobStateInformation structure changes, the device sends the correspond event. Algorithm of **PullMessages** sending for receiving event with the expected state:

Pre-requisite: at the T1 moment the DUT received answer to request which should cause Recording Job State changing (i.e. at the T1 moment Recording Job State at the device has definitely been changed)

1. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1 to find NotificationMessage containing event with required JobState for used Recording.
2. Verify **PullMessagesResponse** message.
3. If no events are returned and CurrentTime of sending **PullMessages** is more than T1+delta, where delta is Operation delay time, skip other steps. If no events returned and CurrentTime of sending **PullMessages** is less or equal than T1+delta go to the step 1.
4. If event is returned, check UTC Time of the received event. If UTC Time is more than T1+delta, skip other steps.
5. If PropertyOperation is equal to "Changed" check event. Otherwise, go to the step 1.
6. Find event with Source.SimpleItem item with Name = "RecordingJobToken" and Value is equal to "JobToken1", Data.SimpleItem item with Name = "State" and Value is equal to required state ("Active", PartiallyActive" or "Idle").
7. If event is not found, go to the step 1, otherwise, go to the next step.
8. In the found Message check Values of State.State and State.Sources.State of Data.ElementItemDescription item. Check that values are equal to the required state ("Active", "PartiallyActive" or "Idle").

Test will be assumed as failed in case:

- The DUT did not send a valid **PullMessagesResponse** message.
- The DUT did not send NotificationMessage with event PropertyOperation = "Changed".
- There was no event which has Source.SimpleItemDescription item with Name = "RecordingJobToken" and Value is equal to "JobToken1", Data.SimpleItemDescription item with Name = "State" and Value is equal to required state.
- State.State parameter value of Data.ElementItemDescription item in NotificationMessage is not equal to required state.
- State.Sources.State parameter value of Data.ElementItemDescription item in NotificationMessage is not equal to required state.

A.63 PullMessages algorithm for check Receiver State changing

If the receiver changes state, the device sends a corresponding event. Algorithm of **PullMessages** sending for receiving event with expected state.

1. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1 to find NotificationMessage containing event with required ReceiverState for used Recording.
2. Verify **PullMessagesResponse** message.
3. If no events returned and CurrentTime of sending **PullMessages** is more than T1+delta, where delta is Operation delay time, skip other steps. If no events are returned and CurrentTime of sending **PullMessages** is less or equal to T1+delta, go to the step 1.
4. If event is returned, check UTC Time of the received event. If UTC Time is more than T1+delta, skip other steps.
5. Find event with Source.SimpleItemDescription Description item with Name = "ReceiverToken" and Value is equal to "ReceiverToken1", Data.SimpleItemDescription Description item with Name = "NewState" and Value is equal to required receiver state.
6. If event is not found, go to the step 1, otherwise, go to the next step.

Test will be assumed as failed in case:

- The DUT did not send a valid **PullMessagesResponse** message.
- The DUT did not send NotificationMessage with event which has Source.SimpleItemDescription Description item with Name = "ReceiverToken" and Value is equal to "ReceiverToken1", Data.SimpleItemDescription Description item with Name = "NewState" and Value is equal to required receiver state.

A.64 Selection or Creation of Recording for recording job creation

ONVIF Client retrieves DynamicRecording capabilities from the DUT and follows the following procedure of Recording creation or selection for using recording for recording job creation:

If DynamicRecording is True, ONVIF Client follows the following procedure of Recording creation:

1. ONVIF Client will invoke **CreateRecordingRequest** message with RecordingConfiguration.Source.SourceId as any URI,

RecordingConfiguration.Source.Name = "CameraName", RecordingConfiguration.Source.Location = "LocationDescription", RecordingConfiguration.Source.Description = "Source Description", RecordingConfiguration.Source.Address as address of the device, RecordingConfiguration.Content = "Recording from device", RecordingConfiguration.MaximumRetentionTime = "PT0S" to create a new recording.

2. Verify the **CreateRecordingResponse** message from the DUT (RecordingToken = "RecordingToken1"). If SOAP 1.2 fault message was returned, than skip other steps and follow the procedure for DynamicRecording=false case.

3. Onvif Client will invoke **GetRecordingOptionsRequest** message (RecordinToken="RecordingToken1") to get number of spare jobs that can be created for the recording.

4. Verify **GetRecordingOptionsResponse** message from the DUT. If Options.Job.Spare > RequiredSpareJobs, then return to test and use created Recording.

5. ONVIF Client will try deleting the RecordingJob via DeleteRecordingJob command (JobToken as Token of the first job received from the **GetRecordingJobsResponse** message) and repeat steps 3-5.

6. Repeare step 5 with the next Recording Job.

If DynamicRecording is False or recording creation procedure has been failed, ONVIF Client follows the following procedure of Recording selection:

1. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.

2. Verify the **GetRecordingsResponse** message from the DUT.

3. ONVIF Client will invoke **GetRecordingOptionsRequest** message (RecordingToken as Token of the first recording in the **GetRecordingsResponse** message which have track type compliant with Capabilities.Encoding value in **GetServiceCapabilitiesResponse** message) to get number of spare jobs that can be created for the recording.

4. If Options.Job.Spare >RequiredSpareJobs, then return to the test and use this Recording.

5. Repeat steps 3-4 for the next RecordingToken from the **GetRecordingsResponse** message.

6. If there is no other Recording, then ONVIF Client will try deleting the RecordingJob via DeleteRecordingJob command (JobToken as Token of the first job received from the **GetRecordingJobsResponse** message) and repeat steps3-5.

7. Repeat step 6 with the next Recording Job.

Note: Test will be assumed as failed in case:

- The DUT did not send a valid **GetRecordingsResponse** message.
- The DUT did not send a valid **GetRecordingOptionsResponse** message.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

A.65 Auto Creation of Receiver

For recording data from receiver ONVIF Client follows the following procedure of Auto Receiver creation by Create Recording Job with Idle Mode and Auto Create Receiver parameter:

1. ONVIF Client will invoke **CreateRecordingJobRequest** message (JobConfiguration.RecordingToken = "RecordingToken1", JobConfiguration.Mode = "Idle", JobConfiguration.Priority = 1, no JobConfiguration.Source.SourceToken.Token, JobConfiguration.Source.SourceToken.Type = "http://www.onvif.org/ver10/schema/Receiver", JobConfiguration.Source.AutoCreateReceiver = true) to create a recording job and auto create receiver.
2. Verify the **CreateRecordingJobResponse** message (JobToken = "JobToken1").
3. Verify that the JobConfiguration returned from CreateRecordingJob is identical to the JobConfiguration passed into CreateRecordingJob, except for the ReceiverToken and the AutoCreateReceiver.
4. Check that **CreateRecordingJobResponse** message contains JobConfiguration.Source.SourceToken.Token = ReceiverToken1 with assigned receiver Token. Check that AutoCreateReceiver field is omitted.
5. ONVIF Client will invoke **GetReceiversRequest** message to retrieve full list of receivers.
6. Verify **GetReceiversResponse** message from the DUT. Check that list of all receivers contains receiver with token equal to ReceiverToken1.

Note: If Receiver was not created for the reason Maximum Number of receivers has been reached, then delete receiver manually and run Annex's steps again.

Note: Test will be assumed as failed in case:

- The DUT sent **CreateRecordingJobResponse** message with at least one of the following JobConfiguration parameters values of which differ from the ones sent in **CreateRecordingJobRequest** message: JobConfiguration.RecordingToken, JobConfiguration.Mode, and JobConfiguration.Priority.

- The DUT sent **CreateRecordingJobResponse** message which didn't contain JobConfiguration.Source.SourceToken.Token element with assigned receiver Token
- The DUT sent JobConfiguration.Source.SourceToken.Type in **CreateRecordingJobRequest** message differs from http://www.onvif.org/ver10/schema/Receiver.
- The DUT sent **CreateRecordingJobResponse** message which contained AutoCreateReceiver field.
- **GetReceiversResponse** message did not contain Recording with Token = JobConfiguration.Source.SourceToken.Token from **CreateRecordingJobResponse** message.

A.66 Selection or Creation of Recording for recording job creation on a Media profile

ONVIF Client retrieves DynamicRecording capabilities from the DUT and follows the following procedure of Recording creation or selection for using recording for recording job creation on a Media profile:

If DynamicRecording is True, ONVIF Client follows the following procedure of Recording creation:

1. ONVIF Client will invoke **CreateRecordingRequest** message with RecordingConfiguration.Source.SourceId as any URI, RecordingConfiguration.Source.Name = "CameraName", RecordingConfiguration.Source.Location = "LocationDescription", RecordingConfiguration.Source.Description = "Source Description", RecordingConfiguration.Source.Address as address of the device, RecordingConfiguration.Content = "Recording from device", RecordingConfiguration.MaximumRetentionTime = "PT0S" to create a new recording.
2. Verify the **CreateRecordingResponse** message from the DUT (RecordingToken = "RecordingToken1"). If SOAP 1.2 fault message was returned, then skip other steps and follow the procedure for DynamicRecording=false case.
3. ONVIF Client will invoke **GetRecordingOptionsRequest** message (RecordinToken="RecordingToken1") to get Compatible Sources list and number of spare jobs that can be created for the recording.
4. Verify **GetRecordingOptionsResponse** message from the DUT.
5. If Options.Job.CompatibleSources list is empty, then ONVIF Client skips other steps and marks the test as failed.

6. If Options.Job.Spare > RequiredSpareJobs return to the test and use created Recording.
7. Client will try deleting the RecordingJob via DeleteRecordingJob command (JobToken as Token of the first job received from the **GetRecordingJobsResponse** message).
8. Repeat 3-7 steps.
9. Delete the next recording job via DeleteRecordingJob command and repeat step 8.

If DynamicRecording is False or recording creation procedure has been failed, ONVIF Client follows the following procedure of Recording selection:

1. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve a complete recordings list.
2. Verify the **GetRecordingsResponse** message from the DUT.
3. ONVIF Client will invoke **GetRecordingOptionsRequest** message (RecordingToken as Token of the first recording in the **GetRecordingsResponse** message) to get Compatible Sources list.
4. Verify **GetRecordingOptionsResponse** message from the DUT.
5. If Options.Job.CompatibleSources list is empty, then repeat 3-4 steps for the next recording. If there is no other recording, then ONVIF Client skips other steps and marks the test as failed.
6. If Options.Job.Spare > RequiredSpareJobs return to the test and use this Recording.
7. Repeat steps 3-6 for the next RecordingToken from the **GetRecordingsResponse** message.
8. If there is no other Recording then ONVIF Client will try to delete the RecordingJob via DeleteRecordingJob command (JobToken as Token of the first job received from the **GetRecordingJobsResponse** message).
9. Repeat 3-7 steps.
10. Delete the next recording job via DeleteRecordingJob command and repeat step 9.

Note: Test will be assumed as failed in case:

- The DUT did not send a valid **GetRecordingsResponse** message.
- The DUT did not send a valid **GetRecordingOptionsResponse** message.

Note: If new Recording was not created and there was no Recording with not empty Compatible Sources list then configure media profile or recording for test manually.

Note: See Annex A.45 for Recording Source Information Parameters Length limitations.

A.67 PullMessages algorithm for check Recording Job State initializing

If Recording Job is created on the device, the device sends the corresponding event with "Initialized" PropertyOperation. Algorithm of **PullMessages** sending for receiving event with the expected state is the following:

1. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1 to find NotificationMessage containing event with required JobState for used Recording.
2. Verify **PullMessagesResponse** from the DUT.
3. Repeat steps 1-2 until Notification for JobToken = JobToken1 with Data.SimpleItem item with Name = "State" and Value is equal to "Active" or "PartiallyActive" is received or operation delay time has expired.

Note: All Notification messages except messages with the expected recording job state will be ignored.