ONVIF®

Recording Search Device Test Specification

Version 20.06

June 2020

REVISION HISTORY

Vers.	Date	Description
12.12	Dec 20, 2012	First version of Recording Search Test Specification
13.06	Jun, 2013	The following test was updated: FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS
13.12	Dec, 2013	The following test was added: GET MEDIA ATTRIBUTES – PTZ MEDIA ATTRIBUTES
14.06	Jun, 2014	The following tests were updated with IDs changing: GET RECORDING SEARCH RESULTS AND GET RECORDINGS CONSISTENCY FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS) GET RECORDING SEARCH RESULTS AND GET RECORDING INFORMATION CONSISTENCY test was added The following Annex was removed: Annex A.3 GetRecordingsResponse.RecordingItem.Tracks and list of Tracks form GetRecordingSearchResultsResponse. ResultList.RecordingInformation.Track consistency The following Annex was added: Annex A.10 Check GetRecordingInformationResponse.RecordingInformation and GetRecordingSearchResultsResponse. ResultList.RecordingInformation consistency
14.12	Dec, 2014	The following tests were updated with IDs changing: FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS) FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS)
18.06	Jun, 2018	Reformatting document using new template
18.12	Sep 28, 2018	The following were updated in the scope of #1738: Sequence diagrams were removed. Annex A.10 Keep Alive Timeout Value (new item) SEARCH-2-1-3 GET RECORDING SEARCH RESULTS WITH MINRESULTS (steps 4, 9 were updated with new KeepAlive timeout) SEARCH-2-1-4 GET RECORDING SEARCH RESULTS WITH MAXRESULTS (steps 4, 9, 14 were updated with new KeepAlive timeout)

		SEARCH-2-1-5 GET RECORDING SEARCH RESULTS WITH WAITTIME (steps 3, 8 were updated with new KeepAlive timeout) SEARCH-2-1-7 FIND RECORDINGS WITH MAXMATCHES (steps 4, 9, 14 were updated with new KeepAlive timeout) SEARCH-2-1-8 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY VIDEO) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-12 GET RECORDING SEARCH RESULTS AFTER END OF SEARCH (ENDSEARCH COMMAND WAS INVOKED) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-13 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY AUDIO) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-14 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY METADATA) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-15 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (VIDEO AND AUDIO) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-16 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (VIDEO AND METADATA) (step 3 was updated with new KeepAlive timeout) SEARCH-2-1-17 GET RECORDING SEARCH RESULTS AND GET RECORDINGS CONSISTENCY (step 5 was updated with new KeepAlive timeout) SEARCH-2-1-18 GET RECORDING SEARCH RESULTS AND GET RECORDING INFORMATION CONSISTENCY (step 3 was updated with new KeepAlive timeout) SEARCH-3-1-5 FIND EVENTS (MAXMATCHES = 1) (step 6 was updated with new KeepAlive timeout) SEARCH-3-1-11 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS) (steps 8, 17 were updated with new KeepAlive timeout) SEARCH-3-1-13 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS) (steps 8, 13 were updated with new KeepAlive timeout) SEARCH-3-1-14 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS) (steps 8, 13 were updated with new KeepAlive timeout) SEARCH-5-1-1 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-5-1-2 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout)

		SEARCH-5-1-3 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-5-1-4 FIND METADATA (MAXMATCHES = 1) (step 6 was updated with new KeepAlive timeout) SEARCH-5-1-5 FIND METADATA (NO RESULTS) (step 6 was updated with new KeepAlive timeout) SEARCH-6-1-8 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-6-1-9 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-6-1-10 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS) (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-6-1-11 FIND PTZ POSITION (MAXMATCHES = 1) (step 6 was updated with new KeepAlive timeout) SEARCH-6-1-12 FIND PTZ POSITIONS USING RECORDING INFORMATION FILTER (step 3 was updated with new KeepAlive timeout) SEARCH-6-1-13 FIND PTZ POSITION – SEARCHING IN A CERTAIN POSITION (steps 6, 11 were updated with new KeepAlive timeout) SEARCH-7-1-1 GET MEDIA ATTRIBUTES – PTZ MEDIA ATTRIBUTES (step 6 was updated with new KeepAlive timeout)
18.12	Dec 21, 2018	Switching Hub description in 'Network Configuration for DUT' section was updated according to #1737
19.06	Feb 06, 2019	The following test cases were added according to #1626: RECORDINGS SEARCH - KEEP ALIVE RECORDINGS SEARCH EXPIRATION EVENTS SEARCH - KEEP ALIVE EVENTS SEARCH EXPIRATION METADATA SEARCH - KEEP ALIVE METADATA SEARCH EXPIRATION PTZ SEARCH - KEEP ALIVE PTZ SEARCH EXPIRATION
19.06	May 08, 2019	The following test cases were updated according to #1627: SEARCH-3-1-11 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING

		ENDPOINTS) (steps 17, 18, 32, 33 were added, fail criteria were updated according checks in steps) SEARCH-3-1-13 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS) (steps 13, 14 were added, fail criteria were updated according checks in steps) SEARCH-3-1-14 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS) (steps 13, 14, 15, 16, 17 were added, fail criteria were updated according checks in steps)
20.06	Dec 18, 2019	The following were updated in the scope of #1436: Reformatting document using new template and format.

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 Search 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 Search Test Specification defines and regulates the conformance testing procedure for the ONVIF conformant devices. Conformance testing is meant to be a 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 Search Test Specification covers ONVIF Recording Search service which is a functional block of [ONVIF Network Interface Specs]. The following sections give a brief overview and scope of each functional block.

1.1.1 Capabilities

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

1.1.2 Recording Search

Recording Search covers the test cases needed for the verification of recording search features as mentioned in [ONVIF Network Interface Specs]. Recording Search section defines different recording search scenarios defined by different search options.

1.1.3 Event Search

Event Search covers the test cases needed for the verification of event search features as mentioned in [ONVIF Network Interface Specs]. Event Search section defines different event search scenarios defined by different search options.

1.1.4 Metadata Search

Metadata Search covers the test cases needed for the verification of metadata search features as mentioned in [ONVIF Network Interface Specs]. Metadata Search section defines different metadata search scenarios defined by different search options.

1.1.5 PTZ Search

PTZ Search covers the test cases needed for the verification of PTZ search features as mentioned in [ONVIF Network Interface Specs]. PTZ Search section defines different PTZ search scenarios defined by different search options.

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 Search Service Specs] ONVIF Recording Search 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].

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

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.

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.

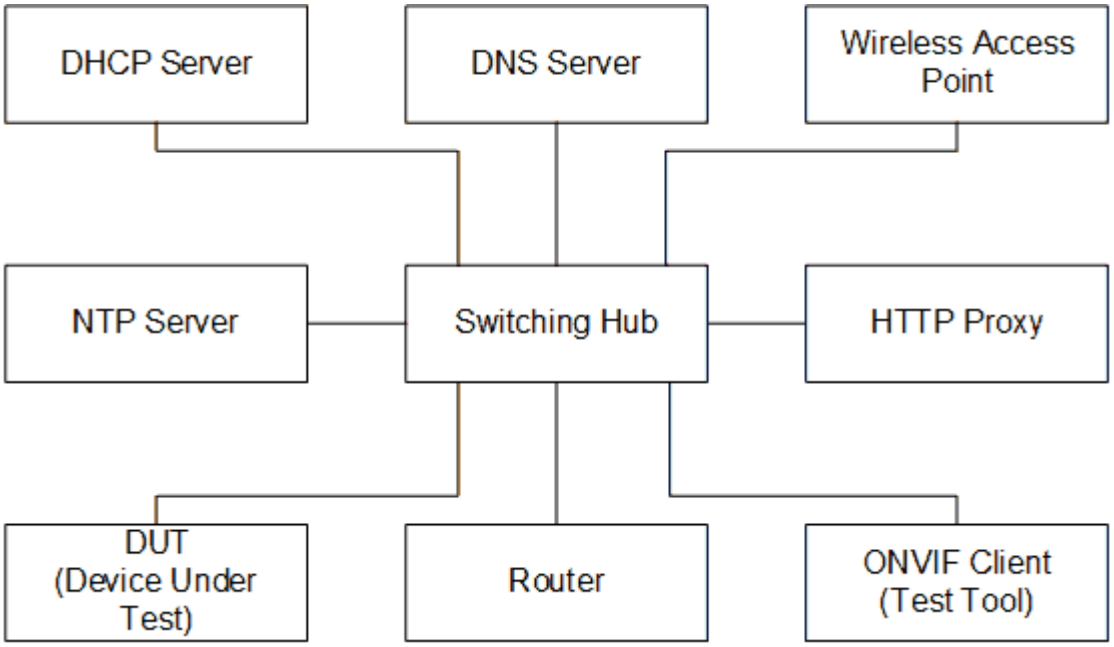
4.1 Test Setup

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 search service capability both in GetServices and GetServiceCapabilities responses. A DUT that does not display recording search service capability constitutes failure of test procedure.

Please refer to Section 5.1 for Capabilities Test Cases.

4.3.2 Recording Search

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

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

Please refer to Section 5.2 for Recording Search Test Cases.

Please refer to Annex A.1 for Recording Environment Pre-Requisite.

4.3.3 Event Search

Event search test case execution would need the recording with events. So device shall have at least one recording with events. ONVIF Client shall explicitly specify the recording token for them.

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

Please refer to Section 5.3 for Event Search Test Cases.

Please refer to Annex A.1 for Recording Environment Pre-Requisite.

4.3.4 Metadata Search

Metadata search test case execution would need the recording with metadata and metadata search support. So device shall have at least one recording with metadata and device shall support metadata search capability. ONVIF Client shall explicitly specify the recording token for them.

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

Please refer to Section 5.4 for Metadata Search Test Cases.

Please refer to Annex A.1 for Recording Environment Pre-Requisite.

4.3.5 Metadata Search

PTZ search test case execution would need the recording with PTZ metadata and PTZ search support. So device shall have at least one recording with PTZ metadata and device shall support PTZ search capability. ONVIF Client shall explicitly specify the recording token for them.

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

Please refer to Section 5.5 for PTZ Search Test Cases.

Please refer to Annex A.1 for Recording Environment Pre-Requisite.

4.3.6 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 Search Test Cases

5.1 Capabilities

5.1.1 RECORDING SEARCH SERVICE CAPABILITIES

Test Case ID: SEARCH-1-1-1

Specification Coverage: Capability exchange

Feature Under Test: GetServiceCapabilities (for Recording Search Service)

WSDL Reference: search.wsdl

Test Purpose: To verify Recording Search Service Capabilities of the DUT.

Pre-Requisite: Recording Search Service was received from the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServiceCapabilitiesRequest** message to retrieve recording search service capabilities of the DUT.
4. Verify the **GetServiceCapabilitiesResponse** from the DUT.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetServiceCapabilitiesResponse**.

5.1.2 GET SERVICES AND GET RECORDING SEARCH SERVICE CAPABILITIES CONSISTENCY

Test Case ID: SEARCH-1-1-2

Specification Coverage: Capability exchange

Feature Under Test: GetServices, GetServiceCapabilities (for Recording Search Service)

WSDL Reference: devicemgmt.wsdl, search.wsdl

Test Purpose: To verify Get Services and Recording Search Service Capabilities consistency.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetServicesRequest** message (IncludeCapability = true) to retrieve all services of the DUT with service capabilities.
4. Verify the **GetServicesResponse** message from the DUT.
5. ONVIF Client will invoke **GetServiceCapabilitiesRequest** message to retrieve Recording Search service capabilities of the DUT.
6. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetServicesResponse** message.
- The DUT did not send valid **GetServiceCapabilitiesResponse** message.
- The DUT sent different Capabilities in **GetServicesResponse** message and in **GetServiceCapabilitiesResponse** message.

Note: Service will be defined as Recording Search service if it will have Namespace element that is equal to "http://www.onvif.org/ver10/search/wsdl".

5.2 Recording Search

5.2.1 GET RECORDING SUMMARY

Test Case ID: SEARCH-4-1-1

Specification Coverage: GetRecordingSummary

Feature Under Test: GetRecordingSummary

WSDL Reference: search.wsdl

Test Purpose: To verify GetRecordingSummary command.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingSummaryRequest** message to retrieve summary information about recordings on the DUT.
4. Verify the **GetRecordingSummaryResponse** message from the DUT.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSummaryResponse** message.
- The DUT sent **GetRecordingSummaryResponse** message with DataUntil less than DataFrom.

5.2.2 GET RECORDING SEARCH RESULTS WITH MINRESULTS

Test Case ID: SEARCH-2-1-3

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that GetRecordingSearchResult command accepts any MinResults.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get current number of recordings N1 from the DUT (see Annex A.3).
4. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
5. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
6. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = N1) to get current search result and search state.
7. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
8. Repeat steps 6-7 until SearchState is equal to Completed.
9. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
10. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
11. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = N1+1) to get current search result and search state.
12. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
13. Repeat steps 11-12 until SearchState is equal to Completed.
14. Verify that sets of recordings from **GetRecordingSearchResultsResponse** messages for the first (steps 4-8) and the second (steps 9-13) recording search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at list one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation. LatestRecording or there was at list one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned a different set of recordings from **GetRecordingSearchResultsResponse**'s for steps 7 and 12 (see Annex A.4).

5.2.3 GET RECORDING SEARCH RESULTS WITH MAXRESULTS

Test Case ID: SEARCH-2-1-4

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that GetRecordingSearchResult command accepts any MaxResults and handles MaxResults parameter correctly.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get current number of recordings N1 from the DUT (see Annex A.3).
4. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10) to start recording search session and retrieve SearchToken.
5. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
6. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MaxResults = 1) to get current search result and search state.
7. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
8. Repeat steps 6-7 until SearchState is equal to Completed.
9. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
10. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
11. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MaxResults = N1) to get current search result and search state.
12. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
13. Repeat steps 11-12 until SearchState is equal to Completed.
14. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
15. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
16. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MaxResults = N1+1) to get current search result and search state.

17. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.

18. Repeat steps 16-17 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned more results in **GetRecordingSearchResultsResponse** than specified in the MaxResults field in **GetRecordingSearchResultsRequest** message.

5.2.4 GET RECORDING SEARCH RESULTS WITH WAITTIME

Test Case ID: SEARCH-2-1-5

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that GetRecordingSearchResult command accepts and honors any WaitTime.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", WaitTime = PT0.5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
7. Repeat steps 5-6 until SearchState is equal to Completed.
8. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
9. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
10. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", WaitTime = PT1S) to get current search result and search state.
11. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list which contains current results of the search with at least the following information for each recording on the list: RecordingToken, Source, Content, Track list, RecordingStatus) from the DUT.
12. Repeat steps 10-11 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT did not return **GetRecordingSearchResultsResponse** message in 1 second after specified WaitTime exceeded.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo is equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).

5.2.5 FIND RECORDINGS WITH MAXMATCHES

Test Case ID: SEARCH-2-1-7

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified MaxMatches.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

3. Get current number of recordings N1 from the DUT (see Annex A.3).
4. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], MaxMatches = 1, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
5. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
6. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1") to get current search result and search state.
7. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list) from the DUT.
8. Repeat steps 6-7 until SearchState is equal to Completed.
9. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], MaxMatches = N1 - 1, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
10. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
11. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1") to get current search result and search state.
12. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list) from the DUT.
13. Repeat steps 11-12 until SearchState is equal to Completed.
14. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], MaxMatches = N1, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
15. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
16. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1") to get current search result and search state.
17. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list) from the DUT.
18. Repeat steps 16-17 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at list one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned total number of Recordings in result greater than specified in MaxMatches for **FindRecordingsRequest** message.

Note: Steps 9-18 will be skipped if total number of recordings on the DUT equal to 1. Steps 9-13 will be skipped if total number of recordings on the DUT equal to 2.

5.2.6 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY VIDEO)

Test Case ID: SEARCH-2-1-8

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified RecordingInformationFilter (filter to fined only recordings with video track).

Pre-Requisite: At least one recording with video track exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Video"])", KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list with recordings, that contains track with video) from the DUT.
7. Repeat steps 6-7 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned Recordings without Track's with TrackType equal to "Video".

5.2.7 GET RECORDING SEARCH RESULTS WITH INVALID SEARCHTOKEN

Test Case ID: SEARCH-2-1-10

Specification Coverage: GetRecordingSearchResult

Feature Under Test: GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify Get Recording Search Result with Invalid Search Token.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

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.

5.2.8 END SEARCH WITH INVALID SEARCHTOKEN

Test Case ID: SEARCH-2-1-11

Specification Coverage: EndSearch

Feature Under Test: EndSearch

WSDL Reference: search.wsdl

Test Purpose: To verify End Search with Invalid Search Token.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

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.

5.2.9 GET RECORDING SEARCH RESULTS AFTER END OF SEARCH (ENDSEARCH COMMAND WAS INVOKED)

Test Case ID: SEARCH-2-1-12

Specification Coverage: FindRecordings, GetRecordingSearchResult, EndSearch

Feature Under Test: FindRecordings, GetRecordingSearchResult, EndSearch

WSDL Reference: search.wsdl

Test Purpose: To verify end of Recording Search session after invoking EndSearch command.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)).
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **EndSearchRequest** message (SearchToken = "Token1").
6. Verify the **EndSearchResponse** message from the DUT.
7. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1").
8. The DUT will generate SOAP 1.2 fault message (InvalidArgVal/InvalidToken).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not send a valid **EndSearchResponse** 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.

5.2.10 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY AUDIO)

Test Case ID: SEARCH-2-1-13

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified RecordingInformationFilter (filter to find only recordings with audio track).

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Audio"])", KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list with recordings, that contains track with audio) from the DUT.
7. Repeat steps 6-7 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to

RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).

- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned Recordings without Track's with TrackType equal to "Audio".
- The DUT did not return at least one Recording in search results if Audio Recording is supported.

Note: The DUT could return no results if Audio is not supported.

5.2.11 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (ONLY METADATA)

Test Case ID: SEARCH-2-1-14

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified RecordingInformationFilter (filter to find only recordings with metadata track).

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Metadata"])", KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.

5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list with recordings, that contains track with metadata) from the DUT.
7. Repeat steps 6-7 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned Recordings without Track's with TrackType equal to "Metadata".
- The DUT did not return at least one Recording in search results in case Metadata Recording is supported.

5.2.12 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (VIDEO AND AUDIO)

Test Case ID: SEARCH-2-1-15

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified RecordingInformationFilter (filter to find recordings with video and audio track).

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Video"]) and boolean(//Track[TrackType = "Audio"])", KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list with recordings, that contains track with video and audio) from the DUT.
7. Repeat steps 6-7 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to

RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).

- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned Recordings without Tracks that have TrackType equal to "Video" and "Audio".
- The DUT did not return at least one Recording in search results if Audio Recording is supported.

Note: The DUT could return no results if Audio is not supported.

5.2.13 FIND RECORDINGS WITH RECORDING INFORMATION FILTER (VIDEO AND METADATA)

Test Case ID: SEARCH-2-1-16

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify FindRecordings with specified RecordingInformationFilter (filter to find recordings with video and metadata track).

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Video"]) and boolean(//Track[TrackType = "Metadata"])", KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.

4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list with recordings, that contains track with video and metadata) from the DUT.
7. Repeat steps 6-7 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned an invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording).
- The DUT returned an invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording).
- The DUT returned Recordings without Tracks that have TrackType equal to "Video" and "Metadata".
- The DUT did not return at least one Recording in search results if Metadata Recording is Supported.

5.2.14 GET RECORDING SEARCH RESULTS AND GET RECORDINGS CONSISTENCY

Test Case ID: SEARCH-2-1-17

Specification Coverage: FindRecordings, GetRecordingSearchResults, GetRecordings

Feature Under Test: FindRecordings, GetRecordingSearchResult, GetRecordings

WSDL Reference: search.wsdl, recording.wsdl

Test Purpose: To verify Get Recording Search Result and Get Recordings consistency.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

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 list of Recordings.
4. Verify the **GetRecordingsResponse** message (RecordingItem list) from the DUT.
5. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
6. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
7. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, MaxResults = 1, WaitTime = PT5S) to get current search result and search state.
8. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list) from the DUT.
9. Repeat steps 7-8 until SearchState is equal to Completed.
10. Verify that list of Recordings from **GetRecordingsResponse** and list of Recordings from **GetRecordingSearchResultsResponse**'s are consistent.

Test Result:

PASS –

- The The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingsResponse** message, if **GetRecordingsRequest** was invoked.
- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording) in GetRecordingSearchResultResponse.
- The DUT returned invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording) in GetRecordingSearchResultResponse.
- The DUT returned list of Recordings from **GetRecordingsResponse** and list of recordings from **GetRecordingSearchResultsResponse**'s that were inconsistent (see Annex A.2).
- The DUT returned the Recording with the same token twice in one or several **GetRecordingSearchResultsResponse** messages or in **GetRecordingsResponse** message.

5.2.15 GET RECORDING SEARCH RESULTS AND GET RECORDING INFORMATION CONSISTENCY

Test Case ID: SEARCH-2-1-18

Specification Coverage: FindRecordings, GetRecordingSearchResult, GetRecordingInformation

Feature Under Test: FindRecordings, GetRecordingSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify Get Recording Search Result and Get RecordingInformation consistency.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client will invoke **FindRecordingsRequest** message (Scope = [empty], no MaxMatches, KeepAlive = keepAliveTimeout (see Annex A.10)) to start recording search session and retrieve SearchToken.
4. Verify the **FindRecordingsResponse** message (SearchToken = "Token1") from the DUT.
5. ONVIF Client will invoke **GetRecordingSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, MaxResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetRecordingSearchResultsResponse** message (SearchState = [CurrentSearchState], RecordingInformation list) from the DUT.
7. Repeat steps 5-6 until SearchState is equal to Completed.
8. ONVIF Client will invoke **GetRecordingInformationRequest** message for the first recording from **GetRecordingSearchResultsResponse**.
9. Verify **GetRecordingInformationResponse** message from the DUT. Compare RecordingInformation structure received in GetRecordingSearchResults and in GetRecordingInformation response according Annex A.9.
10. Repeat steps 8-9 for all other recording from **GetRecordingSearchResultsResponse**.

Test Result:

PASS –

- The The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **FindRecordingsResponse** message.
- The DUT did not send a valid **GetRecordingInformationResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned invalid RecordingInformation.EarliestRecording (e.g. there was no Track for this Recording where Track.DataFrom was equal to RecordingInformation.EarliestRecording or there was at least one Track for this Recording where Track.DataFrom was less than RecordingInformation.EarliestRecording) in GetRecordingSearchResultResponse.

- The DUT returned invalid RecordingInformation.LatestRecording (e.g. there was no Track for this Recording where Track.DataTo was equal to RecordingInformation.LatestRecording or there was at least one Track for this Recording where Track.DataTo was greater than RecordingInformation.LatestRecording) in GetRecordingSearchResultResponse.
- The DUT returned the Recording with the same TrackToken twice in one or several **GetRecordingSearchResultsResponse** messages.
- The DUT returned **GetRecordingInformationResponse** message with RecordingToken not equal to RecordingToken in **GetRecordingInformationRequest** message.
- The DUT returned RecordingInformation structure in **GetRecordingInformationResponse** different from RecordingInformation structure in GetRecordingSearchResultResponse for corresponding recording token (see Annex A.9).

5.2.16 RECORDINGS SEARCH - KEEP ALIVE

Test Case ID: SEARCH-2-1-19

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is kept alive during KeepAliveTime.

Pre-Requisite: Search 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 selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
4. ONVIF Client invokes **FindRecordings** request with parameters
   - Scope.IncludedSources skipped

- Scope.IncludedRecordings skipped
- Scope.RecordingInformationFilter skipped
- MaxMatches skipped
- KeepAliveTime := *keepAliveTime*

5. The DUT responds with **FindRecordingsResponse** message with parameters
   - SearchToken =: *searchToken*
6. ONVIF Client waits during (*keepAliveTime* - 1s)
7. ONVIF Client invokes **GetRecordingSearchResults** request with parameters
   - SearchToken := *searchToken*
   - MinResults skipped
   - MaxResults skipped
   - WaitTime := PT5S
8. The DUT responds with **GetRecordingSearchResultsResponse** message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **FindRecordingsResponse** message.
- The DUT did not send **GetRecordingSearchResultsResponse** message.

5.2.17 RECORDINGS SEARCH EXPIRATION

Test Case ID: SEARCH-2-1-20

Specification Coverage: FindRecordings, GetRecordingSearchResult

Feature Under Test: FindRecordings, GetRecordingSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is rejected when KeepAliveTime expired.

Pre-Requisite: Search 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 selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
4. ONVIF Client invokes **FindRecordings** request with parameters
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings skipped
   - Scope.RecordingInformationFilter skipped
   - MaxMatches skipped
   - KeepAliveTime := *keepAliveTime*
5. The DUT responds with **FindRecordingsResponse** message with parameters
   - SearchToken =: *searchToken*
6. ONVIF Client waits during (*keepAliveTime* + *timeout*)
7. ONVIF Client invokes **GetRecordingSearchResults** request with parameters
   - SearchToken := *searchToken*
   - MinResults skipped
   - MaxResults skipped
   - WaitTime := PT5S
8. The DUT responds with SOAP 1.2 fault message (**InvalidArgVal/InvalidToken**).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **FindRecordingsResponse** message.
- The DUT did not send SOAP 1.2 fault message to **GetRecordingSearchResults**.

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

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

5.3 Event Search

5.3.1 FIND EVENTS (MAXMATCHES = 1)

Test Case ID: SEARCH-3-1-5

Specification Coverage: FindEvents, GetEventSearchResult, GetRecordingInformation

Feature Under Test: FindEvents, GetEventSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify event forward and backward search in case MaxMatches = 1.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).

6. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, IncludeStartState = false, SearchFilter = [empty], MaxMatches = 1, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.
7. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
10. Repeat steps 8-9 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetEventSearchResultsResponse** message.
- The DUT did not send a valid **FindEventsResponse** message.
- The DUT did not reach the Completed state for the search session after one event was returned in **GetEventSearchResultsResponse**.
- The DUT returned more than one event in the results.

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

5.3.2 GET EVENT SEARCH RESULTS WITH INVALID SEARCHTOKEN

Test Case ID: SEARCH-3-1-10

Specification Coverage: GetEventSearchResult

Feature Under Test: GetEventSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify Get Event Search Result with Invalid Search Token.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

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.

5.3.3 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-3-1-11

Specification Coverage: FindEvents, GetEventSearchResult, GetRecordingInformation

Feature Under Test: FindEvents, GetEventSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify event forward and backward search in case the search endpoints are inside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with data from the DUT.
4. ONVIF Client will invoke **GetEventPropertiesRequest** message to retrieve all events supported by the DUT.
5. Verify the **GetEventPropertiesResponse** message from the DUT.
6. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
7. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
8. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T1 + delta1, EndPoint = T2 - delta2, Scope.IncludeRecordings = Token1, IncludeStartState = true, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.
9. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
10. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
11. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
12. Repeat steps 10-11 until SearchState is equal to Completed.
13. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Recording/State" for recording with RecordingToken = Token1.

14. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/State" for each track in recording with RecordingToken = Token1.

15. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/VideoParameters" for each video track in recording with RecordingToken = Token1, if such event is supported by the DUT.

16. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/AudioParameters" for each audio track in recording with RecordingToken = Token1, if such event is supported by the DUT.

17. Verify that DUT does not send any virtual events for other time points than StartPoint time point.

18. Verify that DUT does not send any event outside search endpoints.

19. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T2 - delta2, EndPoint = T1 + delta1, Scope.IncludeRecordings = Token1, IncludeStartState = true, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.

20. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.

21. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

22. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.

23. Repeat steps 19-20 until SearchState is equal to Completed.

24. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Recording/State" for recording with RecordingToken = Token1.

25. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/State" for each track in recording with RecordingToken = Token1.

26. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/VideoParameters" for each video track in recording with RecordingToken = Token1, if such event is supported by the DUT.

27. Verify that DUT sends virtual events for StartPoint time point with the following topics: "tns1:RecordingHistory/Track/AudioParameters" for each audio track in recording with RecordingToken = Token1, if such event is supported by the DUT.

28. Verify that DUT sends virtual events for EndPoint time point with the following topics: "tns1:RecordingHistory/Recording/State" for recording with RecordingToken = Token1.
29. Verify that DUT sends virtual events for EndPoint time point with following topics: "tns1:RecordingHistory/Track/State" for each track in recording with RecordingToken = Token1.
30. Verify that DUT sends virtual events for EndPoint time point with the following topics: "tns1:RecordingHistory/Track/VideoParameters" for each video track in recording with RecordingToken = Token1, if such event is supported by the DUT.
31. Verify that DUT sends virtual events for EndPoint time point with the following topics: "tns1:RecordingHistory/Track/AudioParameters" for each audio track in recording with RecordingToken = Token1, if such event is supported by the DUT.
32. Verify that DUT does not send any virtual events for other time points than StartPoint time point or EndPoint time point.
33. Verify that DUT does not send any event outside search endpoints.
34. Verify that sets of results from **GetEventSearchResultsResponse** messages for the first (steps 8-12) and the second (steps 17-21) event search session are the same except virtual events.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetEventSearchResultsResponse** message.
- The DUT did not send a valid **FindEventsResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned a non-virtual events with topic "tns1:RecordingHistory/Track/State" with invalid Recording token, Track token, and IsDataPresent value (IsDataPresent value shall be alternated from true to false for each Track if any of "tns1:RecordingHistory/Track/State" events exists for this Track).
- The DUT did not return a valid virtual events with topic "tns1:RecordingHistory/Recording/State" for selected recording for StartPoint or EndPoint. (For EndPoint events are required only for backward search).

- The DUT did not return a valid virtual events with topics "tns1:RecordingHistory/Track/State" for each track of selected recording for StartPoint or EndPoint. (For EndPoint events are required only for backward search).
- The DUT did not return a valid virtual events with topics "tns1:RecordingHistory/Track/VideoParameters" for each video track of selected recording for StartPoint or EndPoint. (For EndPoint events are required only for backward search).
- The DUT did not return a valid virtual events with topics "tns1:RecordingHistory/Track/AudioParameters" for each audio track of selected recording for StartPoint or EndPoint. (For EndPoint events are required only for backward search).
- The DUT did returned a non-virtual events with topic "tns1:RecordingHistory/Recording/State" with invalid Recording token, and IsRecording value (IsRecording value shall be alternated from true to false).
- The DUT returned an invalid Recording token or invalid Track token for events with following topic if they were present in search results: "tns1:RecordingHistory/Track/State", "tns1:RecordingHistory/Recording/State", "tns1:RecordingHistory/Track/VideoParameters", and "tns1:RecordingHistory/Track/AudioParameters".
- The DUT returned virtual events for other time points than StartPoint time point for forward search.
- The DUT returned virtual events for other time points than StartPoint time point or EndPoint time point for backward search.
- The DUT returned events outside search endpoints.
- The DUT did not return events in ascending order for steps 8-12 (Time of previous result was less than for the next one).
- The DUT did not return events in descending order for steps 17-21 (Time of previous result was greater than for the next one).
- The DUT did not return the same set of results (excluding virtual events) for steps 8-12 and for steps 17-21 (see Annex A.6).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: Events are supposed to be virtual events if StartStateEvent flag is equal to true.

5.3.4 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS)

Test Case ID: SEARCH-3-1-13

Specification Coverage: FindEvents, GetEventSearchResult, GetRecordingInformation

Feature Under Test: FindEvents, GetEventSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify event forward and backward search in case the search endpoints are equal to the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve list of all the tracks of recording with RecordingToken = Token1.
7. Verify the **GetRecordingsResponse** message.
8. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, IncludeStartState = false, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.

9. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
10. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
11. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
12. Repeat steps 10-11 until SearchState is equal to Completed.
13. Verify that DUT does not send any virtual events.
14. Verify that DUT does not send any event outside search endpoints.
15. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T2, EndPoint = T1, Scope.IncludeRecordings = Token1, IncludeStartState = false, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.
16. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
17. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
18. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
19. Repeat steps 15-16 until SearchState is equal to Completed.
20. Verify that sets of results from **GetEventSearchResultsResponse** messages for the first (steps 8-12) and the second (steps 13-17) event search session are the same.

Test Result:

PASS –

- The The DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetRecordingInformationResponse** message.
- The DUT did not send valid **GetRecordingsResponse** message.
- The DUT did not send a valid **GetEventSearchResultsResponse** message.
- The DUT did not send a valid **FindEventsResponse** message.

- The DUT did not reach the Completed state for the search session.
- The DUT did not return at least two events with topic "tns1:RecordingHistory/Track/State" with valid Recording token, Track token, and IsDataPresent value for each track with data (IsDataPresent value shall be alternated from true to false for each Track if any of "tns1:RecordingHistory/Track/State" events exists for this Track and event with the smallest time for each Track shall have value of IsDataPresent equal to true).
- The DUT did not return at least two events with topic "tns1:RecordingHistory/Recording/State" with valid Recording token, and IsRecording value (IsRecording value shall be alternated from true to false and event with the smallest time shall have value of IsRecording equal to true).
- The DUT returned an invalid Recording token or invalid Track token for events with following topic if they were present in search results: "tns1:RecordingHistory/Track/State", "tns1:RecordingHistory/Recording/State", "tns1:RecordingHistory/Track/VideoParameters", and "tns1:RecordingHistory/Track/AudioParameters".
- The DUT did not return events in ascending order for steps 8-12 (Time of previous result was less than for the next one).
- The DUT did not return events in descending order for steps 13-17 (Time of previous result was greater than for the next one).
- The DUT did not return the same set of results for steps 8-12 and for steps 13-17 (see Annex A.6).
- The DUT returned virtual events.
- The DUT returned events outside search endpoints.

Note: RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message then minimum value of Track.DataFrom will be used as T1.

Note: RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message then maximum value of Track.DataTo will be used as T2.

5.3.5 FIND EVENTS – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-3-1-14

Specification Coverage: FindEvents, GetEventSearchResult, GetRecordingInformation

Feature Under Test: FindEvents, GetEventSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify event forward and backward search in case the search endpoints are outside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording and tracks with data.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **GetRecordingsRequest** message to retrieve list of all the tracks of recording with RecordingToken = Token1.
7. Verify the **GetRecordingsResponse** message.
8. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T1 – delta1, EndPoint = T2 + delta2, Scope.IncludeRecordings = Token1, IncludeStartState = false, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.
9. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
10. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
11. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
12. Repeat steps 9-10 until SearchState is equal to Completed.

13. Verify that DUT does not send any virtual events.
14. Verify that DUT does not send any events outside search endpoints.
15. Verify that DUT does not send any events outside recording endpoints.
16. Verify that DUT does not send any events outside track endpoints (event is outside endpoints of any track).
17. Verify that DUT does not send any "tns1:RecordingHistory/Track/State", "tns1:RecordingHistory/Track/VideoParameters", and "tns1:RecordingHistory/Track/AudioParameters" events outside track endpoints for the track with corresponding token.
18. Verify that DUT does not send any virtual events for other time points than StartPoint time point.
19. ONVIF Client will invoke **FindEventsRequest** message (StartPoint = T2 + delta2, EndPoint = T1 – delta1, Scope.IncludeRecordings = Token1, IncludeStartState = false, SearchFilter = [empty], no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start event search session and retrieve SearchToken.
20. Verify the **FindEventsResponse** message (SearchToken = "SearchToken1") from the DUT.
21. ONVIF Client will invoke **GetEventSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
22. Verify the **GetEventSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
23. Repeat steps 15-16 until SearchState is equal to Completed.
24. Verify that sets of results from **GetEventSearchResultsResponse** messages for the first (steps 8-12) and the second (steps 13-17) event search session are the same.

Test Result:

PASS –

- The The DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetRecordingInformationResponse** message.
- The DUT did not send valid **GetRecordingsResponse** message.
- The DUT did not send valid **GetEventSearchResultsResponse** message.

- The DUT did not send valid **FindEventsResponse** message.
- The DUT did not reach Completed state for the search session.
- The DUT did not return at least two events with topic "tns1:RecordingHistory/Track/State" with valid Recording token, Track token, and IsDataPresent value for each track with data (IsDataPresent value shall be alternated from true to false for each Track if any "tns1:RecordingHistory/Track/State" event exist for this Track).
- There was event for at least one Track with topic "tns1:RecordingHistory/Track/State" with the smallest time which had ‘Changed’ property operation and value of IsDataPresent equal to false.
- The DUT did not return at least two events with topic "tns1:RecordingHistory/Recording/State" with valid Recording token, and IsRecording value (IsRecording value shall be alternated from true to false).
- Event with topic "tns1:RecordingHistory/Recording/State " with the smallest time had ‘Changed’ property operation and value of IsRecording equal to false.
- The DUT returned invalid Recording token or invalid Track token for events with following topic if they were present in search results: "tns1:RecordingHistory/Track/State", "tns1:RecordingHistory/Recording/State", "tns1:RecordingHistory/Track/VideoParameters", and "tns1:RecordingHistory/Track/AudioParameters".
- The DUT did not return events in ascending order for steps 8-12 (Time of previous result was less than for the next one).
- The DUT did not return events in descending order for steps 13-17 (Time of previous result was greater than for next one).
- The DUT did not return the same set of results for steps 8-12 and for steps 13-17 (see Annex A.6).
- The DUT returned virtual events.
- The DUT returned events outside search endpoints.
- The DUT returned events outside recording endpoints.
- The DUT returned events outside track endpoints (event is outside endpoints of any track).
- The DUT returned "tns1:RecordingHistory/Track/State", "tns1:RecordingHistory/Track/VideoParameters", or "tns1:RecordingHistory/Track/AudioParameters" events outside track endpoints for the track with corresponding token.

Note: RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message then minimum value of Track.DataFrom will be used as T1.

Note: RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message then maximum value of Track.DataTo will be used as T2.

5.3.6 EVENTS SEARCH - KEEP ALIVE

Test Case ID: SEARCH-3-1-15

Specification Coverage: FindEvents, GetEventSearchResult

Feature Under Test: FindEvents, GetEventSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is kept alive during KeepAliveTime.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. ONVIF Client invokes **GetRecordingInformation** request with parameters
   - RecordingToken := *recordingToken*
5. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source
   - RecordingInformation.EarliestRecording =: *T1*
   - RecordingInformation.LatestRecording =: *T2*

- RecordingInformation.Content
- RecordingInformation.Track
- RecordingInformation.RecordingStatus

6. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
7. ONVIF Client invokes **FindEvents** request with parameters
   - StartPoint := *T1*
   - EndPoint := *T2*
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings[0] := *recordingToken*
   - Scope.RecordingInformationFilter skipped
   - SearchFilter empty
   - IncludeStartState := false
   - MaxMatches skipped
   - KeepAliveTime := *keepAliveTime*
8. The DUT responds with **FindEventsResponse** message with parameters
   - SearchToken =: *searchToken*
9. ONVIF Client waits during (*keepAliveTime* - 1s)
10. ONVIF Client invokes **GetEventSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped
    - WaitTime := PT5S
11. The DUT responds with **GetEventSearchResultsResponse** message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindEventsResponse** message.
- The DUT did not send **GetEventSearchResultsResponse** message.

5.3.7 EVENTS SEARCH EXPIRATION

Test Case ID: SEARCH-3-1-16

Specification Coverage: FindEvents, GetEventSearchResult

Feature Under Test: FindEvents, GetEventSearchResult

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is rejected when KeepAliveTime expired.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. ONVIF Client invokes **GetRecordingInformation** request with parameters
5. RecordingToken := *recordingToken*
6. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source

- RecordingInformation.EarliestRecording =: *T1*
- RecordingInformation.LatestRecording =: *T2*
- RecordingInformation.Content
- RecordingInformation.Track
- RecordingInformation.RecordingStatus

7. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
8. ONVIF Client invokes **FindEvents** request with parameters
   - StartPoint := *T1*
   - EndPoint := *T2*
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings[0] := *recordingToken*
   - Scope.RecordingInformationFilter skipped
   - SearchFilter empty
   - IncludeStartState := false
   - MaxMatches skipped
   - KeepAliveTime := *keepAliveTime*
9. The DUT responds with **FindEventsResponse** message with parameters
   - SearchToken =: *searchToken*
10. ONVIF Client waits during (*keepAliveTime* + *timeout*)
11. ONVIF Client invokes **GetEventSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped

- WaitTime := PT5S

12. The DUT responds with SOAP 1.2 fault message (**InvalidArgVal/InvalidToken**).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindEventsResponse** message.
- The DUT did not send SOAP 1.2 fault message to **GetEventSearchResults**.

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

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

5.4 Metadata Search

5.4.1 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS)

Test Case ID: SEARCH-5-1-1

Specification Coverage: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

Feature Under Test: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify metadata forward and backward search in case the search endpoints are equal to the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with metadata from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
7. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindMetadataRequest** message at step 6.
10. Repeat steps 8-9 until SearchState is equal to Completed.
11. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T2, EndPoint = T1, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
12. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
13. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
14. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of

the search result is inside the search endpoints that were defined in **FindMetadataRequest** message at step 11.

15. Repeat steps 13-14 until SearchState is equal to Completed.
16. Verify that sets of results from **GetMetadataSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) metadata search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetMetadataSearchResultsResponse** message.
- The DUT did not send a valid **FindMetadataResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned metadata search results with invalid time (Time is not inside the search endpoints).
- The DUT returned an invalid Recording token or invalid Track token for metadata search results.
- The DUT did not return metadata search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return metadata search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.7).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: MetdataFilter1 is specified by user and assigned to data that is contained in the recording with token = Token1.

5.4.2 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-5-1-2

Specification Coverage: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

Feature Under Test: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify metadata forward and backward search in case the search endpoints are outside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with metadata data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T1 – delta1, EndPoint = T2 + delta2, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
7. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.

8. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

9. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the recording endpoints that were defined in **GetRecordingInformationResponse** at step 5.

10. Repeat steps 8-9 until SearchState is equal to Completed.

11. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T2 + delta2, EndPoint = T1 – delta1, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.

12. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.

13. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

14. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the recording endpoints that were defined in **GetRecordingInformationResponse** at step 5.

15. Repeat steps 13-14 until SearchState is equal to Completed.

16. Verify that sets of results from **GetMetadataSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) metadata search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetMetadataSearchResultsResponse** message.
- The DUT did not send a valid **FindMetadataResponse** message.
- The DUT did not reach the Completed state for the search session.

- The DUT returned metadata search results with invalid time (Time is not inside the recording endpoints).
- The DUT returned invalid Recording token or invalid Track token for metadata search results.
- The DUT did not return metadata search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return metadata search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.7).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: MetdataFilter1 is specified by user and assigned to data that is contained in the recording with token = Token1.

5.4.3 FIND METADATA – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-5-1-3

Specification Coverage: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

Feature Under Test: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify metadata forward and backward search in case the search endpoints are inside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with metadata data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T1 + delta1, EndPoint = T2 - delta2, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
7. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindMetadataRequest** message at step 6.
10. Repeat steps 8-9 until SearchState is equal to Completed.
11. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T2 - delta2, EndPoint = T1 + delta1, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
12. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
13. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

14. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindMetadataRequest** message at step 11.
15. Repeat steps 13-14 until SearchState is equal to Completed.
16. Verify that sets of results from **GetMetadataSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) metadata search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetMetadataSearchResultsResponse** message.
- The DUT did not send a valid **FindMetadataResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned metadata search results with invalid time (Time is not inside the search endpoints).
- The DUT returned invalid Recording token or invalid Track token for metadata search result.
- The DUT did not return metadata search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return metadata search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.7).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: MetdataFilter1 is specified by user and assigned to data that is contained in the recording with token = Token1.

5.4.4 FIND METADATA (MAXMATCHES = 1)

Test Case ID: SEARCH-5-1-4

Specification Coverage: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

Feature Under Test: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify metadata search in case MaxMatches = 1.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with metadata data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = MetadataFilter1, MaxMatches = 1, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
7. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of

the search result is inside the search endpoints that were defined in **FindMetadataRequest** message at step 6.

10. Repeat steps 8-9 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetMetadataSearchResultsResponse** message.
- The DUT did not send a valid **FindMetadataResponse** message.
- The DUT did not reach the Completed state for the search session after one metadata was returned in **GetMetadataSearchResultsResponse**.
- The DUT returned more than one metadata in results.

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: MetdataFilter1 is specified by user and assigned to data that is contained in the recording with token = Token1.

5.4.5 FIND METADATA (NO RESULTS)

Test Case ID: SEARCH-5-1-5

Specification Coverage: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

Feature Under Test: FindMetadata, GetMetadataSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify metadata search in case no results for the specified filter exist.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details).

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with metadata data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindMetadataRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, MetadataFilter.MetadataStreamFilter = "boolean(//Track[TrackToken = "NonExistingToken"])", MaxMatches = 1, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start metadata search session and retrieve SearchToken.
7. Verify the **FindMetadataResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetMetadataSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetMetadataSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
10. Repeat steps 8-9 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetMetadataSearchResultsResponse** message.
- The DUT did not send a valid **FindMetadataResponse** message.
- The DUT did not reach the Completed state for the search session.

- The DUT returned at least one metadata in results.

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

5.4.6 GET METADATA SEARCH RESULTS WITH INVALID SEARCHTOKEN

Test Case ID: SEARCH-5-1-6

Specification Coverage: GetMetadataSearchResults

Feature Under Test: GetMetadataSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify Get Metadata Search Result with Invalid Search Token.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

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.

5.4.7 METADATA SEARCH - KEEP ALIVE

Test Case ID: SEARCH-5-1-7

Specification Coverage: FindMetadata, GetMetadataSearchResults

Feature Under Test: FindMetadata, GetMetadataSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is kept alive during KeepAliveTime.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. Set *metadataFilter* := Metadata Filter value provided on Management tab.
5. ONVIF Client invokes **GetRecordingInformation** request with parameters
   - RecordingToken := *recordingToken*
6. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source
   - RecordingInformation.EarliestRecording =: *T1*
   - RecordingInformation.LatestRecording =: *T2*
   - RecordingInformation.Content
   - RecordingInformation.Track

- RecordingInformation.RecordingStatus

7. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
8. ONVIF Client invokes **FindMetadata** request with parameters
   - StartPoint := *T1*
   - EndPoint := *T2*
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings[0] := *recordingToken*
   - Scope.RecordingInformationFilter skipped
   - MetadataFilter.MetadataStreamFilter := *metadataFilter*
   - MaxMatches skipped
   - KeepAliveTime := *keepAliveTime*
9. The DUT responds with **FindMetadataResponse** message with parameters
   - SearchToken =: searchToken
10. ONVIF Client waits during (*keepAliveTime* - 1s)
11. ONVIF Client invokes **GetMetadataSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped
    - WaitTime := PT5S
12. The DUT responds with **GetMetadataSearchResultsResponse** message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindMetadataResponse** message.
- The DUT did not send **GetMetadataSearchResultsResponse** message.

5.4.8 METADATA SEARCH EXPIRATION

Test Case ID: SEARCH-5-1-8

Specification Coverage: FindMetadata, GetMetadataSearchResults

Feature Under Test: FindMetadata, GetMetadataSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is kept alive during KeepAliveTime.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain metadata.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. Set *metadataFilter* := Metadata Filter value provided on Management tab.
5. ONVIF Client invokes **GetRecordingInformation** request with parameters
   - RecordingToken := *recordingToken*
6. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source
   - RecordingInformation.EarliestRecording =: *T1*
   - RecordingInformation.LatestRecording =: *T2*

- RecordingInformation.Content
- RecordingInformation.Track
- RecordingInformation.RecordingStatus

7. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
8. ONVIF Client invokes **FindMetadata** request with parameters
   - StartPoint := *T1*
   - EndPoint := *T2*
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings[0] := *recordingToken*
   - Scope.RecordingInformationFilter skipped
   - MetadataFilter.MetadataStreamFilter := *metadataFilter*
   - MaxMatches skipped
   - KeepAliveTime := *keepAliveTime*
9. The DUT responds with **FindMetadataResponse** message with parameters
   - SearchToken =: *searchToken*
10. ONVIF Client waits during (*keepAliveTime* + *timeout*)
11. ONVIF Client invokes **GetMetadataSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped
    - WaitTime := PT5S
12. The DUT responds with SOAP 1.2 fault message (**InvalidArgVal/InvalidToken**).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindMetadataResponse** message.
- The DUT did not send SOAP 1.2 fault message to **GetMetadataSearchResults**.

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

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

5.5 PTZ Search

5.5.1 GET PTZ POSITION SEARCH RESULTS WITH INVALID SEARCHTOKEN

Test Case ID: SEARCH-6-1-7

Specification Coverage: GetPTZPositionSearchResults

Feature Under Test: GetPTZPositionSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify Get PTZ Position Search Result with Invalid Search Token.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

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.

5.5.2 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS EQUAL TO RECORDING ENDPOINTS)

Test Case ID: SEARCH-6-1-8

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify PTZ position forward and backward search in case the search endpoints are equal to the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.

5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).

6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.

7. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.

8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 6.

10. Repeat steps 8-9 until SearchState is equal to Completed.

11. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T2, EndPoint = T1, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.

12. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.

13. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

14. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 11.

15. Repeat steps 13-14 until SearchState is equal to Completed.

16. Verify that sets of results from **GetPTZPositionSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) PTZ position search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned PTZ position search results with invalid time (Time is not inside the search endpoints).
- The DUT returned invalid Recording token or invalid Track token for PTZ position search results.
- The DUT did not return PTZ position search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return PTZ position search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.8).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.3 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS OUTSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-6-1-9

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify PTZ position forward and backward search in case the search endpoints are outside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T1 – delta1, EndPoint = T2 + delta2, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
7. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time)

of the search result is inside the recording endpoints that were defined in **GetRecordingInformationResponse** at step 5.

10. Repeat steps 8-9 until SearchState is equal to Completed.
11. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T2 + delta2, EndPoint = T1 – delta1, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
12. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
13. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
14. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result inside the recording endpoints that were defined in **GetRecordingInformationResponse** at step 5.
15. Repeat steps 13-14 until SearchState is equal to Completed.
16. Verify that sets of results from **GetPTZPositionSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) PTZ position search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned PTZ position search results with invalid time (Time is not inside the recording endpoints).
- The DUT returned invalid Recording token or invalid Track token for PTZ position search results.

- The DUT did not return PTZ position search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return PTZ position search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.8).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.4 FIND PTZ POSITION – FORWARD AND BACKWARD SEARCH (SEARCH ENDPOINTS INSIDE RECORDING ENDPOINTS)

Test Case ID: SEARCH-6-1-10

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify PTZ position forward and backward search in case the search endpoints are inside the recording endpoints.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T1 + delta1, EndPoint = T2 - delta2, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
7. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 6.
10. Repeat steps 8-9 until SearchState is equal to Completed.
11. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T2 - delta2, EndPoint = T1 + delta1, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
12. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
13. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
14. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time)

of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 11.

15. Repeat steps 13-14 until SearchState is equal to Completed.
16. Verify that sets of results from **GetPTZPositionSearchResultsResponse** messages for the first (steps 6-10) and the second (steps 11-15) PTZ position search session are the same.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT did not reach the Completed state for the search session.
- The DUT returned PTZ position search results with invalid time (Time is not inside the search endpoints).
- The DUT returned invalid Recording token or invalid Track token for PTZ position search results.
- The DUT did not return PTZ position search results in ascending order for steps 6-10 (Time of previous result was greater than for the next one or equal to it).
- The DUT did not return PTZ position search results in descending order for steps 11-15 (Time of previous result was less than for the next one or equal to it).
- The DUT did not return the same set of results for steps 6-10 and for steps 11-15 (see Annex A.8).

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.5 FIND PTZ POSITION (MAXMATCHES = 1)

Test Case ID: SEARCH-6-1-11

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify PTZ position search in case MaxMatches = 1.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', MaxMatches = 1, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
7. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.

9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 6.

10. Repeat steps 8-9 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT returned PTZ position search results with invalid time (Time is not inside the search endpoints).
- The DUT returned invalid Recording token or invalid Track token for PTZ position search results.
- The DUT did not reach Completed state for the search session after one PTZPosition was returned in **GetPTZPositionSearchResultsResponse**.
- The DUT returned more than one PTZ position in results.

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.6 FIND PTZ POSITIONS USING RECORDING INFORMATION FILTER

Test Case ID: SEARCH-6-1-12

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResult, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify FindPTZPosition with the specified RecordingInformationFilter (filter to find only recordings with audio track).

Pre-Requisite: At least one recording with metadata track exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = 0001-01-01T00:00:00Z, Scope.RecordingInformationFilter = "boolean(//Track[TrackType = "Audio"])", SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', MaxMatches = 20, KeepAlive = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
4. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
5. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
6. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], Result list, ResultList.Result.RecordingToken = RecordingToken1, ResultList.Result.Time) from the DUT.
7. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "RecordingToken1") to retrieve information about Recording.
8. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = RecordingToken1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2, RecordingInformation.Track.TrackType = Audio). Check that this recording contains at least

one track with Audio. Check that time (ResultList.Result.Time) of search result returned at step 6 is inside the Recording endpoints.

9. Repeat steps 5-8 until SearchState is equal to Completed.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT returned PTZ position search results with invalid time (ResultList.Result.Time is not inside the Recording endpoints).
- The DUT returned Recordings without Tracks with TrackType equal to "Audio".
- The DUT did not reach the Completed state for the search session.

Note: The DUT could return no result at step 6 if it has no recordings with audio track.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.7 FIND PTZ POSITION – SEARCHING IN A CERTAIN POSITION

Test Case ID: SEARCH-6-1-13

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResult, GetRecordingInformation

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResult, GetRecordingInformation

WSDL Reference: search.wsdl

Test Purpose: To verify FindPTZPosition with specified SearchFilter (filter that uses known PTZPosition).

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T2, EndPoint = T1, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', KeepAlive = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.
7. Verify the **FindPTZPositionResponse** message (SearchToken = "Token1") from the DUT.
8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], Result.Position.PanTilt = PTZPosition1) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 6.
10. Repeat steps 8-9 until SearchState is equal to Completed.
11. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T2, EndPoint = T1, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = PTZPosition1, SearchFilter.MaxPosition.PanTilt = PTZPosition1, SearchFilter.EnterOrExit = 'False', KeepAlive = keepAliveTimeout (see Annex A.10)) to start a PTZ position search session and retrieve SearchToken.
12. Verify the **FindPTZPositionResponse** message (SearchToken = "Token2") from the DUT.
13. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "Token1", MinResults = 1, WaitTime = PT5S) to get current search result and search state.

14. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], Result.Position.PanTilt) from the DUT. Check that time (ResultList.Result.Time) of the search result is inside the search endpoints that were defined in **FindPTZPositionRequest** message at step 11.

15. Repeat steps 13-14 until SearchState is equal to Completed.

16. Check search results: The Result.Position.PanTilt shall be the same as PTZPosition1.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT returned PTZ position search results with invalid time (Time is not inside the search endpoints).
- The DUT returned invalid Recording token or invalid Track token for PTZ position search results.
- The DUT did not reach the Completed state for the search session.
- The DUT returned at step 14 the Position that differs from PTZPosition1.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

5.5.8 PTZ SEARCH - KEEP ALIVE

Test Case ID: SEARCH-6-1-14

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is rejected when KeepAliveTime expired.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. ONVIF Client invokes **GetRecordingInformation** request with parameters
   - RecordingToken := *recordingToken*
5. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source
   - RecordingInformation.EarliestRecording =: *T1*
   - RecordingInformation.LatestRecording =: *T2*
   - RecordingInformation.Content
   - RecordingInformation.Track
   - RecordingInformation.RecordingStatus
6. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
7. ONVIF Client invokes **FindPTZPosition** request with parameters
   - StartPoint := *T1*
   - EndPoint := *T2*
   - Scope.IncludedSources skipped
   - Scope.IncludedRecordings[0] := *recordingToken*

- Scope.RecordingInformationFilter skipped
- SearchFilter.MinPosition.PanTilt := [-1,-1]
- SearchFilter.MaxPosition.PanTilt := [1, 1]
- SearchFilter.EnterOrExit := false
- MaxMatches skipped
- KeepAliveTime := *keepAliveTime*

8. The DUT responds with **FindPTZPositionResponse** message with parameters
   - SearchToken =: *searchToken*
9. ONVIF Client waits during (*keepAliveTime* - 1s)
10. ONVIF Client invokes **GetPTZPositionSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped
    - WaitTime := PT5S
11. The DUT responds with **GetPTZPositionSearchResultsResponse** message.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindPTZPositionResponse** message.
- The DUT did not send **GetPTZPositionSearchResultsResponse** message.

5.5.9 PTZ SEARCH EXPIRATION

Test Case ID: SEARCH-6-1-15

Specification Coverage: FindPTZPosition, GetPTZPositionSearchResults

Feature Under Test: FindPTZPosition, GetPTZPositionSearchResults

WSDL Reference: search.wsdl

Test Purpose: To verify that search session is kept alive during KeepAliveTime.

Pre-Requisite: Search Service is received from the DUT. At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Set *recordingToken* := token of recording provided on Management tab.
4. ONVIF Client invokes **GetRecordingInformation** request with parameters
   - RecordingToken := *recordingToken*
5. The DUT responds with **GetRecordingInformationResponse** message with parameters
   - RecordingInformation.RecordingToken = *recordingToken*
   - RecordingInformation.Source
   - RecordingInformation.EarliestRecording =: *T1*
   - RecordingInformation.LatestRecording =: *T2*
   - RecordingInformation.Content
   - RecordingInformation.Track
   - RecordingInformation.RecordingStatus
6. ONVIF Client selects keep alive time value by following the procedure mentioned in Annex A.10 with the following output parameter
   - out *keepAliveTime* – keep alive time
7. ONVIF Client invokes **FindPTZPosition** request with parameters

- StartPoint := *T1*
- EndPoint := *T2*
- Scope.IncludedSources skipped
- Scope.IncludedRecordings[0] := *recordingToken*
- Scope.RecordingInformationFilter skipped
- SearchFilter.MinPosition.PanTilt := [-1,-1]
- SearchFilter.MaxPosition.PanTilt := [1, 1]
- SearchFilter.EnterOrExit := false
- MaxMatches skipped
- KeepAliveTime := *keepAliveTime*

8. The DUT responds with **FindPTZPositionResponse** message with parameters
   - SearchToken =: *searchToken*
9. ONVIF Client waits during (*keepAliveTime* + *timeout*)
10. ONVIF Client invokes **GetPTZPositionSearchResults** request with parameters
    - SearchToken := *searchToken*
    - MinResults skipped
    - MaxResults skipped
    - WaitTime := PT5S
11. The DUT responds with SOAP 1.2 fault message (**InvalidArgVal/InvalidToken**).

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send **GetRecordingInformationResponse** message.
- The DUT did not send **FindPTZPositionResponse** message.

- The DUT did not send SOAP 1.2 fault message to **GetPTZPositionSearchResults**.

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

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

5.6 Media Attributes

5.6.1 GET MEDIA ATTRIBUTES – PTZ MEDIA ATTRIBUTES

Test Case ID: SEARCH-7-1-1

Specification Coverage: GetMediaAttributes

Feature Under Test: GetMediaAttributes

WSDL Reference: search.wsdl

Test Purpose: To verify PTZ Media Attributes.

Pre-Requisite: At least one recording exists on the DUT (see Annex A.1 for more details). The recording shall contain PTZ metadata and that the matching PTZ search criterion is present in the recording.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Get token Token1 of recording which contains at least one track with PTZ position data from the DUT.
4. ONVIF Client will invoke **GetRecordingInformationRequest** message (RecordingToken = "Token1") to retrieve information about Recording.
5. Verify the **GetRecordingInformationResponse** message (RecordingInformation.RecordingToken = Token1, RecordingInformation.EarliestRecording = T1, RecordingInformation.LatestRecording = T2).
6. ONVIF Client will invoke **FindPTZPositionRequest** message (StartPoint = T1, EndPoint = T2, Scope.IncludeRecordings = Token1, SearchFilter.MinPosition.PanTilt = [-1,-1], SearchFilter.MaxPosition.PanTilt = [1, 1], SearchFilter.EnterOrExit = 'False', no

MaxMatches, KeepAliveTime = keepAliveTimeout (see Annex A.10)) to start PTZ positions search session and retrieve SearchToken.

7. Verify the **FindPTZPositionResponse** message (SearchToken = "SearchToken1") from the DUT.
8. ONVIF Client will invoke **GetPTZPositionSearchResultsRequest** message (SearchToken = "SearchToken1", WaitTime = PT5S) to get current search result and search state.
9. Verify the **GetPTZPositionSearchResultsResponse** message (SearchState = [CurrentSearchState], ResultList) from the DUT.
10. Repeat steps 8-9 until SearchState is equal to Completed or at least one Result was received.
11. If SearchState after step 10 was not Completed, ONVIF Client will invoke EndSearchRequst message (SearchToken = "SearchToken1") to stop searching. Otherwise if Completed state was reached, but no Result was received, fail the test and skip other steps.
12. Select one of received Result and use it for other steps.
13. ONVIF Client will invoke **GetMediaAttributesRequest** message (RecordingToken = "Token1", Time = Result.Time) to media attributes with PTZ media attributes.
14. Verify the **GetMediaAttributesResponse** message (RecordingToken = "Token1", TrackAttributes.MetadataAttributes.CanContainPTZ = true, TrackAttributes.MetadataAttributes.PtzSpaces, other media attributes) from the DUT.
15. Verify that TrackAttributes.MetadataAttributes.CanContainPTZ is equal to true for the Track with TrackInformation.TrackToken = Result.TrackToken.
16. If Prfile G is claimed as supported by the DUT, verify that TrackAttributes.MetadataAttributes.PtzSpaces for the Track with TrackInformation.TrackToken = Result.TrackToken contains PanTilt Position Generic namespace, if Result contains Position.PanTilt.
17. If Prfile G is claimed as supported by the DUT, verify that TrackAttributes.MetadataAttributes.PtzSpaces for the Track with TrackInformation.TrackToken = Result.TrackToken contains Zoom Position Generic namespace, if Result contains Position.Zoom.

Test Result:

PASS –

- The DUT passes all assertions.

FAIL –

- The DUT did not send a valid **GetRecordingInformationResponse** message.
- The DUT did not send a valid **GetPTZPositionSearchResultsResponse** message.
- The DUT did not send a valid **FindPTZPositionResponse** message.
- The DUT did not send a valid **EndSearchResponse** message, if **EndSearchRequest** was invoked.
- The DUT did not send a valid **GetMediaAttributesResponse** message.
- The DUT did not send a TrackAttributes.MetadataAttributes.CanContainPTZ is equal to true for the Track with TrackInformation.TrackToken = Result.TrackToken.
- The DUT did not send a TrackAttributes.MetadataAttributes.PtzSpaces for the Track with TrackInformation.TrackToken = Result.TrackToken contains PanTilt Position Generic namespace, if Result contains Position.PanTilt and Profile G is claimed as supported bu the DUT.
- The DUT did not send a TrackAttributes.MetadataAttributes.PtzSpaces for the Track with TrackInformation.TrackToken = Result.TrackToken contains Zoom Position Generic namespace, if Result contains Position.Zoom and Profile G is claimed as supported bu the DUT.

Note: If RecordingInformation.EarliestRecording is not included in **GetRecordingInformationResponse** message, then minimum value of Track.DataFrom will be used as T1.

Note: If RecordingInformation.LatestRecording is not included in **GetRecordingInformationResponse** message, then maximum value of Track.DataTo will be used as T2.

Note: If **GetPTZPositionSearchResultsResponse** contains responses with Position.PanTilt out of requested range, ONVIF Device Test Tool will report it as warning.

Note: The following Position Generic namespaces are defined for PanTilt and Zoom:

- http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace
- http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace

Annex A Helper Procedures and Additional Notes

A.1 Recording Environment Pre-Requisite

The following pre-requisites shall have all fields filled in before executing tests for Replay Service:

1. At least one recording shall be present at the DUT and specified for test execution.
2. The recording shall be stopped before executing tests.
3. The recording shall contain at least one track each for video, audio (if Audio Recording is supported) and Metadata (if Metadata Recording is supported).
4. The recording shall contain data at least in the video track, audio track (if Audio Recording is supported), and metadata track (if Metadata Recording is supported) and configuration of each track shall be consistent for the duration of the recording.
5. The recording shall contain at least one gap.
6. The recording span (including gap) should be not more than 3 minutes.
7. Each recording event should be at least one minute for the recording.

A.2 Check list of Recordings from GetRecordingsResponse and list of recordings form GetRecordingSearchResultsResponses

To check the list of Recordings from **GetRecordingsResponse** and the list of recordings from **GetRecordingSearchResultsResponse**'s consistency the following steps will be done:

1. Verify that **GetRecordingsResponse** contains the same number of RecordingItem's with the total number of RecordingInformation's which were in **GetRecordingSearchResultsResponse**'s. If number of recordings is different for these two cases, fail the check and skip the next steps.
2. Verify that for each RecordingItem in **GetRecordingsResponse** message there is RecordingInformation item in one of the **GetRecordingSearchResultsResponse** messages with RecordingInformation.RecordingToken equal to RecordingItem.RecordingToken. If this verification does not pass, fail the check and skip the next steps.
3. Verify that if RecordingInformation.RecordingToken is equal to RecordingItem.RecordingToken for RecordingItem from **GetRecordingsResponse** and for

RecordingInformation in one of the **GetRecordingSearchResultsResponse**'s the following statements are true:

3.1. RecordingInformation.Source and RecordingItem.Source are equal to each other (e.g. have the same values for each field).

3.2. RecordingInformation.Content and RecordingItem.Content are equal to each other.

If one of the statements is not true, fail the check.

A.3 Get Total Number of Recordings from the DUT

The following procedure will be used to get the total number of recordings from the DUT:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetRecordingSummaryRequest** message to get the total number of recordings.
4. Verify the **GetRecordingSummaryResponse** message (DataFrom, DataUntil, NumberRecordings) from the DUT.
5. Summary.NumberRecordings value from **GetRecordingSummaryResponse** message will be assumed as the total number of Recordings on the DUT.

A.4 Check that DUT return same set of recordings for two search recording sessions

To check that the list of Recordings from **GetRecordingSearchResultsResponse** messages for two search recording sessions contains the same set of Recordings the following steps will be done:

1. Verify that **GetRecordingSearchResultsResponse** messages for the first search recording session contain the same number of RecordingInformation items with **GetRecordingSearchResultsResponse** messages for the second search recording session. If the number of items is different for these two cases, fail the check and skip the next steps.
2. Verify that for each result in **GetRecordingSearchResultsResponse** messages in the first search recording session, there is the same result in **GetRecordingSearchResultsResponse** messages in the second search recording

session (e.g. result with the same RecordingToken). If this verification does not pass, fail the check and skip the next steps.

3. Verify that the following statements are true if RecordingInformation.RecordingToken is equal for two RecordingInformation items from the first and the second search recording session:

   3.1. RecordingInformation.Source items are equal to each other (e.g. have the same values for each field).

   3.2. RecordingInformation.EarliestRecording items are equal to each other or not included for both RecordingInformation items.

   3.3. RecordingInformation.LatestRecording items are equal to each other or not included for both RecordingInformation items.

   3.4. RecordingInformation.Content items are equal to each other.

   3.5. RecordingInformation.Track lists are equal to each other (see Annex A.5).

   3.6. RecordingInformation.RecordingStatus items are equal to each other.

If one of the statements is not true, fail the check.

A.5 Check that DUT return same set of tracks for two RecordingInformation items

To check that the list of Track items for two RecordingInformation items contains the same set of Tracks the following steps will be done:

1. Verify that RecordingInformation items contain the same number of Track items. If the number of items is different, fail the check and skip the next steps.

2. Verify that for each RecordingInformation.Track item in the first RecordingInformation item, there is the same Track item in the second RecordingInformation item (e.g. Track items with the same TrackToken). If this verification does not pass, fail the check and skip the next steps.

3. Verify that the following statements are true if Track.TrackToken is equal for two Track items from the first and the second RecordingInformation item:

   3.1. Track.TrackType items are equal to each other.

   3.2. Track.Description items are equal to each other.

3.3. Track.DataFrom items are equal to each other.

3.4. Track.DataTo items are equal to each other.

If one of statements is not true fail the check.

A.6 Check that DUT return same set of event Results for two search event sessions

To check that the list of Result items from **GetEventSearchResultsResponse** messages for two search event sessions contains the same set of Result items the following steps will be done:

1. Verify that **GetEventSearchResultsResponse** messages for the first search event session contain the same number of Result items with **GetEventSearchResultsResponse** messages for the second search event session. If the number of items is different for these two cases, fail the check and skip the next steps.

2. Verify that for each Result item in **GetEventSearchResultsResponse** messages in the first search event session, there is the same Result item in **GetEventSearchResultsResponse** messages in the second search event session. If this verification does not pass, fail the check and skip the next steps.

3. Verify that the following statements are true if two Result items are the same:

   3.1. Result.Event.SubscriptionReference items are equal to each other (e.g. have the same values for each field).

   3.2. Result.Event.ProducerReference items are equal to each other (e.g. have the same values for each field).

   3.3. Result.Event.Message items are equal to each other (e.g. have the same values for each field).

   3.4. Result.StartStateEvent items are equal to each other.

If one of the statements is not true, fail the check.

Note: Two Event Results will be assumed as the same, if they have the same values for GetEventSearchResultsResponse.ResultList.Result.RecordingToken, GetEventSearchResultsResponse.ResultList.Result.TrackToken, GetEventSearchResultsResponse.ResultList.Result.Time, GetEventSearchResultsResponse.ResultList.Result.Event.Topic and fields and values of GetEventSearchResultsResponse.ResultList.Result.Event.Message.Data.

A.7 Check that DUT returns the same set of metadata Results for two search metadata sessions

To check that the list of Result items from **GetMetadataSearchResultsResponse** messages for two search metadata sessions contains the same set of Result items the following steps will be done:

1. Verify that **GetMetadataSearchResultsResponse** messages for the first search metadata session contain the same number of Result items with **GetMetadataSearchResultsResponse** messages for the second search metadata session. If the number of items is different for these two cases, fail the check and skip the next steps.

2. Verify that for each Result item in **GetMetadataSearchResultsResponse** messages in the first search metadata session, there is the same Result item in **GetMetadataSearchResultsResponse** messages in the second search metadata session. If this verification does not pass, fail the check and skip the next steps.

Note: two Metadata Results will be assumed as the same, if they have the same values for: GetMetadataSearchResultsResponse.ResultList.Result.RecordingToken, GetMetadataSearchResultsResponse.ResultList.Result.TrackToken, GetMetadataSearchResultsResponse.ResultList.Result.Time

A.8 Check that DUT returns the same set of PTZ Position Results for two search PTZ position sessions

To check that the list of Result items from **GetPTZPositionSearchResultsResponse** messages for two search PTZ Position sessions contains the same set of Result items the following steps will be done:

1. Verify that **GetPTZPositionSearchResultsResponse** messages for the first search PTZ Position session contain the same number of Result items with **GetPTZPositionSearchResultsResponse** messages for the second search PTZ Position session. If the number of items is different for these two cases, fail the check and skip the next steps.

2. Verify that for each Result item in Get PTZPositionSearchResultsResponse messages in the first search PTZ Position session, there is the same Result item in Get PTZPositionSearchResultsResponse messages in the second search PTZ Position session. If this verification does not pass, fail the check and skip the next steps.

Note: Two PTZ Position Results will be assumed as the same, if they have the same values for: GetPTZPositionSearchResultsResponse.ResultList.Result.RecordingToken, GetPTZPositionSearchResultsResponse.ResultList.Result.TrackToken, GetPTZPositionSearchResultsResponse.ResultList.Result.Time, GetPTZPositionSearchResultsResponse.ResultList.Result.Pozition.

A.9 Check GetRecordingInformationResponse.RecordingInformation and GetRecordingSearchResultsResponse.ResultList.RecordingInformation consistency

To check the RecordingInformation structure consistency from **GetRecordingInformationResponse** and from **GetRecordingSearchResultsResponse** ONVIF Client will compare GetRecordingInformationResponse.RecordingInformation structure with GetRecordingSearchResultsResponse.ResultList.RecordingInformation stucture for corresponding RecordingToken. The following steps will be done:

1. Verify that GetRecordingInformationResponse.RecordingInformation.Source.SourceID equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Source.SourceID.
2. Verify that GetRecordingInformationResponse.RecordingInformation.Source.Name equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Source.Name.
3. Verify that GetRecordingInformationResponse.RecordingInformation.Source.Location equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Source.Location.
4. Verify that GetRecordingInformationResponse.RecordingInformation.Source.Description equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Source.Description.
5. Verify that GetRecordingInformationResponse.RecordingInformation.Source. Address equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Source.Address.
6. Verify that if RecordingInformation.EarliestRecording is skipped in **GetRecordingInformationResponse**, then ResultList.RecordingInformation.EarliestRecording is skipped in **GetRecordingSearchResultsResponse**.

7. If RecordingInformation.EarliestRecording is not skipped in **GetRecordingInformationResponse**, then verify that GetRecordingInformationResponse.EarliestRecording equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.EarliestRecording.

8. Verify that if RecordingInformation.LatestRecording is skipped in **GetRecordingInformationResponse**, then ResultList.RecordingInformation.LatestRecording is skipped in **GetRecordingSearchResultsResponse**.

9. If RecordingInformation.LatestRecording is not skipped in **GetRecordingInformationResponse**, then verify that GetRecordingInformationResponse.LatestRecording equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.EarliestRecording.

10. Verify that GetRecordingInformationResponse.RecordingInformation.Content equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.Content.

11. Verify that GetRecordingInformationResponse.RecordingInformation contains the same number of Tracks with number of Tracks from GetRecordingSearchResultsResponse.ResultList.RecordingInformation item. If number of tracks is different for these two cases, fail the check and skip the next steps.

12. Verify that for each RecordingInformation.Track.TrackToken in **GetRecordingInformationResponse** message, there is RecordingInformation.Track item in the **GetRecordingSearchResultsResponse** messages with TrackToken equal to RecordingInformation.Track.TrackToken. If this verification does not pass, fail the check and skip the next steps.

13. Verify that if RecordingInformation.Track.TrackToken from **GetRecordingInformationResponse** is equal to RecordingInformation.Track.TrackToken in **GetRecordingSearchResultsResponse**, the following statements are true:

    13.1. RecordingInformationTrack.TrackType and RecordingInformation.Track.TrackType are equal to each other.

    13.2. RecordingInformationTrack.Description and RecordingInformation.Track.Description are equal to each other.

    13.3. RecordingInformationTrack.DataFrom and RecordingInformation.Track.DataFrom are equal to each other.

    13.4. RecordingInformationTrack.DataTo and RecordingInformation.Track.DataTo are equal to each other.

14. Verify that GetRecordingInformationResponse.RecordingInformation.RecordingStatus equals to GetRecordingSearchResultsResponse.ResultList.RecordingInformation.RecordingStatus.

If one of the statements is not true, fail the check.

A.10 Keep Alive Timeout Value

Keep Alive Timeout Value (*keepAliveTimeout*) will be taken as maximum value between 10 seconds and *timeBetweenRequests* + 3 second (where *timeBetweenRequests* is Time Between Requests field of ONVIF Device Test Tool). The *keepAliveTimeout* shall be in xs:duration format.