ONVIF™

Event Handling Test Specification

Version 18.06

June 2018

REVISION HISTORY

Vers.	Date	Description
17.06	Jun 15, 2017	First issue. Was created by splitting Base Test Specification.
17.12	Sept 19, 2017	The following test cases were added according to #1480: EVENT-2-1-28 BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE EVENT-3-1-36 REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE The following test cases were changed according to #1480: EVENT-2-1-21 renamed from BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE to BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE (NEGATIVE TEST) EVENT-3-1-19 renamed from REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE to REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE (NEGATIVE TEST)
17.12	Oct 30, 2017	The following test case was added according to #1478: EVENT-3-1-37 REALTIME PULLPOINT SUBSCRIPTION – MAXIMUM SUPPORTED NUMBER OF NOTIFICATION PULL POINTS
17.12	Oct 31, 2017	The following test case was updated according to #1401: EVENT-3-1-12 REALTIME PULLPOINT SUBSCRIPTION – RENEW EVENT-3-1-19 REALTIME PULLPOINT SUBSCRIPTION – UNSUBSCRIBE EVENT-3-1-20 REALTIME PULLPOINT SUBSCRIPTION – TIMEOUT
18.06	Apr 17, 2018	The following test cases were added according to #1340: EVENT-2-1-29 BASIC NOTIFICATION INTERFACE - MESSAGE CONTENT FILTER EVENT-3-1-38 REALTIME PULLPOINT SUBSCRIPTION - MESSAGE CONTENT FILTER
18.06	Jun 21, 2018	Reformatting document using new template

Table of Contents

1 Introduction

The goal of the ONVIF test specification set is to make it possible to realize fully interoperable IP physical security implementations from different vendors. The set of ONVIF test specifications describes the test cases need to verify the [ONVIF Network Interface Specs] and [ONVIF Conformance] requirements. Also the test cases are to be basic inputs for some Profile specification requirements. It also describes the test framework, test setup, pre-requisites, test policies needed for the execution of the described test cases.

This ONVIF Event Handling Specification acts as a supplementary document to the [ONVIF Network Interface Specs], illustrating test cases that need to be executed and passed. And also this specification also 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. As the test tool performs as a Client during testing, this test tool is referred as ONVIF Client hereafter.

1.1 Scope

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

The principal intended purposes are:

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

This specification does not address the following:

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

This ONVIF Event Handling Test Specification covers core parts of functional blocks in [ONVIF Network Interface Specs]. The following sections describe the brief overview and scope of each functional block.

1.1.1 Events

Event handling covers the test cases for the verification of the Event service as mentioned in [ONVIF Network Interface Specs] and [ONVIF Event WSDL].

The event handling test cases cover the following mandatory interfaces:

- Basic Notification Interface
  - This test specification provides test cases to verify the implementation of the Basic Notification Interface of a device. The mechanisms to subscribe and unsubscribe to an event are covered as well as the mechanism to renew a subscription manager and receive events via **Notify** messages. The correct use of the message content filter is also tested.
- Real Time Pull Point Notification Interface
  - Test cases to verify the implementation of the Realtime PullPoint Interface are provided by this test specification. The CreatePullPointSubscription command is tested as well as the PullMessages command in combination with message content filtering to retrieve the events.
- Seek
  - Test cases to verify the implementation of the Seek Interface are provided by this test specification. The Seek command is tested as well as the PullMessages command to retrieve the events from persistent notification storage.

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/

- [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

- [RFC 3986] "Uniform Resource Identifier (URI): Generic Syntax", T. Berners-Lee et. al., January 2005.:

  http://www.ietf.org/rfc/rfc3986

3 Terms and Definitions

3.1 Conventions

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

3.2 Definitions

This section describes terms and definitions used in this document.

Profile	See ONVIF Profile Policy.
ONVIF Device	Computer appliance or software program that exposes one or multiple ONVIF Web Services.
ONVIF Client	Computer appliance or software program that uses ONVIF Web Services.
SOAP	SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.
Device Test Tool	ONVIF Device Test Tool that tests ONVIF Device implementation towards the ONVIF Test Specification set.
Address	An address refers to a URI
Capability	The capability commands allow a client to ask for the services provided by an ONVIF device.
Network	A network is an interconnected group of devices communicating using the Internet protocol.
Proxy Server	A server that services the requests of its clients by forwarding requests to other servers. A Proxy provides indirect network connections to its clients.
Switching Hub	A device for connecting multiple Ethernet devices together, making them act as a single network segment.
Target Service	An endpoint that makes itself available for discovery.

3.3 Abbreviations

This section describes abbreviations used in this document.

DUT	Device Under Test
DP	Discovery Proxy
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol

HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transport Protocol over Secure Socket Layer
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
NTP	Network Time Protocol
POSIX	Portable Operating System Interface
RTSP	Real Time Streaming Protocol
RTP	Real-time Transport Protocol
UTC	Coordinated Universal Time
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 describes the test setup procedure and prerequisites needed, and the test policies that should be followed for test case execution.

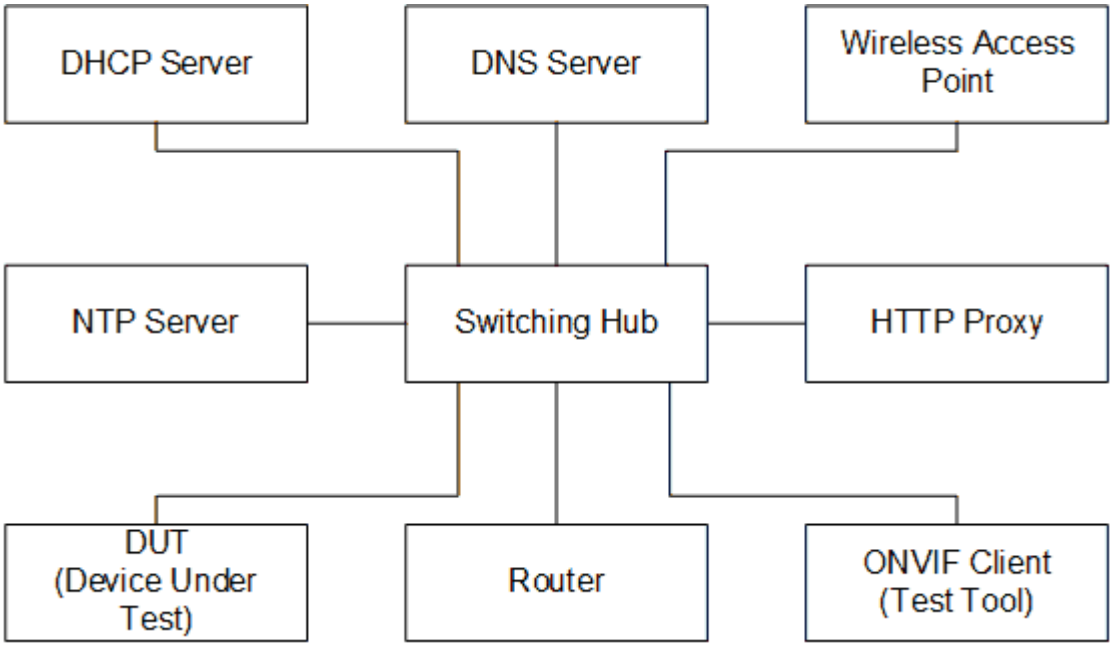
4.1 Test Setup

4.1.1 Network Configuration for DUT

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

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

Figure 4.1. Test Configuration for DUT



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

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

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

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

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

DHCP Server: provides IPv4 Address to the connected devices.

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

Switching Hub: provides network connectivity among all the test equipments in the test environment. All devices should be connected to the Switching Hub.

Router: provides router advertisements for IPv6 configuration.

4.2 Prerequisites

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

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

4.3 Test Policy

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

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

4.3.1 Event Handling

Prior to the execution of Event handling test cases, DUT shall be discovered by the ONVIF Client and it shall demonstrate event capabilities to ONVIF Client using the device management service.

If the DUT supports "property events" the ONVIF Client uses the SetSynchronizationPoint method from the event service to trigger events for testing Basic Notification Interface, Realtime PullPoint Interface; the ONVIF Client uses the SetSynchronizationPoint from the Media service to trigger events for testing metadata streaming.

If the DUT does not support "property events", the event should be triggered manually.

The time of the ONVIF Client and the DUT should be synchronized.

In certain test cases the Test Tool may create a Subscription Manager representing the subscription. In such cases the procedure will make sure that all newly created Subscription Managers are deleted at the end of the test procedure.

In certain test cases the Test Tool may create or change media entities (e.g. add a MetadataConfiguration to a profile). In such cases the procedure will delete those modifications at the end of the test procedure.

Refer to Section 5 for Event handling Test Cases.

5 Event Handling Test Cases

5.1 Event Properties

5.1.1 GET EVENT PROPERTIES

Test Case ID: EVENT-1-1-2

Specification Coverage: GetEventProperties

Feature under test: GetEventProperties

WSDL Reference: event.wsdl

Test Purpose: To verify DUT GetEventProperties 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 **GetEventProperties** request to retrieve information about the FilterDialects, schema files and supported topics of the DUT.
4. Verify that DUT sends **GetEventPropertiesResponse** message.
5. Validate that the mandatory TopicExpressionDialects are supported by the DUT (http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete and http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet).
6. Validate that the mandatory MessageContentFilterDialects is supported by the DUT (http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter).
7. Verify that the DUT returns a valid topic namespace.
8. Verify that the DUT supports at least one TopicSet, validate that the TopicSet is well formed.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse** message.
- The DUT did not support the mandatory TopicExpressionDialects.
- The DUT did not support the mandatory MessageContentFilterDialects.
- The DUT did not support a valid topic namespace.
- The DUT did not support at least one TopicSet or the TopicSet is invalid.
- The DUT did not send a valid WS-Addressing Action URI in SOAP Header for **GetEventPropertiesResponse** message (see Annex A.3).

5.2 Basic Notification Interface

5.2.1 BASIC NOTIFICATION INTERFACE - SUBSCRIBE

Test Case ID: EVENT-2-1-9

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify DUT Subscribe 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 **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends **SubscribeResponse** message. Verify that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values

for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not return a valid SubscriptionReference.
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

5.2.2 BASIC NOTIFICATION INTERFACE - RENEW

Test Case ID: EVENT-2-1-12

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe, Renew

WSDL Reference: event.wsdl

Test Purpose: To verify Renew 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 **Subscribe** request with an InitialTerminationTime of 10s.
4. Verify that the DUT sends a **SubscribeResponse**.
5. Validate CurrentTime and TerminationTime (TerminationTime >= CurrentTime + InitialTerminationTime).
6. ONVIF Client will invoke **Renew** request with a TerminationTime of 10s to ensure that the Subscription times out after 10s.
7. Verify that the DUT sends a **RenewResponse** message.
8. Verify CurrentTime and TerminationTime (TerminationTime >= CurrentTime + TerminationTime).
9. ONVIF Client will invoke **Renew** request with a TerminationTime in xs:dateTime format. The TerminationTime shall be current time + 10s.
10. Verify that the DUT sends a **RenewResponse** message.
11. Verify CurrentTime and TerminationTime (TerminationTime(Response) >= TerminationTime(Request) and CurrentTime(Response) <= TerminationTime(Response)).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime.
- The DUT did not send a **RenewResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **RenewResponse** message (see Annex A.3).

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

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

Note: If DUT cannot accept the set value to a TerminationTime, ONVIF Client retries to send the Renew request MinimumTime value included in UnacceptableTerminationTimeFault.

5.2.3 BASIC NOTIFICATION INTERFACE - NOTIFY

Test Case ID: EVENT-2-1-17

Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Feature Under Test: Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

Test Purpose: To verify **Notify** message

Pre-Requisite: The DUT shall provide at least one event. The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property events or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **Subscribe** request with an InitialTerminationTime of 60s to ensure that the SubscriptionManager is deleted after one minute.
4. Verify that the DUT sends a **SubscribeResponse** with valid values for SubscriptionReference, TerminationTime and CurrentTime TerminationTime >= CurrentTime + InitialTerminationTime.
5. Invoke **SetSynchronizationPoint** request to trigger a property event.
6. Verify that DUT sends **SetSynchronizationPointResponse**.
7. If the DUT does not support property events, the test operator has to trigger an event manually.
8. Verify that DUT sends **Notify** message(s).

9. Verify received **Notify** messages (correct value for UTC time, TopicExpression and wsnt:Message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **SetSynchronizationPointResponse** message.
- The DUT did not a **Notify** message.
- The DUT sends an invalid **Notify** message.
- DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).
- DUT did not send valid WS-Addressing Action URI in SOAP Header for **SetSynchronizationPointResponse** message (see Annex A.3).
- DUT did not send valid WS-Addressing Action URI in SOAP Header for **Notify** message (see Annex A.3).
- DUT did not send WS-Addressing MessageID SOAP Header if WS-Addressing ReplyTo is included in **Notify** message.
- DUT sent invalid WS-Addressing MessageID.
- DUT use wrong HTTP address for **Notify** message.

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

Note: See Annex A.1 for instructions on how to construct Subscribe when it is required for the ONVIF Client to receive all events supported by the DUT.

5.2.4 BASIC NOTIFICATION INTERFACE - NOTIFY FILTER

Test Case ID: EVENT-2-1-18

Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Feature Under Test: GetEventProperties, Subscribe, Unsubscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

Test Purpose: To verify that the device sends Notification messages; to verify if the DUT handles event filtering in a correct way.

Pre-Requisite: The DUT shall provide at least one event. The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property event or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger the event manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request.
4. Verify that the DUT sends a **GetEventPropertiesResponse** message, select one Topic.
5. ONVIF Client will invoke **Subscribe** request with this Topic as Filter and an InitialTerminationTime of 60s to ensure that the SubscriptionManager is deleted after one minute.
6. Verify that the DUT sends a **SubscribeResponse** message with valid values for SubscriptionReference, TerminationTime and CurrentTime TerminationTime >= CurrentTime + 60S.
7. Invoke **SetSynchronizationPoint** request to trigger an event.
8. Verify that the DUT sends **SetSynchronizationPointResponse** message.
9. If the DUT does not support property events, the operator has to trigger an event manually.
10. Verify that the DUT sends **Notify** message(s).
11. Check that at least one **Notify** message that contains a property event is returned. Verify this **Notify** messages (correct value for Utc time, TopicExpression and wsnt:Message).
12. Check if **Notify** message(s) are filtered according to the selected Filter.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse** message.
- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send a **SetSynchronizationPointResponse** message.
- The DUT did not a **Notify** message that contains a property event
- The DUT send an invalid **Notify** message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **GetEventPropertiesResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SetSynchronizationPointResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **Notify** message (see Annex A.3).
- DUT did not send WS-Addressing MessageID in SOAP Header if WS-Addressing ReplyTo is included in **Notify** message.
- DUT sent invalid WS-Addressing MessageID in SOAP Header in **Notify** message.
- DUT used wrong HTTP address for **Notify** message.

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

5.2.5 BASIC NOTIFICATION INTERFACE - INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-2-1-19

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or "InvalidMessageContentExpressionFault" is returned if a Subscribe Request with an invalid MessageContentFilter is invoked.

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 **Subscribe** request with an invalid Filter (boolean (// tt:SimpleItem[@Name="xyz" and @Value="xyz"])).
4. Verify that the DUT generates an "InvalidFilterFault" or an "InvalidMessageContentExpressionFault" fault message.
5. Validate the fault message (valid UTC time, valid description).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send an "InvalidFilterFault" or "InvalidMessageContentExpressionFault" fault message.
- The DUT did not send a valid fault message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

5.2.6 BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE (NEGATIVE TEST)

Test Case ID: EVENT-2-1-21

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe, Unsubscribe

WSDL Reference: event.wsdl

Test Purpose: To verify Unsubscribe 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 **Subscribe** request (InitialTerminationTime = PT10S) to instantiate a SubscriptionManager.
4. Verify that the DUT sends a **SubscribeResponse** message with valid values for SubscriptionManager, CurrentTime and TerminationTime.
5. ONVIF Client will invoke **Unsubscribe** request.
6. Verify that the DUT sends an **UnsubscribeResponse** message.
7. ONVIF Client will invoke a **Renew** request to verify that the SubscriptionManager is deleted.
8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a "ResourceUnknown" fault message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime >= CurrentTime + InitialTerminationTime).
- The DUT did not send an **UnsubscribeResponse** message.
- The DUT did not send a Soap 1.2 fault message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **UnsubscribeResponse** message (see Annex A.3).

- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

5.2.7 BASIC NOTIFICATION INTERFACE - RESOURCE UNKNOWN

Test Case ID: EVENT-2-1-22

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe, Unsubscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that a Soap 1.2 Fault Message is returned if an UnsubscribeRequest to a non-existing Subscription is sent.

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 **Subscribe** request with an InitialTerminationTime of 10 s to ensure that the SubscriptionManager will be deleted after testing.
4. Verify that the DUT sends a **SubscribeResponse** message with valid values for SubscriptionReference and TerminationTime).
5. ONVIF Client will invoke **Unsubscribe** request to delete the SubscriptionManager.
6. Verify that the DUT sends an **UnsubscribeResponse** message.
7. Send the second Unsubscribe Request to the just deleted SubscriptionReference.
8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a "ResourceUnknown" or a "UnableToDestroySubscription" Fault).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid SubscriptionReference.
- The DUT did not send an **UnsubscribeResponse** message.
- The DUT did not delete the SubscriptionManager.
- The DUT did not send a Soap 1.2 Fault.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **UnsubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

5.2.8 BASIC NOTIFICATION INTERFACE - INVALID TOPIC EXPRESSION

Test Case ID: EVENT-2-1-23

Specification Coverage: Basic Notification Interface

Feature Under Test: Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" is returned if a Subscribe Request with an invalid Topic Expression is invoked.

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 **Subscribe** request with an invalid Topic Expression (boolean(//tt:SimpleItem[@Name="xyz" and @Value="xyz"])).
4. Verify that the DUT generates an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.
5. Validate the fault message (valid UTC time, valid description).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.
- The DUT did not send a valid fault message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

5.2.9 BASIC NOTIFICATION INTERFACE - SET SYNCHRONIZATION POINT

Test Case ID: EVENT-2-1-24

Specification Coverage: Basic Notification Interface, SetSynchronizationPoint

Feature Under Test: Subscribe, SetSynchronizationPoint, Notify

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT sends properties to the Client when SetSynchronizationPoint operation is invoked.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain topic with MessageDescription with IsProperty = true, skip other steps.
6. ONVIF Client sets the following
   - *topic* := the Event topic provided on Management Tab
7. ONVIF Client invokes **Subscribe** request with parameters
   - ConsumerReference := NotificationConsumer interface of the ONVIF Client
   - Filter.TopicExpression := *topic*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
   - InitialTerminationTime := PT60S
8. The DUT responds with **SubscribeResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime

- TerminationTime

9. Until *timeout1* timeout expires:

   9.1. The DUT sends **Notify** message with parameters:

   - NotificationMessage list := *notificationList*

   9.2. If *notificationList* contains NotificationMessage (*notification*) with Message.Message.@PropertyOperation = Initialized, ONVIF Client sets the following and goes to step 10:

   - *notificationList*[0].NotificationMessage = *notification*, where NotificationMessage is the first NotificationMessage with Message.Message.@PropertyOperation = Initialized

   9.3. If *timeout1* expires for step 9.1 without **Notify** message that contains NotificationMessage with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.

10. If *notification*.Topic value is not equal to *topic*, FAIL the test and skip other steps.

11. ONVIF Client invokes **SetSynchronizationPoint**.

12. The DUT responds with **SetSynchronizationPointResponse** message.

13. Until *timeout1* timeout expires:

    13.1. The DUT sends **Notify** message with parameters:

    - NotificationMessage list := *notificationList*

    13.2. If *notificationList* contains the following NotificationMessage (*repeatedNotification*), go to step 14:

    - *repeatedNotification*.Message.Message.@PropertyOperation = Initialized
    - *repeatedNotification*.Topic = *topic*
    - *repeatedNotification*.Message.Message.Source item is equal to *notification*.Message.Message.Source item if *notificationMessage*.Message.Message contains not empty Source element
    - *repeatedNotification*.Message.Message.Key item is equal to *notification*.Message.Message.Key item If *notificationMessage*.Message.Message contains not empty Key element

13.3. If *timeout1* expires for step 13.1 without **Notify** message that is corresponding to step 13.2, FAIL the test and skip other steps.

14. If subscription timeout is not expired, ONVIF Client invokes **Unsubscribe** request to the subscription endpoint *s*.

15. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.
- The DUT returned **GetEventPropertiesResponse** without topic with IsProperty = true.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send **SetSynchronizationPointResponse** message.
- The DUT did not send **UnsubscribeResponse message** message.

Note: ONVIF Client shall use Renew request in order to postpone the termination time.

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

5.2.10 BASIC NOTIFICATION INTERFACE – CONJUNCTION IN NOTIFY FILTER (OR OPERATION)

Test Case ID: EVENT-2-1-25

Specification Coverage: Topic Filter (ONVIF Core Specification), Basic Notification Interface (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports "or" operation in conjunction of topics in the Topic Expressions.

Pre-Requisite: If the DUT does not support at least two property events, the test operator has to provide two topics for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* contains less than 2 topics, skip other steps.
6. ONVIF Client sets the following
   - *topic1* := the first Event topic provided on Management Tab
   - *topic2* := the second Event topic provided on Management Tab
7. ONVIF Client invokes **Subscribe** request with parameters
   - ConsumerReference := NotificationConsumer interface of the ONVIF Client
   - Filter.TopicExpression := *topic1*|*topic2*
   - Filter.TopicExpression.@Dialect := http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
   - InitialTerminationTime := PT60S
8. The DUT responds with **SubscribeResponse** message with parameters
   - SubscriptionReference =: *s*

- CurrentTime =: *ct*
- TerminationTime =: *tt*

9. Until *timeout1* timeout expires, repeat the following steps:

   9.1. The DUT sends **Notify** message with parameters:

   - NotificationMessage list := *notificationList*

   9.2. For each NotificationMessage (*notification*) in *notificationList*

   9.2.1. If *notification*.Topic value is not equal to *topic1* or *topic2*, FAIL the test and skip other steps.

   9.3. If NotificationMessage with topic value is equal to *topic1* and NotificationMessage with topic value is equal to *topic2* were received, go to step 10.

   9.4. If *timeout1* expires for step 9.2 without NotificationMessage with topic value equals to *topic1* or without NotificationMessage with topic value equals to *topic2*, FAIL the test and skip other steps.

10. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property enents if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least 2 topics with IsPropery = true, ONVIF Client provides user interaction window at step 9 and Test Operator has to generate events manually.

5.2.11 BASIC NOTIFICATION INTERFACE – TOPIC SUB-TREE IN PULLMESSAGES FILTER

Test Case ID: EVENT-2-1-26

Specification Coverage: Topic Filter (ONVIF Core Specification), Basic Notification Interface (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports "//." operation in topic filter in the Topic Expressions.

Pre-Requisite: If the DUT does not support property event, the test operator has to provide topic for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. ONVIF Client sets the following
   - *topic* := the first Event topic provided on Management Tab
   - *root* := root element of *topic*

6. ONVIF Client invokes **Subscribe** request with parameters
   - ConsumerReference := NotificationConsumer interface of the ONVIF Client
   - Filter.TopicExpression := *root*//.
   - Filter.TopicExpression.@Dialect := http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet
   - InitialTerminationTime := PT60S
7. The DUT responds with **SubscribeResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
8. Until *timeout1* timeout expires:
   8.1. The DUT sends **Notify** message with parameters:
      - NotificationMessage list := *notificationList*

   8.2. For each NotificationMessage (*notification*) in *notificationList*

      8.2.1. If root element of *notification*.Topic is not equal to *root*, FAIL the test and skip other steps.

   8.3. If NotificationMessage with topic value is equal to *topic* was received, go to step 9.

   8.4. If *timeout1* expires for step 8.2 without NotificationMessage with topic value equals to *topic*, FAIL the test and skip other steps.
9. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.
10. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property event if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least one topic with IsPropery = true, ONVIF Client provides user interaction window and Test Operator has to generate events manually.

5.2.12 BASIC NOTIFICATION INTERFACE – CONJUNCTION IN NOTIFY FILTER (TOPIC SUB-TREE AND OR OPERATION)

Test Case ID: EVENT-2-1-27

Specification Coverage: Topic Filter (ONVIF Core Specification), Basic Notification Interface (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports combination of "or" operation and "//." operation in topic filter in the Topic Expressions.

Pre-Requisite: If the DUT does not support at least two property events, the test operator has to provide two topics for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

- TopicSet =: *topicSet*
- TopicExpressionDialect list
- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. If *topicSet* contains less than 2 topics, skip other steps.
6. ONVIF Client sets the following
   - *topic1* := the first Event topic provided on Management Tab
   - *topic2* := the second Event topic provided on Management Tab
   - *root1* := root element of *topic1*
7. ONVIF Client invokes **Subscribe** request with parameters
   - ConsumerReference := NotificationConsumer interface of the ONVIF Client
   - Filter.TopicExpression := *root1*//|*topic2*.
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with **SubscribeResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
9. Until *timeout1* timeout expires:
   - 9.1. The DUT sends **Notify** message with parameters:
     - NotificationMessage list := *notificationList*
   - 9.2. For each NotificationMessage (*notification*) in *notificationList*
     - 9.2.1. If root element of *notification*.Topic is not equal to *root1* or if *notification*.Topic is not equal to *topic2*, FAIL the test and skip other steps.
   - 9.3. If NotificationMessage with topic value is equal to *topic1* and NotificationMessage with topic value is equal to *topic2* were received, go to step 10.

9.4. If *timeout1* expires for step 9.2 without NotificationMessage with topic value equals to *topic1* or without NotificationMessage with topic value equals to *topic2*, FAIL the test and skip other steps.

10. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property events if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least 2 topics with IsPropery = true, ONVIF Client provides user interaction window at step 9 and Test Operator has to generate events manually.

5.2.13 BASIC NOTIFICATION INTERFACE - UNSUBSCRIBE

Test Case ID: EVENT-2-1-28

Specification Coverage: Basic Notification Interface

Feature Under Test: Unsubscribe

WSDL Reference: event.wsdl

Test Purpose: To verify DUT Unsubscribe 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 invokes **Subscribe** with parameters:
   - InitialTerminationTime = 60 second
4. The DUT responds with **SubscribeResponse** messagewith parameters:
   - SubscriptionReference =: *s*
   - CurrentTime
   - TerminationTime
5. Set *timeout1* := "PT1S"
6. Wait until *timeout1* Timeout expires.
7. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
8. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

5.2.14 BASIC NOTIFICATION INTERFACE - MESSAGE CONTENT FILTER

Test Case ID: EVENT-2-1-29

Specification Coverage: Subscribe, Message Content Filter

Feature Under Test: Subscribe

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports message content filter.

Pre-Requisite: Message Content Filter is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list := *messageContentFilterDialectList*
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain at least one topic with MessageDescription.@IsProperty = "true", skip other steps.
6. If *messageContentFilterDialectList* does not contain at least one item with non empty value, FAIL the test and skip other steps.
7. If *messageContentFilterDialectList* does not contain item with value is equal to "http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter", skip other steps.
8. ONVIF Client sets the following
   - *topic* := the Event topic provided on Management Tab
9. ONVIF Client invokes **Subscribe** request with parameters
   - ConsumerReference := NotificationConsumer interface of the ONVIF Client
   - Filter.TopicExpression := *topic*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
   - InitialTerminationTime := PT60S

10. The DUT responds with **SubscribeResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
11. Until *timeout1* timeout expires:
    11.1. The DUT sends **Notify** message with parameters:
    - NotificationMessage list := *notificationMessageList*

    11.2. If *notificationMessageList* contains NotificationMessage with Message.Message.@PropertyOperation = Initialized:
    - set *notification1* := the first NotificationMessage in *notificationMessageList* with Message.Message.@PropertyOperation = Initialized
    - Go to 12 [40] step.

    11.3. If *timeout1* expires for step 11 [40] without NotificationMessage message with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.
12. If *notification1*.Topic value is not equal to *topic*, FAIL the test and skip other steps.
13. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
14. The DUT responds with **UnsubscribeResponse** message.
15. ONVIF Client generates message content filter by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - in *notification1* - Notification message
    - out *messageContentFilter* - message content filter
    - out *name* - SimpleItem Name
    - out *value* - SimpleItem Value
16. ONVIF Client invokes **Subscribe** request with parameters
    - ConsumerReference := NotificationConsumer interface of the ONVIF Client
    - Filter.TopicExpression skipped

- Filter.MessageContent := *messageContentFilter*
- Filter.MessageContent.@Dialect := "http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter"
- InitialTerminationTime := PT60S

17. The DUT responds with **SubscribeResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
18. Until *timeout1* timeout expires, repeat the following steps:
    18.1. The DUT sends **Notify** message with parameters:
        - NotificationMessage list := *notificationMessageList*

    18.2. For each notification message (*notificationMessage*) in *notificationMessageList*:
        - If *notificationMessage*.Message.Message does not contain SimpleItem with Name = *name* and with Value = *value*, FAIL the test and skip other steps.

    18.3. If notificationMessageList contains notification with Topic = *topic* and with Message.Message.PropertyOperation="Initialized", go to 19 [41] step.

    18.4. If *timeout1* expires for step 18 without NotificationMessage message with Topic = *topic* with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.
19. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
20. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.
- The DUT returned **GetEventPropertiesResponse** message with empty TopicSet.

FAIL –

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

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

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

5.3 Real-Time Pull-Point Notification Interface

5.3.1 REALTIME PULLPOINT SUBSCRIPTION - CREATE PULL POINT SUBSCRIPTION

Test Case ID: EVENT-3-1-9

Specification Coverage: CreatePullPointSubscription

Feature Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify the DUT CreatePullPointSubscription 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 **CreatePullPointSubscription** request to instantiate a Subscription Manager. The **CreatePullPointSubscription** request does not contain a TopicExpression or Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the SubscriptionManager is terminated after end of this test.
4. Verify that the DUT sends **CreatePullPointSubscriptionResponse** message.
5. Validate that valid values for SubscriptionReference CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not return a valid SubscriptionReference.
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).

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

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

5.3.2 REALTIME PULLPOINT SUBSCRIPTION - RENEW

Test Case ID: EVENT-3-1-12

Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Feature Under Test: CreatePullPointSubscription, Renew

WSDL Reference: event.wsdl

Test Purpose: To verify Renew command

Pre-Requisite: Profile A or Profile C or Profile Q or Profile S or Profile G is supported by the DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreatePullPointSubscription** request with an InitialTerminationTime of 10s.
4. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
5. Validate CurrentTime and TerminationTime (TerminationTime >= CurrentTime + 10s).
6. ONVIF Client will invoke **Renew** request with a TerminationTime of 10s to ensure that the Subscription times out after the test.

7. Verify that the DUT sends a **RenewResponse** message.
8. Verify CurrentTime and TerminationTime (TerminationTime >= CurrentTime + 10s).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime.
- The DUT did not send a **RenewResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime.
- DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- DUT did not send valid WS-Addressing Action URI in SOAP Header for **RenewResponse** message (see Annex A.3).

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

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

5.3.3 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES

Test Case ID: EVENT-3-1-15

Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

Feature Under Test: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT shall provide at least one event. The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the device does not support property events or if it is not possible to invoke a SetSynchronizationPoint, the test operator has to trigger event manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **CreatePullPointSubscription** request with a suggested timeout of PT60S.
4. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
5. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
6. ONVIF Client will invoke **SetSynchronizationPoint** request to trigger an event.
7. Verify that the DUT sends a **SetSynchronizationPointResponse** message.
8. If the DUT does not support property events, the operator has to trigger an event manually.
9. Verify that the DUT sends a **PullMessagesResponse** that contains at least one NotificationMessage that represents a property.
10. Verify NotificationMessage (a maximum number of 2 Notification Messages is included in the **PullMessagesResponse** message; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

- The DUT did not send a **SetSynchronizationPointResponse** message.
- The DUT did not send a **PullMessagesResponse** message.
- The **PullMessagesResponse** message does not contain a NotificationMessage.
- The **PullMessagesResponse** message contains more than 2 NotificationMessages.
- The NotificationMessages are not well formed.
- The **PullMessagesResponse** message contains invalid values for Current or TerminationTime.
- The DUT did not send a **PullMessagesFaultResponse** message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SetSynchronizationPointResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **PullMessagesResponse** message (see Annex A.3).

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

Note: It may be possible that some other events than the event which is being verified will be sent as PullMessages response during this test case. In such case, ONVIF Client should simply discard such response and they retry PullMessages request for the very event for verification.

Note: During test case execution, it should be guaranteed that the DUT should not delete the property event which is being used for verification.

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

Note: See Annex A.1 for instructions on how to construct Subscribe when it is required for the ONVIF Client to receive all events supported by the DUT.

5.3.4 REALTIME PULLPOINT SUBSCRIPTION - PULLMESSAGES FILTER

Test Case ID: EVENT-3-1-16

Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages, MessageFilter

Feature Under Test: GetEventProperties, CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify PullMessages command

Pre-Requisite: The DUT shall provide at least one event. The test operator has to ensure that the event is triggered and sent out. ONVIF Client will invoke a SetSynchronizationPoint request. If the DUT does not support property events of if it is not possible to invoke a SynchronizationPoint, the test operator has to trigger an event manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes GetEventProperties command to retrieve the supported Topics.
4. Verify that the DUT sends a **GetEventPropertiesResponse**; select one Topic
5. ONVIF Client will invoke **CreatePullPointSubscription** request with a suggested timeout of PT60S and a Filter including the selected Topic.
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
7. Validate CurrentTime and TerminationTime and SubscriptionReference.
8. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 2.
9. ONVIF Client will invoke **SetSynchronizationPoint** request to trigger a property event.
10. Verify that the DUT sends a **SetSynchronizationPointResponse** message.
11. If the DUT does not support property events, the operator has to trigger an event manually.
12. Verify that the DUT sends a **PullMessagesResponse** that contains at least one NotificationMessage.
13. Verify that that a maximum number of 2 Notification Messages is included in the PullMessages Response.

14. Verify that at least one property event is returned.
15. Verify that this NotificationMessage is well formed; Verify CurrentTime and TerminationTime (TerminationTime > CurrentTime).
16. Verify that the Topic of the NotificationMessage matches the filter.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send a **GetEventPropertiesResponse**
- The DUT did not send a valid **GetEventPropertiesResponse** (containing at least one Topic).
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
- The DUT did not send a **SetSynchronizationPointResponse** message.
- The DUT did not send a **PullMessagesResponse** message.
- The **PullMessagesResponse** message does not contain a NotificationMessage.
- The **PullMessagesResponse** message contains more than 2 NotificationMessages.
- The **PullMessagesResponse** message does not contain at least one event.
- The NotificationMessages are not well formed.
- The NotificationMessage contains to a topic that was not requested.
- The **PullMessagesResponse** message contains invalid values for Current or TerminationTime.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **GetEventPropertiesResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SetSynchronizationPointResponse** message (see Annex A.3).

- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **PullMessagesResponse** message (see Annex A.3).

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

Note: It may be possible that some other events than the event which is being verified will be sent as PullMessages response during this test case. In such case, ONVIF Client should simply discard such response and they retry PullMessages request for the very event for verification.

Note: During test case execution, it should be guaranteed that the DUT should not delete the property event which is being used for verification.

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

5.3.5 REALTIME PULLPOINT SUBSCRIPTION - INVALID MESSAGE CONTENT FILTER

Test Case ID: EVENT-3-1-17

Specification Coverage: CreatePullPointSubscription

Feature Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or "InvalidMessageContentExpressionFault" is returned if a CreatePullPointSubscription command with an invalid MessageContentFilter is invoked.

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 **CreatePullPointSubscription** request with an invalid Filter boolean(//tt:SimpleItem[@Name="xyz" and @Value="xyz"]).

4. Verify that the DUT generates an "InvalidFilterFault" or an "InvalidMessageContentExpressionFault" fault message. Validate that Utc time and description are correct.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send an "InvalidFilterFault" or "InvalidMessageContentExpressionFault" fault message.
- The DUT did not send a valid fault message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

5.3.6 REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE (NEGATIVE TEST)

Test Case ID: EVENT-3-1-19

Specification Coverage: Basic Notification Interface, CreatePullPointSubscription

Feature Under Test: CreatePullPointSubscription, Unsubscribe, Renew

WSDL Reference: event.wsdl

Test Purpose: To verify Unsubscribe 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 **CreatePullPointSubscription** request (InitialTerminationTime=PT10S) to instantiate a SubscriptionManager.

4. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate CurrentTime and TerminationTime.
5. ONVIF Client will invoke **Unsubscribe** request to terminate the SubscriptionManager.
6. Verify that the DUT sends an **UnsubscribeResponse** message.
7. ONVIF Client will invoke a **Unsubscribe** request to verify that the SubscriptionManager is deleted.
8. Verify that the DUT sends a Soap 1.2 Fault (e.g. a "ResourceUnknown" fault message).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime=CurrentTime + 10s).
- The DUT did not send an **UnsubscribeResponse** message.
- The DUT did not send a Soap 1.2 Fault.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **UnsubscribeResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

5.3.7 REALTIME PULLPOINT SUBSCRIPTION - TIMEOUT

Test Case ID: EVENT-3-1-20

Specification Coverage: CreatePullPointSubscription, Basic Notification Interface

Feature Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify if a SubscriptionManager times out correctly.

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 **CreatePullPointSubscription** request with a suggested timeout of PT20S.
4. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message.
5. Validate CurrentTime and TerminationTime and SubscriptionReference.
6. Wait for 20 s (SubscriptionManager timeout).
7. ONVIF Client will invoke **Unsubscribe** request to check if the SubscriptionManager is timed out.
8. Verify that the DUT sends a Soap 1.2 fault (e.g. a "ResourceUnknown" fault).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime >= CurrentTime + 10s).
- The DUT did not send a Soap 1.2 fault.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).

- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

Note: If DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the CreatePullPointSubscription request with MinimumTime value included in UnacceptableInitialTerminationTime fault.

5.3.8 REALTIME PULLPOINT SUBSCRIPTION - INVALID TOPIC EXPRESSION

Test Case ID: EVENT-3-1-22

Specification Coverage: CreatePullPointSubscription

Feature Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that a correct error message "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" is returned if a CreatePullPointSubscription command with an invalid TopicExpression is invoked.

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 **CreatePullPointSubscription** request with an invalid Topic Expression "xyz".
4. Verify that the DUT generates an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.
5. Validate valid the fault message (valid Utc time, valid description).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send an "InvalidFilterFault" or "TopicNotSupported" or "InvalidTopicExpressionFault" fault message.
- The DUT did not send a valid fault message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for Fault message (see Annex A.3).

5.3.9 REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES AS KEEP-ALIVE

Test Case ID: EVENT-3-1-24

Specification Coverage: Realtime PullPoint Notification Interface, CreatePullPointSubscription, PullMessages

Feature Under Test: CreatePullPointSubscription, PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify PullMessages command as the one being kept alive.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request to retrieve the supported Topics.
4. Verify that the DUT sends a **GetEventPropertiesResponse** message.
5. ONVIF Client will invoke **CreatePullPointSubscription** message (no filter, no InitialTerminationTime).
6. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate CurrentTime and TerminationTime. Mark Termination Time as T1. Mark Current Time as C1.
7. ONVIF Client waits for 1s.
8. ONVIF Client will invoke **PullMessages** request with a PullMessages Timeout of [T1 – C1 + 20s] and a MessageLimit of 1 to update termination time of subscription.

9. Verify **PullMessagesResponse** message or **PullMessagesFaultResponse** from the DUT.
10. If **PullMessagesResponse** is received from the DUT then ONVIF Client will invoke **PullMessages** request with a PullMessages Timeout equal to MaxTimeout from **PullMessagesFaultResponse** message and a MessageLimit of 1 to update termination time of subscription. Otherwise, go to step 11.
11. Verify **PullMessagesResponse** message from the DUT.
12. Validate Current Time and Termination Time. Check that TerminationTime > CurrentTime. Check that TerminationTime > T1.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime.
- The DUT did not send a **PullMessagesResponse** message or **PullMessagesFaultResponse** message at step 6.
- The DUT did not send a **PullMessagesResponse** message at step 8.
- The **PullMessagesResponse** message contained invalid values for Current or TerminationTime.
- The **PullMessagesResponse** message to the **PullMessages** request with a PullMessages Timeout of [T1 – C1 + 20s] contained TerminationTime value less or equal to T1.

Note: If the DUT does not update termination time after PullMessagesRequest, then the ONVIF Client passes the test with a warning.

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

Note: If the DUT does not support property events, then test operator will send event manually (user interaction window will be used).

Note: To avoid long time of test execution, the test tool waits PullMessagesResponse during Operation delay timeout time. If the DUT does not send PullMessagesResponse during Operation delay timeout time, the test tool terminates subscription and fails the test.

5.3.10 REALTIME PULLPOINT SUBSCRIPTION - SET SYNCHRONIZATION POINT

Test Case ID: EVENT-3-1-25

Specification Coverage: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

Feature Under Test: CreatePullPointSubscription, SetSynchronizationPoint, PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT sends properties to the Client when SetSynchronizationPoint operation is invoked.

Pre-Requisite: None.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain topic with MessageDescription with IsProperty = true, skip other steps.
6. ONVIF Client sets the following
   - *topic* := the Event topic provided on Management Tab
7. ONVIF Client invokes **CreatePullPointSubscription** request with parameters

- Filter.TopicExpression := *topic*
- Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"

8. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
9. Until *timeout1* timeout expires, repeat the following steps:
   9.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   9.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   9.3. The DUT responds with **PullMessagesResponse** message with parameters:
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   9.4. If *notificationMessageList* contains NotificationMessage with Message.Message.@PropertyOperation = Initialized set the following:
      - *notification* := the first NotificationMessage in *notificationMessageList* with Message.Message.@PropertyOperation = Initialized
   9.5. If *timeout1* expires for step 9 without NotificationMessage message with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.
10. If *notification*.Topic value is not equal to topic, FAIL the test and skip other steps.
11. ONVIF Client invokes **SetSynchronizationPoint**.
12. The DUT responds with **SetSynchronizationPointResponse** message.

13. Until *timeout1* timeout expires:

    13.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

    13.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

    - Timeout := PT60S
    - MessageLimit := 1

    13.3. The DUT responds with **PullMessagesResponse** message with parameters:

    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage list =: *notificationMessageList*

    13.4. If *notificationMessageList* contains the following NotificationMessage (*repeatedNotification*), go to step 14:

    - *repeatedNotification*.Message.Message.@PropertyOperation = Initialized
    - *repeatedNotification*.Topic = *topic*
    - *repeatedNotification*.Message.Message.Source item is equal to *notification*.Message.Message.Source item if *notificationMessage*.Message.Message contains not empty Source element
    - *repeatedNotification*.Message.Message.Key item is equal to *notification*.Message.Message.Key item If *notificationMessage*.Message.Message contains not empty Key element

    13.5. If *timeout1* expires for step 13 without *repeatedNotification* Notification, FAIL the test and skip other steps.

14. If subscription timeout is not expired, ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.

15. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.
- The DUT returned **GetEventPropertiesResponse** message with empty TopicSet.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send a **SetSynchronizationPointResponse** message.
- The DUT did not send a **PullMessagesResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

5.3.11 REALTIME PULLPOINT SUBSCRIPTION – PULLMESSAGES TIMEOUT

Test Case ID: EVENT-3-1-32

Specification Coverage: PullMessages

Feature Under Test: PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts 60s timeout value in PullMessages request.

Pre-Requisite: None

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
   - Filter skipped
4. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
5. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
   - Timeout := PT60S

- MessageLimit := 1

6. The DUT responds with **PullMessagesResponse** message with parameters:
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list
7. If *tt* is not more than *ct*, FAIL the test and skip other steps.
8. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.
9. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

5.3.12 REALTIME PULLPOINT SUBSCRIPTION – CONJUNCTION IN PULLMESSAGES FILTER (OR OPERATION)

Test Case ID: EVENT-3-1-33

Specification Coverage: Topic Filter (ONVIF Core Specification), Create pull point subscription (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports "or" operation in conjunction of topics in the Topic Expressions.

Pre-Requisite: If the DUT does not support at least two property events, the test operator has to provide two topics for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* contains less than 2 topics, skip other steps.
6. ONVIF Client sets the following
   - *topic1* := the first Event topic provided on Management Tab
   - *topic2* := the second Event topic provided on Management Tab
7. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
   - Filter.TopicExpression := *topic1*|*topic2*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*

- TerminationTime =: *tt*

9. Until *timeout1* timeout expires, repeat the following steps:

   9.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

   9.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

   - Timeout := PT60S
   - MessageLimit := 1

   9.3. The DUT responds with **PullMessagesResponse** message with parameters:

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   9.4. For each NotificationMessage (*notification*) in *notificationMessageList*

   9.4.1. If *notification*.Topic value is not equal to *topic1* or *topic2*, FAIL the test and skip other steps.

   9.5. If NotificationMessage with topic value is equal to *topic1* and NotificationMessage with topic value is equal to *topic2* were received, go to step 10.

   9.6. If *timeout1* expires for step 9 without NotificationMessage with topic value equals to *topic1* or without NotificationMessage with topic value equals to *topic2*, FAIL the test and skip other steps.

10. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message

- The DUT did not send a **UnsubscribeResponse**.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property enents if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least 2 topics with IsPropery = true, ONVIF Client provides user interaction window at step 10 and Test Operator has to generate events manually.

5.3.13 REALTIME PULLPOINT SUBSCRIPTION – TOPIC SUB-TREE IN PULLMESSAGES FILTER

Test Case ID: EVENT-3-1-34

Specification Coverage: Topic Filter (ONVIF Core Specification), Create pull point subscription (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports "//." operation in topic filter in the Topic Expressions.

Pre-Requisite: If the DUT does not support property event, the test operator has to provide topic for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

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

- MessageContentFilterDialect list
- MessageContentSchemaLocation list

5. ONVIF Client sets the following
   - *topic* := the Event topic provided on Management Tab
   - *root* := root element of *topic*
6. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
   - Filter.TopicExpression := *root*//.
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
7. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
8. Until *timeout1* timeout expires, repeat the following steps:
   8.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
   8.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
      - Timeout := PT60S
      - MessageLimit := 1
   8.3. The DUT responds with **PullMessagesResponse** message with parameters:
      - CurrentTime =: *ct*
      - TerminationTime =: *tt*
      - NotificationMessage list =: *notificationMessageList*
   8.4. For each NotificationMessage (*notification*) in *notificationMessageList*
      a. If root element of *notification*.Topic is not equal to *root*, FAIL the test and skip other steps.

8.5. If NotificationMessage with topic value is equal to *topic* was received, go to step 9.

8.6. If *timeout1* expires for step 8 without NotificationMessage with topic value equals to *topic*, FAIL the test and skip other steps.

9. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.

10. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property event if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least one topic with IsPropery = true, ONVIF Client provides user interaction window and Test Operator has to generate events manually.

5.3.14 REALTIME PULLPOINT SUBSCRIPTION – CONJUNCTION IN NOTIFY FILTER (TOPIC SUB-TREE AND OR OPERATION)

Test Case ID: EVENT-3-1-35

Specification Coverage: Topic Filter (ONVIF Core Specification), Create pull point subscription (ONVIF Core Specification)

Feature Under Test: Topic Filter, http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet TopicExpression Dialect, CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports combination of "or" operation and "//." operation in topic filter in the Topic Expressions.

Pre-Requisite: If the DUT does not support at least two property events, the test operator has to provide two topics for test on the Management tab and trigger the events manually.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
5. If *topicSet* contains less than 2 topics, skip other steps.
6. ONVIF Client sets the following
   - *topic1* := the first Event topic provided on Management Tab
   - *topic2* := the second Event topic provided on Management Tab
   - *root1* := root element of *topic1*
7. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
   - Filter.TopicExpression := *root*//.|*topic2*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
8. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters

- SubscriptionReference =: *s*
- CurrentTime =: *ct*
- TerminationTime =: *tt*

9. Until *timeout1* timeout expires, repeat the following steps:

   9.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.

   9.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters

   - Timeout := PT60S
   - MessageLimit := 1

   9.3. The DUT responds with **PullMessagesResponse** message with parameters:

   - CurrentTime =: *ct*
   - TerminationTime =: *tt*
   - NotificationMessage list =: *notificationMessageList*

   9.4. For each NotificationMessage (*notification*) in *notificationMessageList*

   9.4.1. If root element of *notification*.Topic is not equal to *root1* or if *notification*.Topic is not equal to *topic2*, FAIL the test and skip other steps.

   9.5. If NotificationMessage with topic value is equal to *topic1* and NotificationMessage with topic value is equal to *topic2* were received, go to step 10.

   9.6. If *timeout1* expires for step 9 without NotificationMessage with topic value equals to *topic1* or without NotificationMessage with topic value equals to *topic2*, FAIL the test and skip other steps.

10. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s*.

11. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **GetEventPropertiesResponse** message.
- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

Note: Test Operator has to select property event if supported by the Device on the Management tab in Event section.

Note: if *topicSet* does not contain at least 2 topics with IsPropery = true, ONVIF Client provides user interaction window at step 9 and Test Operator has to generate events manually.

5.3.15 REALTIME PULLPOINT SUBSCRIPTION - UNSUBSCRIBE

Test Case ID: EVENT-3-1-36

Specification Coverage: Unsubscribe

Feature Under Test: Unsubscribe

WSDL Reference: event.wsdl

Test Purpose: To verify the DUT Unsubscribe 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 invokes **CreatePullPointSubscription** request.
4. The DUT responds with a **CreatePullPointSubscriptionResponse** message with parameters
   - SubscriptionReference =: *s*
   - CurrentTime

- TerminationTime

5. Set *timeout1* := "PT1S"
6. Wait until *timeout1* Timeout expires.
7. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
8. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **UnsubscribeResponse** message.

5.3.16 REALTIME PULLPOINT SUBSCRIPTION – MAXIMUM SUPPORTED NUMBER OF NOTIFICATION PULL POINTS

Test Case ID: EVENT-3-1-37

Specification Coverage: Pull Point Lifecycle (ONVIF Core Specification), Create pull point subscription (ONVIF Core Specification)

Feature Under Test: CreatePullPointSubscription

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT creates a new pull point on each CreatePullPointSubscription command as long as the number of instantiated pull points does not exceed the capability MaxPullPoints.

Pre-Requisite: DUT supports MaxPullPoints capability feature as indicated by Capabilities.MaxPullPoints.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.

2. Start the DUT.
3. ONVIF Client retrieves Event Service capabilities by following the procedure mentioned in Annex A.7 with the following input and output parameters
   - out *cap* - Event Service capabilities
4. If *cap* does not contain MaxPullPoints, FAIL the test and skip other steps.
5. If *cap*.MaxPullPoints < 2, skip other steps.
6. ONVIF Client invokes **GetEventProperties** request.
7. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list
   - MessageContentSchemaLocation list
8. If *topicSet* does not contain at least one topic with MessageDescription with IsProperty = true, ONVIF Client will provide an User Interaction window on step 10.4.3. Test Operator shall provide any property event manually.
9. Set *countPullPoints* := 1.
10. Until *countPullPoints* < = *cap*.MaxPullPoints, repeat the following steps:

    10.1. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
    - Filter skipped
    - InitialTerminationTime skipped

    10.2. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s<n>*, where n is an index that is equal to *countPullPoints*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*

10.3. If *s<n>* = *s<n-1>*, close all subscriptions and FAIL the test.

10.4. Until *timeout1* timeout expires, repeat the following steps:

10.4.1. ONVIF Client waits for time *t* := min{(*tt-ct*)/2, 1 second}.

10.4.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s<n>* with parameters

- Timeout := PT60S
- MessageLimit := 1

10.4.3. The DUT responds with **PullMessagesResponse** message with parameters:

- CurrentTime =: *ct*
- TerminationTime =: *tt*
- NotificationMessage list =: *notificationMessageList*

10.4.4. If *notificationMessageList* is empty, go to step 10.4.

10.4.5. If *timeout1* expires for step 10 without NotificationMessage in *notificationMessageList*, FAIL the test and skip other steps.

10.5. Set *countPullPoints* := *countPullPoints* + 1.

11. For each subscription endpoint *s<n>*

11.1. ONVIF Client sends an **Unsubscribe** request to the subscription endpoint *s<n>*.

11.2. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send **PullMessagesResponse** message
- The DUT did not send a **UnsubscribeResponse**.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

5.3.17 REALTIME PULLPOINT SUBSCRIPTION - MESSAGE CONTENT FILTER

Test Case ID: EVENT-3-1-38

Specification Coverage: CreatePullPointSubscription, PullMessages, Message Content Filter

Feature Under Test: CreatePullPointSubscription, PullMessages

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT supports message content filter.

Pre-Requisite: Message Content Filter is supported by DUT.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client invokes **GetEventProperties** request.
4. The DUT responds with a **GetEventPropertiesResponse** message with parameters
   - TopicNamespaceLocation list
   - FixedTopicSet
   - TopicSet =: *topicSet*
   - TopicExpressionDialect list
   - MessageContentFilterDialect list := *messageContentFilterDialectList*
   - MessageContentSchemaLocation list
5. If *topicSet* does not contain at least one topic with MessageDescription.@IsProperty = "true", skip other steps.
6. If *messageContentFilterDialectList* does not contain at least one item with non empty value, FAIL the test and skip other steps.
7. If *messageContentFilterDialectList* does not contain item with value is equal to "http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter", skip other steps.
8. ONVIF Client sets the following

- *topic* := the Event topic provided on Management Tab

9. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
   - Filter.TopicExpression := *topic*
   - Filter.TopicExpression.@Dialect := "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
10. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
11. Until *timeout1* timeout expires, repeat the following steps:
    11.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
    11.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
    - Timeout := PT60S
    - MessageLimit := 1

    11.3. The DUT responds with **PullMessagesResponse** message with parameters:
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
    - NotificationMessage list =: *notificationMessageList*

    11.4. If *notificationMessageList* contains NotificationMessage with Message.Message.@PropertyOperation = Initialized:
    - set *notification1* := the first NotificationMessage in *notificationMessageList* with Message.Message.@PropertyOperation = Initialized
    - Go to 12 step.

    11.5. If *timeout1* expires for step 11 without NotificationMessage message with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.

12. If *notification1*.Topic value is not equal to *topic*, FAIL the test and skip other steps.
13. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.
14. The DUT responds with **UnsubscribeResponse** message.
15. ONVIF Client generates message content filter by following the procedure mentioned in Annex A.6 with the following input and output parameters
    - in *notification1* - Notification message
    - out *messageContentFilter* - message content filter
    - out *name* - SimpleItem Name
    - out *value* - SimpleItem Value
16. ONVIF Client invokes **CreatePullPointSubscription** request with parameters
    - Filter.TopicExpression skipped
    - Filter.MessageContent := *messageContentFilter*
    - Filter.MessageContent.@Dialect := "http://www.onvif.org/ver10/tev/messageContentFilter/ItemFilter"
17. The DUT responds with **CreatePullPointSubscriptionResponse** message with parameters
    - SubscriptionReference =: *s*
    - CurrentTime =: *ct*
    - TerminationTime =: *tt*
18. Until *timeout1* timeout expires, repeat the following steps:
    18.1. ONVIF Client waits for time *t* := min{(*tt*-*ct*)/2, 1 second}.
    18.2. ONVIF Client invokes **PullMessages** to the subscription endpoint *s* with parameters
        - Timeout := PT60S
        - MessageLimit := 1
    18.3. The DUT responds with **PullMessagesResponse** message with parameters:
        - CurrentTime =: *ct*

- TerminationTime =: *tt*
- NotificationMessage list =: *notificationMessageList*

18.4. If *notificationMessageList* is not empty:

- If *notificationMessageList* contains more than one notification item, FAIL the test and skip other steps.
- If *notificationMessageList*[0].Message.Message does not contain SimpleItem with Name = *name* and with Value = *value*, FAIL the test and skip other steps.
- If *notificationMessageList*[0].Topic = *topic* and *notificationMessageList*[0].Message.Message.PropertyOperation="Initialized", go to 19 [75] step.

18.5. If *timeout1* expires for step 18 without NotificationMessage message with Topic = *topic* with Message.Message.@PropertyOperation = Initialized, FAIL the test and skip other steps.

19. ONVIF Client invokes **Unsubscribe** to the subscription endpoint *s*.

20. The DUT responds with **UnsubscribeResponse** message.

Test Result:

PASS –

- DUT passes all assertions.
- The DUT returned **GetEventPropertiesResponse** message with empty TopicSet.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send a **PullMessagesResponse** message.

Note: *timeout1* will be taken from Operation Delay field of ONVIF Device Test Tool.

5.4 Namespace Handling

5.4.1 EVENT - NAMESPACES (DEFAULT NAMESPACES FOR EACH TAG)

Test Case ID: EVENT-4-1-6

Specification Coverage: None

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

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 **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the device sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not return a valid SubscriptionReference.
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only types of response shall be the same).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Note: Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have default namespaces definition in each tag (see examples in Annex A.2).

5.4.2 EVENT - NAMESPACES (DEFAULT NAMESPACES FOR PARENT TAG)

Test Case ID: EVENT-4-1-7

Specification Coverage: None

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

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 **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not return a valid SubscriptionReference.
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Note: Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have default namespaces definition in parent tag (see examples in Annex A.2).

5.4.3 EVENT - NAMESPACES (NOT STANDARD PREFIXES)

Test Case ID: EVENT-4-1-8

Specification Coverage: None

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

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 **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter.

The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.

6. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not return a valid SubscriptionReference
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Note: Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with not standard prefixes (see examples in Annex A.2).

5.4.4 EVENT - NAMESPACES (DIFFERENT PREFIXES FOR THE SAME NAMESPACE)

Test Case ID: EVENT-4-1-9

Specification Coverage: None

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

Pre-Requisite: Access Rules 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 will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that ONVIF Client sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

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

- The DUT did not return a valid SubscriptionReference
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only types of response shall be the same).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value which is contained in UnacceptableInitialTerminationTimeFault.

Note: Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with different prefixes for the same namespace (see examples in Annex A.2).

5.4.5 EVENT - NAMESPACES (THE SAME PREFIX FOR DIFFERENT NAMESPACES)

Test Case ID: EVENT-4-1-10

Specification Coverage: None

Feature Under Test: None

WSDL Reference: event.wsdl

Test Purpose: To verify that the DUT accepts requests for Event Service with different namespaces definition.

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 **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
4. Verify that the DUT sends **SubscribeResponse** message. Verify that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).
5. ONVIF Client will invoke **Subscribe** request to instantiate a Subscription Manager. The **Subscribe** request does not contain a TopicExpression or a Message Content Filter. The Message contains an InitialTerminationTime of 10s to ensure that the Subscription is terminated after end of this test.
6. Verify that the DUT sends **SubscribeResponse** message. Validate that a valid SubscriptionReference is returned (valid EndpointReference); verify that valid values for CurrentTime and TerminationTime are returned (TerminationTime >= CurrentTime + InitialTerminationTime).

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **SubscribeResponse** message.
- The DUT did not return a valid SubscriptionReference
- The DUT did not return valid values for CurrentTime and TerminationTime.
- The DUT returned different results at step 4 and step 6 (EndpointReference, TerminationTime, CurrentTime fields could be different, only type of response shall be the same).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SubscribeResponse** message (see Annex A.3).

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

Note: If the DUT cannot accept the set value to an InitialTerminationTime, ONVIF Client retries to send the Subscribe request with MinimumTime value included in UnacceptableInitialTerminationTimeFault.

Note: Everything that happens at step 4 shall happen at step 6 as well. Otherwise, it indicates that the DUT processes namespaces in a wrong way and the test shall be failed.

Note: All requests for steps 5-6 to the DUT shall have namespaces definition with the same prefixes for different namespaces (see examples in Annex A.2).

5.5 Capabilities

5.5.1 EVENT SERVICE CAPABILITIES

Test Case ID: EVENT-5-1-1

Specification Coverage: Capability exchange

Feature Under Test: GetServiceCapabilities (for Event Service)

WSDL Reference: events.wsdl

Test Purpose: To verify Event Service Capabilities of the DUT.

Pre-Requisite: Event 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 **GetServiceCapabilities** request to retrieve event service capabilities of the DUT.
4. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send valid **GetServiceCapabilitiesResponse** message.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **GetServiceCapabilitiesResponse** message (see Annex A.3).

5.5.2 GET SERVICES AND EVENT SERVICE CAPABILITIES CONSISTENCY

Test Case ID: EVENT-5-1-2

Specification Coverage: Capability exchange

Feature Under Test: GetServices, GetServiceCapabilities (for Event Service)

WSDL Reference: devicemgmt.wsdl, events.wsdl

Test Purpose: To verify Get Services and Events 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 **GetServices** request (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 **GetServiceCapabilities** request to retrieve Event service capabilities of the DUT.
6. Verify the **GetServiceCapabilitiesResponse** message from the DUT.

Test Result:

PASS –

- 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.

- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **GetServiceCapabilitiesResponse** message (see Annex A.3).

Note: Service will be defined as Event service if it has Namespace element that is equal to "http://www.onvif.org/ver10/events/wsdl".

5.6 Seek

5.6.1 SEEK EVENTS – BEGIN OF BUFFER

Test Case ID: EVENT-6-1-3

Specification Coverage: BeginOfBuffer (ONVIF Core Specification), Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Feature Under Test: Seek

WSDL Reference: event.wsdl

Test Purpose: To verify Seek function and that events are received from persistent notification storage when pull point was set before beginning of buffer. Verify tns1:EventBuffer/Begin event.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. ONVIF Client will invoke **GetEventProperties** request to retrieve all events supported by the DUT.
4. Verify the **GetEventPropertiesResponse** message from the DUT.
5. Check if there is an event with Topic tns1:EventBuffer/Begin. If there is no event with such Topic, skip other steps, fail the test and go to the next test.
6. Check that this event is not a Property event (MessageDescription.IsProperty = false).
7. ONVIF Client will invoke **CreatePullPointSubscription** request with TopicFilter = tns1:EventBuffer/Begin.

8. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.

9. ONVIF Client will invoke **Seek** request (UtcTime = [Current UTC Time], Reverse = true) to start reverse seek.

10. Verify that the DUT sends a **SeekResponse** message.

11. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.

12. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage.

13. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the **PullMessagesResponse** message; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

14. Verify that received event is event with Topic = tns1:EventBuffer/Begin.

15. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.

16. Verify that the DUT sends a **PullMessagesResponse** that contains no NotificationMessages.

17. ONVIF Client will invoke **Seek** request (UtcTime = Message.UtcTime of tns1:EventBuffer/Begin message from step 12, Reverse = false) to put pull point in the past.

18. Verify that the DUT sends a **SeekResponse** message.

19. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.

20. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage.

21. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the **PullMessagesResponse** message; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).

22. Verify that received event is event with Topic = tns1:EventBuffer/Begin.

23. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.

24. Verify that the DUT sends a **PullMessagesResponse** that contains no NotificationMessages.

25. Verify that there was **PullMessagesResponse** without messages.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
- The DUT did not send a **PullMessagesResponse** message.
- The DUT did not send a **GetEventsPropertiesResponse** message.
- The **PullMessagesResponse** message does not contain a NotificationMessage.
- The **PullMessagesResponse** message contains more than 1 NotificationMessages.
- The NotificationMessages are not well formed.
- The **PullMessagesResponse** message contains invalid values for Current or TerminationTime.
- The DUT did not send event with Topic = tns1:EventBuffer/Begin from persistent notification storage for step 12 and 20.
- The DUT sent event for step 16 and 24.
- The DUT did not return correct event with Topic = tns1:EventBuffer/Begin in GetEventsPropertiesResponse.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SetSynchronizationPointResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **PullMessagesResponse** message (see Annex A.3).

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

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

5.6.2 SEEK EVENTS

Test Case ID: EVENT-6-1-4

Specification Coverage: Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Feature Under Test: Seek

WSDL Reference: event.wsdl

Test Purpose: To verify Seek function and those events are received from persistent notification storage.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Find the beginning of buffer time BeginOfBufferTime1 (see Annex A.4)
4. ONVIF Client will invoke **CreatePullPointSubscription** request.
5. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
6. ONVIF Client will invoke **Seek** request (UtcTime = BeginOfBufferTime1 – 1s, no Reverse) to put pull point in the past.
7. Verify that the DUT sends a **SeekResponse** message.
8. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.

9. Verify that the DUT sends a **PullMessagesResponse**.
10. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the **PullMessagesResponse** message; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
11. Verify that Message.UtcTime is greater or equal to BeginOfBufferTime1.
12. Repeat steps 8-11 until Message.UtcTime is equal or greater than time of Pullpoint Subscrbtion creation (CurrentTime from step 5) or there are **PullMessagesResponse** without Notifications to get all messages from persistent notification storage.
13. Verify that the first notification has Topic = tns1:EventBuffer/Begin.
14. Verify that all events across **PullMessagesResponse** messages are ordered by increasing Message.UtcTime and if events occur in the incorrect order that they are only misplaced by a maximum of 5 seconds.
15. Verify that there is at least one message from persistent notification storage.
16. Verify that all Messages have Message.UtcTime that is greater or equal to BeginOfBufferTime1.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
- The DUT did not send a **PullMessagesResponse** message.
- The DUT did not send a **SeekResponse** message.
- The **PullMessagesResponse** message does not contain a NotificationMessage.
- The **PullMessagesResponse** message contains more than 1 NotificationMessages.
- The NotificationMessages are not well formed.
- The **PullMessagesResponse** message contains invalid values for Current or TerminationTime.

- The DUT did not return at least one NotificationMessage from persistent notification storage.
- The DUT did not return NotificationMessages from persistent notification storage ordered by increasing Message.UtcTime and events that occurred in the incorrect order were misplaced by more than 5 seconds.
- The DUT returned NotificationMessages from persistent notification storage that has Message.UtcTime < BeginOfBufferTime1.
- The DUT returned the first Notification with Topic which differs from tns1:EventBuffer/Begin.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SeekResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **PullMessagesResponse** message (see Annex A.3).

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

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

5.6.3 SEEK EVENTS – REVERSE

Test Case ID: EVENT-6-1-5

Specification Coverage: BeginOfBuffer (ONVIF Core Specification), Real-time Pull-Point Notification Interface (ONVIF Core Specification), Seek (ONVIF Core Specification), Persistent notification storage (ONVIF Core Specification)

Feature Under Test: Seek

WSDL Reference: event.wsdl

Test Purpose: To verify Seek function and those events are received from persistent notification storage in reverse order. Verify tns1:EventBuffer/Begin event.

Pre-Requisite: Event Service was received from the DUT. Persistent notification storage is supported by the DUT. Persistent notification storage contains notifications.

Test Configuration: ONVIF Client and DUT

Test Procedure:

1. Start an ONVIF Client.
2. Start the DUT.
3. Find beginning of buffer time BeginOfBufferTime1 (see Annex A.4)
4. ONVIF Client will invoke **CreatePullPointSubscription** request.
5. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
6. ONVIF Client will invoke **Seek** request (UtcTime = [Current UTC Time], Reverse = «true») to start reverse seek.
7. Verify that the DUT sends a **SeekResponse** message.
8. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
9. Verify that the DUT sends a **PullMessagesResponse**.
10. Verify NotificationMessage (a maximum number of 1 Notification Messages is included in the **PullMessagesResponse** message; well formed and valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
11. Verify that Message.UtcTime is greater or equal to BeginOfBufferTime1.
12. Repeat steps 8-11 until Notification with Topic tns1:EventBuffer/Begin is received or there is **PullMessagesResponse** message without Notifications to get all messages from persistent notification storage.
13. Verify that the last notification has Topic = tns1:EventBuffer/Begin.
14. Verify that all events across **PullMessagesResponse** messages are ordered by decreasing Message.UtcTime and if events occur in the incorrect order that they are only misplaced by a maximum of 5 seconds.
15. Verify that there is at least one message from persistent notification storage.
16. Verify that all Messages have Message.UtcTime that is greater or equal to BeginOfBufferTime1.

Test Result:

PASS –

- DUT passes all assertions.

FAIL –

- The DUT did not send **CreatePullPointSubscriptionResponse** message.
- The DUT did not send valid values for CurrentTime and TerminationTime (TerminationTime > CurrentTime).
- The DUT did not send a **PullMessagesResponse** message.
- The DUT did not send a **SeekResponse** message.
- The **PullMessagesResponse** message does not contain a NotificationMessage.
- The **PullMessagesResponse** message contains more than 1 NotificationMessages.
- The NotificationMessages are not well formed.
- The **PullMessagesResponse** message contains invalid values for Current or TerminationTime.
- The DUT did not return at least one NotificationMessage from persistent notification storage.
- The DUT did not return NotificationMessages from persistent notification storage ordered by decreasing Message.UtcTime and events that occurred in the incorrect order were misplaced by more than 5 seconds.
- The DUT returned NotificationMessages from persistent notification storage that has Message.UtcTime < BeginOfBufferTime1.
- The DUT returned the last Notification with Topic which differs from tns1:EventBuffer/Begin.
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **CreatePullPointSubscriptionResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **SeekResponse** message (see Annex A.3).
- The DUT did not send valid WS-Addressing Action URI in SOAP Header for **PullMessagesResponse** message (see Annex A.3).

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

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

Annex A Helper Procedures and Additional Notes

A.1 Subscribe and CreatePullPointSubscription for Receiving All Events

When subscribing for events an ONVIF Client might be interested in receiving all or some of the Events produced by the DUT.

If an ONVIF Client is interested in receiving some events it includes a filter tag in the CreatePullPointSubscription or Subscribe requests describing the events which the ONVIF Client is interested in (see examples in the [ONVIF Core Specs]).

If an ONVIF Client is interested in receiving all events from a device it does not include the Filter sub tag in the Subscribe or CreatePullPointSubscription request.

Example:

The following Subscribe and CreatePullPointSubscription requests can be used if an ONVIF Client is interested in receiving all events.

1. Subscribe request:

```
<m:Subscribe xmlns:m="http://docs.oasis-open.org/wsn/b-2"
 xmlns:m0="http://www.w3.org/2005/08/addressing">
 <m:ConsumerReference>
 <m0:Address>
  http://192.168.0.1/events
 </m0:Address>
 </m:ConsumerReference>
</m:Subscribe>
```

2. CreatePullPointSubscription request

```
<m:CreatePullPointSubscription
 xmlns:m="http://www.onvif.org/ver10/events/wsdl"/>
```

A.2 Example of Requests for Namespaces Test Cases

For the execution of namespaces test cases, ONVIF Client shall send a request with specific namespaces definition. Examples of how this request shall look like are the following.

1. Defaults Namespaces Definition in Each Tag Examples

   Subscribe request example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
 <Header xmlns="http://www.w3.org/2003/05/soap-envelope">
 <Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/
  05/soap-envelope" xmlns="http://www.w3.org/2005/08/
  addressing">
  http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/
  SubscribeRequest</Action>
 <MessageID xmlns="http://www.w3.org/2005/08/addressing">
  urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc7e</MessageID>
 <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
  <Address xmlns="http://www.w3.org/2005/08/addressing">
  http://www.w3.org/2005/08/addressing/anonymous</Address>
 </ReplyTo>
 <Security xmlns="http://docs.oasis-open.org/wss/2004/
 01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <UsernameToken xmlns="http://docs.oasis-open.org/wss/2004/01/
  oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <Username xmlns="http://docs.oasis-open.org/wss/2004/01/
   oasis-200401-wss-wssecurity-secext-1.0.xsd">
   user</Username>
  <Password xmlns="http://docs.oasis-open.org/wss/2004/01/
   oasis-200401-wss-wssecurity-secext-1.0.xsd" Type=
   "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
   wss-username-token-profile-1.0#PasswordDigest">
   5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>
  <Nonce xmlns="http://docs.oasis-open.org/wss/2004/01/
   oasis-200401-wss-wssecurity-secext-1.0.xsd">
   ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>
  <Created xmlns="http://docs.oasis-open.org/wss/2004/01/
   oasis-200401-wss-wssecurity-utility-1.0.xsd">
   2011-04-25T09:27:48Z</Created>
```

```
 </UsernameToken>
</Security>
<To t:shallUnderstand="1"
 xmlns:t="http://www.w3.org/2003/05/soap-envelope"
 xmlns="http://www.w3.org/2005/08/addressing">
 http://169.254.141.200/onvif/services</To>
</Header>
<Body xmlns="http://www.w3.org/2003/05/soap-envelope">
<Subscribe xmlns="http://docs.oasis-open.org/wsn/b-2">
 <ConsumerReference
 xmlns="http://docs.oasis-open.org/wsn/b-2">
 <Address xmlns="http://www.w3.org/2005/08/addressing">
  http://192.168.10.66/onvif_notify_server</Address>
 </ConsumerReference>
 <InitialTerminationTime>PT10S</InitialTerminationTime>
</Subscribe>
</Body>
</Envelope>
```

2. Defaults Namespaces Definition in Parent Tag Examples

   Subscribe request example:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
 <Header>
 <Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/2003/
  05/soap-envelope"
  xmlns="http://www.w3.org/2005/08/addressing">
  http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/
  SubscribeRequest</Action>
 <MessageID xmlns="http://www.w3.org/2005/08/addressing">
  urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc7e</MessageID>
 <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
  <Address>http://www.w3.org/2005/08/addressing/
  anonymous</Address>
 </ReplyTo>
 <Security xmlns="http://docs.oasis-open.org/wss/2004/
  01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <UsernameToken>
```

```
 <Username>user</Username>
 <Password Type="http://docs.oasis-open.org/wss/2004/01/
  oasis-200401-wss-username-token-profile-1.0
  #PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</Password>
 <Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</Nonce>
 <Created xmlns="http://docs.oasis-open.org/wss/2004/01/
  oasis-200401-wss-wssecurity-utility-1.0.xsd">
  2011-04-25T09:27:48Z</Created>
 </UsernameToken>
</Security>
<To t:shallUnderstand="1"
 xmlns:t="http://www.w3.org/2003/05/soap-envelope"
 xmlns="http://www.w3.org/2005/08/addressing">
 http://169.254.141.200/onvif/services</To>
</Header>
<Body>
<Subscribe xmlns="http://docs.oasis-open.org/wsn/b-2">
 <ConsumerReference
 xmlns="http://docs.oasis-open.org/wsn/b-2">
 <Address xmlns="http://www.w3.org/2005/08/addressing">
  http://192.168.10.66/onvif_notify_server</Address>
 </ConsumerReference>
 <InitialTerminationTime xmlns="http://docs.oasis-open.org/
 wsn/b-2">PT10S</InitialTerminationTime>
</Subscribe>
</Body>
</Envelope>
```

3. Namespaces Definition with non-Standard Prefixes Examples

   Subscribe request example:

```
<prefix0:Envelope
 xmlns:prefix0="http://www.w3.org/2003/05/soap-envelope"
 xmlns:prefix1="http://www.w3.org/2003/05/soap-envelope"
 xmlns:prefix2="http://www.w3.org/2005/08/addressing"
 xmlns:prefix3="http://docs.oasis-open.org/wss/2004/01/
 oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:prefix4="http://docs.oasis-open.org/wss/2004/01/
```

```
oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:prefix5="http://docs.oasis-open.org/wsn/b-2">
<prefix0:Header>
<prefix2:Action prefix1:shallUnderstand="1">
 http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/
 SubscribeRequest</prefix2:Action>
<prefix2:MessageID>urn:uuid:9f2a12de-3a76-461b-a421-
 e472517bcc7e</prefix2:MessageID>
<prefix2:ReplyTo>
 <prefix2:Address>http://www.w3.org/2005/08/addressing/
 anonymous</prefix2:Address>
</prefix2:ReplyTo>
<prefix4:Security>
 <prefix4:UsernameToken>
 <prefix4:Username>service</prefix4:Username>
 <prefix4:Password Type="http://docs.oasis-open.org/wss/
  2004/01/oasis-200401-wss-username-token-profile-1.0
  #PasswordDigest">tg6EnHtyMWW8eUfntHO6XpPjOsg=
  </prefix4:Password>
 <prefix4:Nonce>iBIGpSuHtNPbdSWGzG48ng==</prefix4:Nonce>
 <prefix3:Created>2011-04-25T10:54:34Z</prefix3:Created>
 </prefix4:UsernameToken>
</prefix4:Security>
<prefix2:To prefix1:shallUnderstand="1">
 http://169.254.141.200/onvif/services</prefix2:To>
</prefix0:Header>
<prefix0:Body>
<prefix5:Subscribe>
 <prefix5:ConsumerReference>
 <prefix2:Address xmlns="http://www.w3.org/2005/08/
  addressing">http://192.168.10.66/onvif_notify_server
  </prefix2:Address>
 </prefix5:ConsumerReference>
 <prefix5:InitialTerminationTime>PT10S
 </prefix5:InitialTerminationTime>
</prefix5:Subscribe>
</prefix0:Body>
</prefix0:Envelope>
```

4. Namespaces Definition with Different Prefixes for the Same Namespace Examples

Subscribe request example:

```
<e1:Envelope xmlns:e1="http://www.w3.org/2003/05/soap-envelope">
 <e2:Header xmlns:e2="http://www.w3.org/2003/05/soap-envelope">
 <e3:Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/
  2003/05/soap-envelope" xmlns:e3="http://www.w3.org/2005/08/
  addressing">http://docs.oasis-open.org/wsn/bw-2/
  NotificationProducer/SubscribeRequest</e3:Action>
 <e4:MessageID xmlns:e4="http://www.w3.org/2005/08/addressing">
  urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc7e</e4:MessageID>
 <e5:ReplyTo xmlns:e5="http://www.w3.org/2005/08/addressing">
  <e6:Address xmlns:e6="http://www.w3.org/2005/08/addressing">
  http://www.w3.org/2005/08/addressing/anonymous</e6:Address>
 </e5:ReplyTo>
 <e7:Security xmlns:e7="http://docs.oasis-open.org/wss/2004/
  01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <e8:UsernameToken xmlns:e8="http://docs.oasis-open.org/wss/
  2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <e9:Username xmlns:e9="http://docs.oasis-open.org/wss/
   2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   user</e9:Username>
  <e10:Password xmlns:e10="http://docs.oasis-open.org/wss/
   2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type
   ="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
   wss-username-token-profile-1.0#PasswordDigest">
   5zjIbmvWxVevGlpqg6Qnt9h8Fmo=</e10:Password>
  <e11:Nonce xmlns:e11="http://docs.oasis-open.org/wss/
   2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   ikcoiK+AmJvA5UpfxTzG8Q==</e11:Nonce>
  <e12:Created xmlns:e12="http://docs.oasis-open.org/wss/
   2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   2011-04-25T09:27:48Z</e12:Created>
  </e8:UsernameToken>
 </e7:Security>
 <e13:To t:shallUnderstand="1"
  xmlns:t="http://www.w3.org/2003/05/soap-envelope"
  xmlns:e13="http://www.w3.org/2005/08/addressing">
  http://169.254.141.200/onvif/services</e13:To>
```

```
 </e2:Header>
 <e1:Body>
 <e14:Subscribe
  xmlns:e14="http://docs.oasis-open.org/wsn/b-2">
  <e15:ConsumerReference
  xmlns:e15="http://docs.oasis-open.org/wsn/b-2">
  <e16:Address
   xmlns:e16="http://www.w3.org/2005/08/addressing">
   http://192.168.10.66/onvif_notify_server</e16:Address>
  </e15:ConsumerReference>
  <e14:InitialTerminationTime>PT10S
  </e14:InitialTerminationTime>
 </e14:Subscribe>
 </e1:Body>
</e1:Envelope>
```

5. Namespaces Definition with the Same Prefixes for Different Namespaces Examples

   Subscribe request example:

```
<p1:Envelope xmlns:p1="http://www.w3.org/2003/05/soap-envelope">
 <p1:Header>
 <p1:Action u:shallUnderstand="1" xmlns:u="http://www.w3.org/
  2003/05/soap-envelope" xmlns:p1="http://www.w3.org/2005/08/
  addressing">http://docs.oasis-open.org/wsn/bw-2/
  NotificationProducer/SubscribeRequest</p1:Action>
 <p1:MessageID xmlns:p1="http://www.w3.org/2005/08/addressing">
  urn:uuid:9f2a12de-3a76-461b-a421-e472517bcc7e</p1:MessageID>
 <p1:ReplyTo xmlns:p1="http://www.w3.org/2005/08/addressing">
  <p1:Address>http://www.w3.org/2005/08/addressing/anonymous
  </p1:Address>
 </p1:ReplyTo>
 <p1:Security xmlns:p1="http://docs.oasis-open.org/wss/2004/
  01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <p1:UsernameToken>
  <p1:Username>user</p1:Username>
  <p1:Password Type="http://docs.oasis-open.org/wss/
   2004/01/oasis-200401-wss-username-token-profile-1.0
   #PasswordDigest">5zjIbmvWxVevGlpqg6Qnt9h8Fmo=
```

```
  </p1:Password>
 <p1:Nonce>ikcoiK+AmJvA5UpfxTzG8Q==</p1:Nonce>
 <p1:Created xmlns:p1="http://docs.oasis-open.org/wss/
  2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  2011-04-25T09:27:48Z</p1:Created>
 </p1:UsernameToken>
</p1:Security>
<p1:To t:shallUnderstand="1"
 xmlns:t="http://www.w3.org/2003/05/soap-envelope"
 xmlns:p1="http://www.w3.org/2005/08/addressing">
 http://169.254.141.200/onvif/services</p1:To>
</p1:Header>
<p1:Body>
<p1:Subscribe xmlns:p1="http://docs.oasis-open.org/wsn/b-2">
 <p1:ConsumerReference
 xmlns:p1="http://docs.oasis-open.org/wsn/b-2">
 <p1:Address
  xmlns:p1="http://www.w3.org/2005/08/addressing">
  http://192.168.10.66/onvif_notify_server</p1:Address>
 </p1:ConsumerReference>
 <p1:InitialTerminationTime xmlns:p1=
 "http://docs.oasis-open.org/wsn/b-2">PT10S
 </p1:InitialTerminationTime>
</p1:Subscribe>
 </p1:Body>
</p1:Envelope>
```

A.3 Action URI"s for Event Service Messages

The following Action URI"s shall be used for Event Service:

Table A.1. Action URI"s for Event Service Messages

Message	Action URI of WS-Addressing
Notify	http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify
SubscribeRequest	http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRequest
SubscribeResponse	http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeResponse

Message	Action URI of WS-Addressing
RenewRequest	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/RenewRequest
RenewResponse	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/RenewResponse
UnsubscribeRequest	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeRequest
UnsubscribeResponse	http://docs.oasis-open.org/wsn/bw-2/SubscriptionManager/UnsubscribeResponse
GetEventPropertiesRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest
GetEventPropertiesResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesResponse
CreatePullPointSubscriptionRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionRequest
CreatePullPointSubscriptionResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionResponse
PullMessagesRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest
PullMessagesResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesResponse
SetSynchronizationPointRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointRequest
SetSynchronizationPointResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SetSynchronizationPointResponse
GetServiceCapabilitiesResponse	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilities
GetServiceCapabilitiesRequest	http://www.onvif.org/ver10/events/wsdl/EventPortType/GetServiceCapabilitiesRequest
SeekRequest	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekRequest
SeekResponse	http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/SeekResponse

Message	Action URI of WS-Addressing
All faults	http://www.w3.org/2005/08/addressing/soap/fault

A.4 Find begin of buffer time

The following algorithm will be used to get a current begin of buffer time:

1. ONVIF Client will invoke **CreatePullPointSubscription** request with TopicFilter = tns1:EventBuffer/Begin.
2. Verify that the DUT sends a **CreatePullPointSubscriptionResponse** message. Validate that correct values for CurrentTime and TerminationTime and SubscriptionReference are returned.
3. ONVIF Client will invoke **Seek** request (UtcTime = [Current UTC Time], Reverse = true) to start a reverse seek.
4. Verify that the DUT sends a **SeekResponse** message.
5. ONVIF Client will invoke **PullMessages** command with a PullMessagesTimeout of 20s and a MessageLimit of 1.
6. Verify that the DUT sends a **PullMessagesResponse** that contains one NotificationMessage.
7. Verify that the received event is event with Topic = tns1:EventBuffer/Begin.

Note: Message.UtcTime from Notification message with Topic = tns1:EventBuffer/Begin will be used as BeginOfBufferTime1.

Note: if PullMessagesResponse will be empty or there will be no Notification message with Topic = tns1:EventBuffer/Begin, test cases shall be failed.

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

A.5 Get Event Service Capabilities

Name: HelperGetServiceCapabilities

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

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

Input: None

Returns: The service capabilities (*cap*).

Procedure:

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

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

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

A.6 Generate Message Content Filter

Name: HelperGenerateMessageContentFilter

Procedure Purpose: Helper procedure to generate message content filter for subscription.

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

Input: Notification message (*notificationMessage*)

Returns: Message content filter (*messageContentFilter*), SimpleItem Name *name*, SimpleItem Value *value*.

Procedure:

1. If *notificationMessage* does not contain Message.Message.Source item, FAIL the test and skip other steps.
2. Set *name* := *notificationMessage*.Message.Message.Source.SimpleItem[0].Name
3. Set *value* := *notificationMessage*.Message.Message.Source.SimpleItem[0].Value
4. Set *messageContentFilter* := "boolean(//tt:SimpleItem[@Name="{0}"and @Value="{1}"])", whre {0} = *name* and {1} = *value*.

Procedure Result:

PASS –

- DUT passes all assertions.

FAIL –

- None.